

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belagavi-590 014



A Dissertation Report on
CNN Based Security Authentication for Wireless Multimedia
Networks

*submitted as a partial fulfillment of the requirements
for the award of the degree*

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

By

Gautham S K
1BY19SCS02

Under the guidance

Dr. Anjan Krishnamurthy,
Associate Professor & PG Coordinator



Department of Computer Science and Engineering
B M S Institute of Technology and Management
BENGALURU - 560064

2021

B M S Institute of Technology and Management

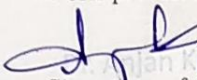
(Affiliated to VTU, Belagavi)

BENGALURU - 560064

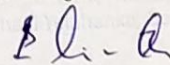


CERTIFICATE

Certified that the project work entitled "CNN Based Security Authentication for Wireless Multimedia Networks" has been carried out by **Gautham S K (1BY19SCS02)**, a bonafide student of BMS Institute of Technology and Management, Bengaluru as a partial fulfillment for the award of Master of Technology in Computer Science and Engineering Visvesvaraya Technological University, Belagavi during the year 2020-2021. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project work report has been approved as it satisfies the academic requirement in respect of interim project work prescribed for the said degree.


28/7/2021
Signature of Guide

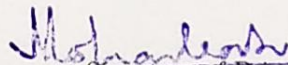
(Dr Anjan Krishnamurthy)


Signature of Head of Department
Professor and Head
Department of Computer Science & Engineering
BMS Institute of Technology & Management
Yelahanka, Bangalore- 560 064

Name of Examiner

1:

2:


Signature of Principal
PRINCIPAL
BMS Institute of Technology & Management
(Dr. Mohan Babu G.N)
Avalahalli, Yelahanka, Bengaluru - 56

Signature of Examiner

DECLARATION

I, Gautham SK, student of fourth semester M.Tech, in the Department of Computer Science and Engineering, B M S Institute of Technology and Management, Bengaluru declare that the project work entitled “CNN Based Security Authentication for Wireless Multimedia Networks” has been carried out by me and submitted in partial fulfillment of the course requirements for the award of degree in Master of Technology in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2019 -2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Gautham SK

1BY19SCS02

MTech in CSE Department of Computer Science and Engineering,

B M S Institute of Technology and Management

Bengaluru -560064

Date of Submission:

ACKNOWLEDGEMENT

I am happy to present this project after completing it successfully. This work would not have been possible without the guidance, assistance and suggestions of many individuals. I would like to express my deep sense of gratitude and indebtedness to each and every one who has helped me to make this work success.

I gracefully thank my Internal Guide, **Dr Anjan Krishnamurthy, Associate Professor & PG Coordinator, Department of Computer Science and Engineering, BMS Institute of Technology & Management** for his guidance, intangible support and advice throughout the dissertation. It would have not been possible without his support and guidance.

I sincerely thank my PEC member **Dr. Anil GN, Dean Academics, BMS Institute of Technology & Management**, for his constant support, advice and guidance throughout the dissertation.

I sincerely convey my thanks to **Head of the Department, Dr. Bhuvanewari C.M, Department of Computer Science and Engineering, BMS Institute of Technology & Management**, for her constant encouragement and support.

I express my heartfelt gratitude and sincere thanks to **Dr. Mohan Babu G.N, Principal of BMS IT & M** for his constant encouragement and inspiration.

Finally, I would like to thank to my family, friends and all those who are involved in successful completion of project work.

Gautham SK

ABSTRACT

Security as become a major concern for wireless multimedia networks because of their role in providing various services. The cost and importance provided for the security systems are less or considered as the final phase, due to this the system can face a number of consequences and huge loss in terms of cost and clients. When traditional security techniques are followed in the system it leads to inadequacies in identifying emerging security threats and also lacks in computing efficiency. Furthermore, conventional upper-layer authentication doesn't provide any protection for physical layer, thus leading to leakage of privacy data which is a major concern.

Keep these issues in mind, the report has envisioned an artificial intelligence based security authentication system that is lightweight, adaptive and doesn't require any explicit programming. Here, the security authentication system is build using neural networks that is build on convolutional filters that explore the physical layer attributes and understand the structure of the system, thereby classifying the different devices to there respective classes.

Experimental analysis and validation can ensure that the privacy of wireless multimedia devices can be achieved and also ensuring lightweight authentication, thereby solving the concerns of traditional security authentication system. The neural model is also trained using Gaussian noise of different standard deviation so that it can be used in a practical scenario.

Contents

ACKNOWLEDGEMENT	i
ABSTRACT	ii
CONTENTS	ii
1 INTRODUCTION	1
1.1 Background	1
1.2 Motivation	2
1.3 Problem Statement	3
1.4 Aim and Objective	3
1.5 Scope	4
1.6 Literature Survey	4
1.7 Summary	9
1.8 Organization of the thesis	10
2 Core Concepts of Neural Network	11
2.1 Convolutional Neural Network	11
2.2 Rectified Linear Unit (ReLU)	13
2.3 Pooling Layer	13
2.4 Fully Connected Layer	14
2.5 Summary	14
3 SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION	15
3.1 Preamble	15
3.2 Hardware Requirements	15

3.3	Software Requirements	16
3.4	Supporting Libraries	16
3.5	Other Non-Functional Requirements	16
4	SYSTEM DESIGN	18
4.1	Preamble	18
4.2	System Architecture	18
4.3	Design of Data Flow Diagram	19
4.3.1	Data Flow Diagram - Level 0	20
4.3.2	Data Flow Diagram - Level 1	20
4.3.3	Data Flow Diagram - Level 2	20
4.3.4	Use Case Diagram	23
4.3.5	UML Activity Diagram	24
4.3.6	Summary	25
5	IMPLEMENTATION	26
5.1	Programming Language and Platform Selection	26
5.2	Process involved in implementation	26
5.2.1	Splitting	27
5.2.2	Preprocessing	27
5.2.3	Data Modelling	28
5.2.4	Feature Extraction	28
5.2.5	Classifiers	30
5.3	Implementation Challenges	32
5.4	Summary	33
6	SOFTWARE TESTING	34
6.1	Software Testing	34
6.2	Unit Test of CNN model without noise	34
6.3	Unit Test of CNN model with Noise (Standard deviation: 0.1)	35
6.4	Unit Test of CNN model with Noise (Standard deviation: 0.2)	36

6.5	Unit Test of CNN model with Noise (Standard deviation: 0.3)	36
6.6	Unit Test of CNN model with Noise (Standard deviation: 0.4)	37
6.7	Unit Test of CNN model with Noise (Standard deviation: 0.5)	38
6.8	Integration Testing	38
6.9	Importing Dataset	39
6.10	Importing user defined function	39
6.11	System Testing	40
7	EXPERIMENTAL ANALYSIS AND RESULTS	41
7.1	Experimental Analysis	41
7.1.1	Confusion Matrix of CNN Model without Noise	42
7.1.2	Confusion Matrix of CNN Model with Noise 0.1	43
7.1.3	Confusion Matrix of CNN Model with Noise 0.2	44
7.1.4	Confusion Matrix of CNN Model with Noise 0.3	45
7.1.5	Confusion Matrix of CNN Model with Noise 0.4	46
7.1.6	Confusion Matrix of CNN Model with Noise 0.5	47
7.2	Analysis and Validation	48
7.3	Model Accuracy of CNN Model without Noise	48
7.4	Model Accuracy of CNN Model with Noise 0.1	49
7.5	Model Accuracy of CNN Model with Noise 0.2	49
7.6	Model Accuracy of CNN Model with Noise 0.3	50
7.7	Model Accuracy of CNN Model with Noise 0.4	51
7.8	Model Accuracy of CNN Model with Noise 0.5	51
8	CONCLUSION	53
8.1	Limitations	53
8.2	Future Scope	53
APPENDICES61		

List of Figures

2.1	Example of a filter convolving on a input to create a feature map (Source: machinelearningmastery.com)	12
4.1	System Architecture	19
4.2	DFD - Level 0	20
4.3	DFD - Level 1	21
4.4	DFD - Level 2	22
4.5	Use Case Diagram	23
4.6	UML Activity Diagram	24
5.1	Dataset Splitting	27
5.2	Preprocessing of Dataset	27
5.3	Resampling of dataset	28
5.4	Feature Extraction	29
5.5	Convolution 1D	30
5.6	BatchNormalization()	30
5.7	MaxPooling()	31
5.8	Flatten()	31
5.9	Dense()	32
7.1	CNN without Noise - Confusion Matrix	42
7.2	CNN with Noise 0.1 - Confusion Matrix	43
7.3	CNN with Noise 0.2 - Confusion Matrix	44
7.4	CNN with Noise 0.3 - Confusion Matrix	45

7.5	CNN with Noise 0.4 - Confusion Matrix	46
7.6	CNN with Noise 0.5 - Confusion Matrix	47
7.7	CNN without Noise - Model Accuracy	48
7.8	CNN with Noise 0.1 - Model Accuracy	49
7.9	CNN with Noise 0.2 - Model Accuracy	50
7.10	CNN with Noise 0.3 - Model Accuracy	50
7.11	CNN with Noise 0.4 - Model Accuracy	51
7.12	CNN with Noise 0.5 - Model Accuracy	52

List of Tables

6.1	Testing of CNN without noise	35
6.2	Testing of CNN with noise 0.1	35
6.3	Testing of CNN with noise 0.2	36
6.4	Testing of CNN with noise 0.3	37
6.5	Testing of CNN with noise 0.4	37
6.6	Testing of CNN with noise 0.5	38
6.7	Import modules	39
6.8	Import dataset	39
6.9	Import user defined function	40
6.10	System Testing	40

GLOSSARY

NIST	National Institute of Standard and Technology
CNN	Convolutional Neural Network
RSS	Received Signal Strength
DAS	Distance Between Adjacent
PCC	Pearson Coorelation Coefficient
ELM	Extreme Learning Model
PHY	Physical
AUC	Authentication
ML	Machine Learning
DNN	Deep Neural Network
CPNN	Convolution Preprocessing Neural Network
SLF	Secure Loss Function
SL	Safety Learning
Tx	Transmitting
Rx	Receiver
RF-PUF	Radio-Frequency Physical Unclonable Function
SWIPT	Simultaneous Wireless Information & Power Transfer
WPCN	Wireless Powered Communication Network
FNN	Feed Forward Neural Network
SNR	Signal-To $_{NoiseRatio}$

Chapter 1

INTRODUCTION

1.1 Background

The approach of advances in 5G and Internet of Things (IoT) envoys the appearance of the following flood of omnipresent connected society [1] [2] [3] [4] [5]. Specifically, when artificial intelligence and multimedia networks convergence it brings wide range of services and applications for training, monitoring and working in the areas of smart home, transportation, smart city, healthcare and many more [6] [7]. Multimedia applications will significantly extend the manner in which people see the world and be applied to individuals' every day lives.

Be that as it may, the complexity of multimedia frameworks as well as drastically expanding utilization of multimedia sensors inside the smart process bring numerous security and privacy challenges. For instance, when the multimedia network gather different information through various sensors, the attacker/malicious sensors could delude the user by misusing interactivity and giving false messages to the frameworks [8] [9]. Possible security attacks could prompt disastrous consequences what's more, can cause avalanche like damages in wireless multimedia networks [10]. Also, the widely used resource controlled multimedia devices are defenseless against attacks, causing damages to the multimedia network through different types of wireless attacks. Thus, it is important to plan a compelling protection mechanism for wireless multimedia networks to guarantee the security of communication transmissions. Normally, access control mechanism

and authentication are considered as key security procedures and basic design for multimedia networks [11]. These procedures secure communication by affirming the identities of all users and their access to the approved network. In any case, the disposition of large number of devices brings new challenges for security setups. Expressly, those multimedia devices that works on low-delay transmissions couldn't uphold the authentication technique that require very high computational overhead, and the sensors contained in the network require lightweight processing cost to guarantee communication performance. Hence, to protect against the attacks in wireless multimedia networks [12], this paper focuses on the difficulties looked by traditional verification schemes and further proposes new light weight security strategy method.

1.2 Motivation

The motivation of the project is the difficulties faced from conventional security approaches. Customary key-based cryptography strategies require huge resources and high processing capabilities, which is not effective for wireless multimedia networks. All the more significantly, digital key might be undermined in security management procedures, like key management or key transmission.

Conventional security procedures may experience attacks from adversary due to increases in the number of multimedia sensors which leads to the increase in the complexity of network scenarios. Security strategies executed on higher layers of networks face difficult to maintain the balance between the security and cost, which leads to inability to secure legitimate communication. Due to this The malignant devices can access the sensitive data and damage the authentication process. Therefore, artificial intelligence based lightweight authentication system will very much useful to overcome the these security issues.

Conventional authentication procedures require more efforts to extricate complex features to increase the security levels, which leads to higher communication and processing overheads, thereby leading to communication latency. This is not acceptable for a real-time wireless multimedia network. The conventional authentication procedures need time

to manually select the statistical characteristics, which is not a non-adaptive authentication process. Therefore there is need for authentication mechanism which are adaptive in nature.

Traditional authentication strategies faces difficulties while establishing an accurate detection model in a practical scenarios. This is because the model is trained using limited statistical properties to predict the outcome. These bring loopholes in the learning model which is a threat for continuous learning. To improve the security of the confirmation measure, it is important to plan a shrewd verification approach that doesn't need express programming. So, to increase the security of the authentication, we need to build design a model that does not require any explicit programming.

1.3 Problem Statement

The problem are derived after making a thorough observation and analysis about the neural network that can recognizes legitimate nodes in the wireless multimedia network and also classify four different types of wireless network attacks. So need to design system that allow to:

- Accurately and efficiently classify the multimedia nodes as legitimate or illegitimate and also classify the type of attack.
- Designing the CNN architecture which will produce the best analysis result.
- Designing models that can also classify multimedia noise which is under noise with different ranges i.e. is from standard deviation of 0.1 to 0.5, where the trained model can also be used in the practical environment.

1.4 Aim and Objective

The project objective is as follows:

- To study various neural network methods.

- To identify the appropriate neural network techniques that can solve the mentioned problem statements.
- To select the appropriate dataset to build the neural network for the problem statement.
- To build the required CNN architecture to obtain the required results.
- Perform the required analysis and validation of the neural model.

1.5 Scope

The focus of the project is to build a neural network model (CNN) that can recognize legitimate and illegitimate wireless multimedia nodes and also classify the type of wireless attack. The model also build to classify multimedia nodes that is under noise, so the build model can also be used in the practical environment.

1.6 Literature Survey

A slope authentication [12] is proposed for physical layer, which is based on the optimal feature selection method and neural network., a new index, feature validity value, is first introduced to evaluate the impact of sensitive features. Then, based on the new feature validity index, an algorithm is designed to select the optimal features from the legitimate devices. This algorithm is in a position to alleviate the over-fitting problem of the underlying neural network to an out sized extent. The selected optimal features [13] are used to train the underlying neural network, and finally, an optimal classifier is constructed to detect the illegitimate devices.

A generalized model for physical-layer-based confidential data transmission and wireless authentication [14] which is based on channel uncertainty and available design dimensions such as time, frequency, and space. The limitation of the model is it shows low accuracy in multi carrier system.

A new blind authentication [15] scheme at the physical layer based on the techniques of blind known interference cancellation. The scheme is based on Simplified Neural Network which is a modified version a feed forward neural network, it takes mean and variance close to 0 & 1 of the training data set. It uses scaled exponential linear unit and uses alpha dropout to prevent overfitting. The model is trained with 72 million records which is divided into 5 classes namely normal traffic, evasion, white box attack, reliability and theft. The neural network consists of 3 hidden layer which consist of 16 neurons for each layer. This is first paper which as normalized the input features of deep learning model in IoT dataset, which resulted in performance increase but became vulnerable to adversarial attacks.

A physical-layer authentication scheme based on extreme learning machine (ELM) [16] that exploit multi-dimensional characters of radio channels and use the training data generated from the spoofing model to improve the spoofing detection accuracy. ELM is a modified version of feed forward neural network mainly used for classification and regression. The advantage is fast learning speed, ease implementation and minimal human intervention. It consists of multi- layer neural network so there is no need of iteration and weight of nodes are solved by least square solution. They also proposed a pseudo adversary model which generates adversarial samples which is based on Euclidean distance. The limitation is it increases of Bayes risk due to insufficient training data.

A novel threshold-free PHY-AUC method based on machine learning (ML) [17], which adopts channel matrices estimated by the wireless nodes. It uses channel matrices estimated by wireless nodes as the authentication input and investigates the optimal dimension of channel matrices to improve accuracy. The channel matrices provide information about spatial multiplexing to determine is it possible for multi-layer data transmission. The limitation is that is the matrices are subdivided in small matrices which decreases the information about the wireless channel.

A deep learning based physical layer authentication framework is proposed [18]. Three algorithms are used:

- Deep neural network (DNN).

- Convolutional neural network (CNN).
- Convolution preprocessing neural network (CPNN).

It is based on spoofing attacks, the input to the model is channel state information which describes the properties of the communication link. DNN is a modified version of artificial neural network because of the increase in the number of hidden layers, which contains weights and thresholds which represent the activation function. CNN is a neural network with convolution layer and pooling layer. Convolution layer Computes multiple attributes paralleling to produce set of activation function. Pooling layer reduces the data dimension without losing the valid information. The core idea of CPNN is to perform offline convolution preprocessing before training the neural network which reduce the data dimension and extract the feature information.

A secured transmission is achieved by utilizing a modified secure loss function (SLF) based on cross-entropy which is based on machine-learning libraries [19]. SLF approach is applied in a Gaussian wiretap channel setup. It is based on cross entropy loss which measure the performance of classification model whose output values between 0 1. The model is trained based on the log loss, where if the log loss is more the predictability of the model decreases. The model consists of 3 phases where the adversary node is trained with different scenarios and based in the learning the sender and receiver node is trained. A Dual Learning-based safe Semi-supervised learning [20], which employs dual learning to estimate the safety or risk of the unlabeled instances. For safe exploitation of the unlabeled instances, used supervised learning (SL) to analyze the risk of the unlabeled instances. The limitation is that it predicts bad results for more than 200 instances it is safer to use clustering and regression for instances for more than 200.

RF-PUF [21] which is a deep neural network-based framework that allows real-time authentication of wireless nodes based on effects of inherent process variation on RF properties of the wireless transmitters (Tx), detected through in-situ machine learning at the receiver (Rx) end. The limitation is that the Rx requires two neural networks which can be implemented using the on-board microprocessor at a nominal power cost which is not significant if the network is asymmetric.

The paper [22] explores how to model an intrusion detection system based on deep learning which is based on recurrent neural networks. Recurrent neural network is an advanced version of feed forward neural network, in this node between the hidden layer are connected with each other, so it not only depends on the weights but also hidden vector. It is based directional loop, that is it memorizes the previous input information and apply it to the current output. It also takes series of input, so it produces a different output depending on the previous input.

The paper [23], briefly review the concepts of machine learning and there propose in the compelling applications of 5G networks, including cognitive radios, massive MIMO, smart grid, device-to device communications, and so on. It specifies different algorithm based on supervised, un-supervised and reinforcement learning based on 5G and which algorithm can be used based on the type of dataset or attributes. It also talks on heterogeneous network which is combination of KNN and SVM.

A deep learning memory structure which enables the IoT devices [24] to extract a set of stochastic features from their generated signal and dynamically watermark these features into the signal. It also provides of deep learning basic based on physical layer, list of layer types, list of activation functions, list loss functions, deep learning libraries, network dimension, convolution layer, examples of machine learning application for physical layer and challenges faced in physical layer

The paper [25], briefs about how deep learning is used to handle wireless OFDM channels in an end-to-end manner. This approach estimates CSI implicitly and recovers the transmitted symbols directly. Its limitation of the paper is that it as not specified a good generalization ability which led to disagreement with the channel models used in training stage.

Investigate the attack model for IoT systems and review the IoT security solutions based on machine learning (ML) techniques [26] including supervised learning, unsupervised learning, and reinforcement learning. It specifies different attack that can take place in IoT devices and steps to prevent these attacks. It also specified different each attack which security techniques and machine learning techniques that can be used. And concluding the challenges faced in IoT devices.

Survey is performed on the applications of DL algorithms [27] for different network layers, including physical layer modulation/coding, link layer access control/resource allocation, and routing layer path search, and traffic balancing. It specifies the benefits of using deep learning techniques used in physical layer authentication and different deep learning techniques used in different layer of networks especially physical layer and also specified different deep learning frameworks used in physical layer.

A joint design of the AN-aided transmission [28] and the power allocation to maximize the secrecy rate at the destination, under the harvested energy constraint at the ERs. In the paper RF signals carries both energy and information for the receiver nodes. Two system has been proposed Simultaneous Wireless Information Power Transfer (SWIPT) and Wireless Powered Communication Network (WPCN). They have used the concept of Artificial Noise (AN) that is created by the neural model trained using the channel information state.

Considered a secure communication in a multi user wireless network [29], where full-duplex (FD) users operate to enhance wireless physical layer security and used this to exploit the user selection scheme to strengthen the secure performance. The limitation of the paper is that while modelling they have considered imperfect channel state information by considering channel estimation errors.

Considered interpreting a communications system as an auto encoder [30] and developed a fundamental new way to think about communications system design as an end-to-end reconstruction task that seeks to jointly optimize transmitter and receiver components during a single process. The challenge is the scalability to long block lengths.

Considered a variant of the Feed Forward Neural Network FNN known as the Self-normalizing Neural Network (SNN) [31] and compare its performance with the FNN for classifying intrusion attacks in an IoT network. The limitation of the paper is that performance accuracy for adversarial attacks was still below 50% and could not be regarded as a suitable defense against adversarial attacks.

Considered multi-user system which is equipped with FD legitimate receivers [32] with advantage of FD capability of the receivers to send jamming signals against the eavesdropper. The limitation of the paper is, when the transmit power of source is higher

than 10 dB, the secrecy rate is reduced. Performance of All-in jamming method isn't the simplest solution for the case of low SNR.

Proposed an adaptive physical layer authentication scheme based on machine learning as an intelligent process to learn and utilize the complex time-varying environment [33] and to improve the reliability and robustness of physical layer authentication. The limitation of the paper is that while training the paper they have only considered the 3 attributes of the physical layer.

Proposed a novel authentication method based on sparse representation [34] for the reconciliation in physical layer key Generation. The limitation of the paper is the they have only considered low signal-to noise ration which is not application in the case of practical environment.

Proposed a physical-layer spoofing detecting scheme [35], were signal processing and feature recognition are utilized to improve the detection performance. The model is based on kernel-based Machine Learning Technique which is a non-parametric learning technique, they estimate the physical layer attributes using linear function or polynomial function. They have also considered the receiving time along with the other attributes. The limitation is that feature selection is still depending on experience and appropriate features are crucial impact on the recognition performance.

1.7 Summary

To summaries this particular section, the section starts with the background of AI technologies combined with multimedia networks which brings wide range of technologies and the security issues or consequences that follows. it also talks about the traditional or convectional security authentication system and its fall backs, thereby leading to its motivation to build a better authentication system. Followed by the problem statements of the proposed model building, the aim, objectives and scope of the security system. Finally the literature survey that build the road map and provided with knowledge and techniques used to build the proposed security authentication system.

1.8 Organization of the thesis

The rest of this report is structured as follows. Chapter II describe about the fundamental of the neural network and different methods that have been used in the neural model.

Chapter III gives a description about the software and hardware requirement utilized in the model and the supporting libraries used to build the neural network.

Chapter IV describes the system architecture, use case diagram, UML activity diagram and data flow diagram.

Chapter V talks about the implementation and process involved in building the neural network.

Chapter VI described the software testing and validation process involved in the neural network that includes the validation of different model based on standard deviation of noise, integration testing, importing of dataset, user defined dataset and the system testing.

Chapter VII describes about the experimental analysis and the results of the neural network, how much effective is the neural network and also based on the noise deviation.

Chapter VIII is the concluding part of the report or thesis or the project which talks about the limitation and conclusion of the project. future research prospects.

Chapter 2

Core Concepts of Neural Network

The root concept of neural network is artificial intelligence. A neural network is made up of a collection of algorithms that can recognize the relationship within a set of data through a process that is similar to the way in which human brain works. In general neural network refers to a system of neurons that can adapt to the changing input, so that the network generates the same results without the need of redesigning the output criteria. One of the algorithm that is based on neural network is Convolutional Neural Network (CNN).

2.1 Convolutional Neural Network

The **convolutional neural network** is majorly build by convolutional layer which perform an operation know as "convolution. A convolution is a simple process where a filter convolve through the input which results in an activation. Repeating the same process with the same filter to an input results in a map of activation's called a feature map, which indicates the relationship and strength of the detected feature present in the input.

Generally, convolution is a linear operation that involves multiplication of set of weights with the input, which is similar to the traditional neural network. The multiplication is performed between the array of input data and array of weights, commonly know as filters.

The size of the filters are always smaller than the input data and the multiplication operation that is performed between the filter sized input and the filter is dot product which performs element wise multiplication and the result that is obtained is summed, which is a single value. The operation that is performed is commonly referred as scalar product because the result obtained is a single value.

The size of the filter is intentionally set to a size smaller than the input, so that the same filter i.e. weights, can be multiplied multiple times with the input array at different points on the input which includes left to right, top to bottom. This systematic approach of same filter convolving across the input bring out a powerful idea which is commonly know as translation in-variance, where the filter that is designed to detect specific type feature from the input, then the process of the filter that systematically moves across the entire input allows the filter to discover the specific feature present anywhere in the input. The output of the multiplication operation is a single value, because the operation is preformed multiple times, it results to a two-dimensional array which is called as feature map.

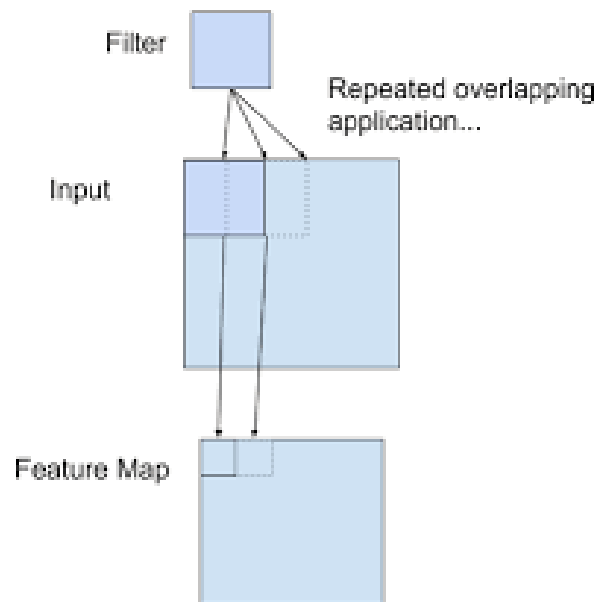


Figure 2.1: Example of a filter convolving on a input to create a feature map (Source: machinelearningmastery.com)

2.2 Rectified Linear Unit (ReLU)

ReLU is an implementation that combines non-linearity and the rectification layer of a CNN. It is linear function defined as:

$$Y_i^{(l)} = \max(0, Y_i^{(l-1)}) \quad (2.1)$$

ReLU effectively propagate the gradient thereby reducing the vanishing gradient problem that commonly seen in the deep neural architecture. It solves the cancellation problem by thresholds the negative values to zero which results in more sparse activation function in the output. ReLU consists of simple operations like comparison which is efficient to implement in CNN.

2.3 Pooling Layer

One of the limitation of feature map which is generated by the convolutional layer is that they store the precise location of each feature present in the input, so if there is a small movement in the position of feature in the input it will lead to a different feature map. So an approach to solve this problem is down sampling also know as pooling layer. This layer ignores the weakest features that is present in the input, retaining only the strongest features that is present in the input thereby reducing the size of the feature map. This layer comes after the ReLU layer. It works similar to the convolutional layer where a pooling filter that performs a specific operation convolves through the feature map. The size of the filter is smaller than the input feature map. Two common pooling operation are:

- **Max Pooling:** It calculates the maximum value present in each patch of feature map.
- **Average Pooling:** It calculates the average value present in each patch of feature map.

2.4 Fully Connected Layer

Fully Connected Layer is simple feed forward neural network. These network does not contain any loops, the information moves forward from the input node to the hidden nodes and ending at the output node. The output generated by the pooling layer is flattened and then given as input to the fully connected layer. Flattening is a process where the output from the pooling layer is converted into a vector. Fully connected layer works similar to the artificial neural network. The calculation performed in each layer of fully connected network is:

$$[g(Wx + b)] \tag{2.2}$$

where,

x: Input vector with dimension [**pl**, 1]

W: Weight matrix with dimension [**pl**, **nl**]

b: Bias vector with dimension [**pl**, 1]

g: Activation Function

pl: Number of neurons in the previous layer

nl: Number of neurons in the current layer

The last layer of fully connected layer is **softmax activation function** which provides the probability of an input which belongs to a particular class.

2.5 Summary

This section dealt with the core concepts of neural network. Here, different layers or concepts in which a neural network can be build and explains each operation performed by the layers. It also explore how these layers react in a neural network.

Chapter 3

SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION

3.1 Preamble

To build a neural network, at first it needs to ensure that we have the necessary software and hardware requirements. I need to ensure that the hardware and software that is chosen is able to handle and process the large amount of data so that it gives the required output. The core of the neural network training is the dataset. The software and the hardware is going to perform its tasks in the dataset. The dataset obtained contains the physical layer attributes of the multimedia devices, which is used to understand how the physical layer attributes are related to each device.

3.2 Hardware Requirements

- Processor CPU: Intel Core i5 or above (for better performance)
- Graphics Card: CUDA enabled graphics card
- RAM: Minimum 4GB or above (for better performance)

3.3 Software Requirements

- Programming Language: Python 3.9
- Operating System: Windows 8.1 or above
- IDE: Anaconda(Jupyter Notebook), Visual Studio Code

3.4 Supporting Libraries

1. `to_categorical(Keras)`

It is used to transform the training data before passing it to the model. It convert the the input data into vectors which can be processed by the neural network model.

2. `to_weight(Keras)`

This function is used to penalize the under or over represented classes present in the training data set. It belongs to `fit()` function that maps classes to a weight value.

3. `Convolution1D(TensorFlow)`

It creates a convolution kernel that convolve with the input over a single dimension to produce output and then activation is added to the output.

4. `MaxPool1D`

The layer created is used to down sample the input which helps in over-fitting and reduce the computational cost by keeping only the strong features present.

5. `Flatten`

This layer is defined before fully connected network to provide 1D dimensional array(vector).

3.5 Other Non-Functional Requirements

Some of the non-functional requirements are:

- Modularity: The model is build in number of methods, so only limited changes needs to be done, if any modification needs to be done and it is also easy to handle any bugs or error present in the implementation.
- Scalability: The authentication system is highly scalable because the neural network almost analysis around 2 crore of attribute values for each epochs.

Chapter 4

SYSTEM DESIGN

4.1 Preamble

Every implementation of a system requires its own design phase. The design phase includes building the system architecture. different levels of data flow diagram to represent the data flow of the system, the use case diagram which represent the different entities of the system and the task performed by each entities and finally the UML activity diagram which represent the behavior of the system.

4.2 System Architecture

The system architecture described in fig 4.1; describes about a neural network model that can classify multimedia devices present in wireless network into five classes. The five classes are:

1. Legitimate Devices
2. Devices under Sybil Attack
3. Devices under Black-hole Attack
4. Devices under Jamming Attack
5. Devices under Exhaustion

The neural network model that is build is Convolutional Neural Network, the model is trained using the training dataset, which consist of physical layer attributes of wireless multimedia device. Once the model is trained, the trained model is validated using the testing dataset, which gives out the analysis and validation of the model.

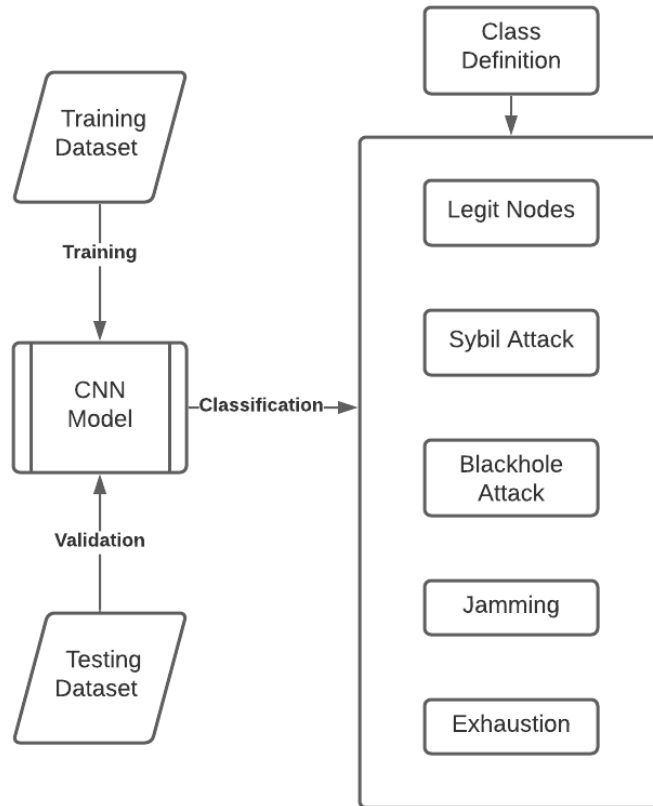


Figure 4.1: System Architecture

4.3 Design of Data Flow Diagram

Data Flow Diagram are used to represent the data flow of the system. It describes the steps or process involved in a system i.e. from input till report generation. It also show the transaction from one state to another state of the system. The neural network that is build in this thesis will represented in three different levels that are numbered 0, 1 and 2.

4.3.1 Data Flow Diagram - Level 0

DFD level 0 is also known as Context Diagram. It describes the complete overview of the system that has been modelled.

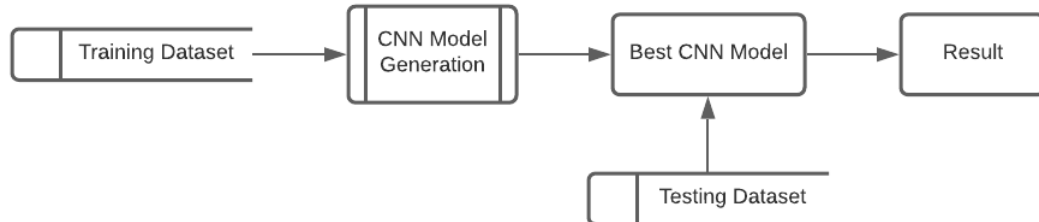


Figure 4.2: DFD - Level 0

Fig 4.2 describes the level 0 DFD of the system, which gives a general idea of how the system works. It represents the system as a high level process and its relationship with other entities. It starts with CNN model generation using the training dataset, where multiple models will be created within which a best model will be selected which will be validated and analyzed using the testing dataset and the results will be generated.

4.3.2 Data Flow Diagram - Level 1

DFD level 1 provides more detail about the context diagram. Here the high level process of the context diagram is represented into its sub-processes. The DFD level 1 of the system is shown in fig 4.3

The level 1 DFD describes the process that takes place in the dataset. Here the training data undergoes two types of transaction called as re-sampling, which is used to create a balanced dataset and are appended with Gaussian noise, which is then followed by training of model and validation using the testing dataset.

4.3.3 Data Flow Diagram - Level 2

DFD level 2 of the system is described in fig 4.4 which goes deeper into the sub-processes of level 1 DFD. It gives a clear picture of how the system works and different process in

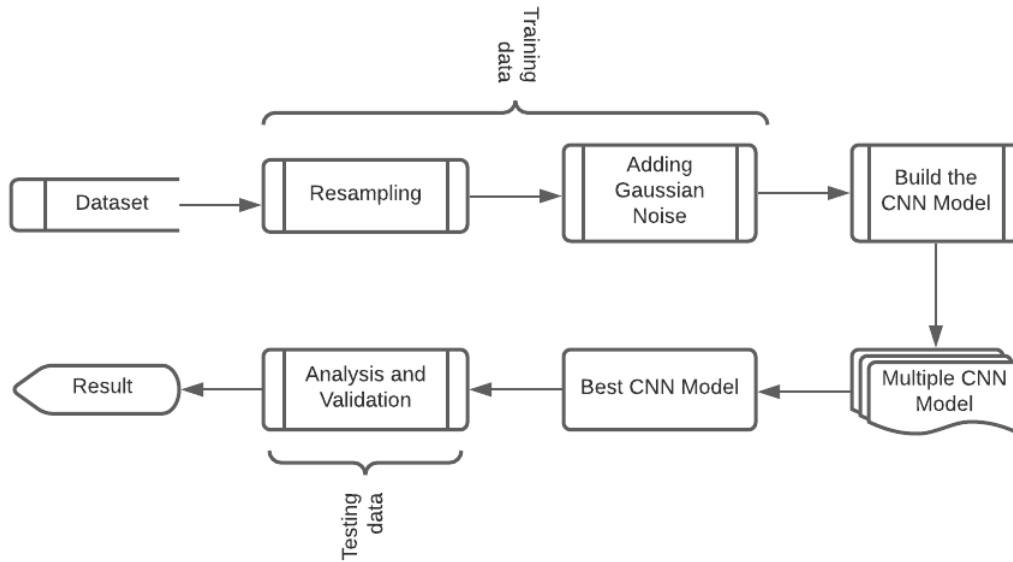


Figure 4.3: DFD - Level 1

each step.

The level 2 diagram describes the system into three categories which are preprocessing, data scaling and model building. Each categories is explained graphically in detail and gives a complete idea of how the neural network is build.

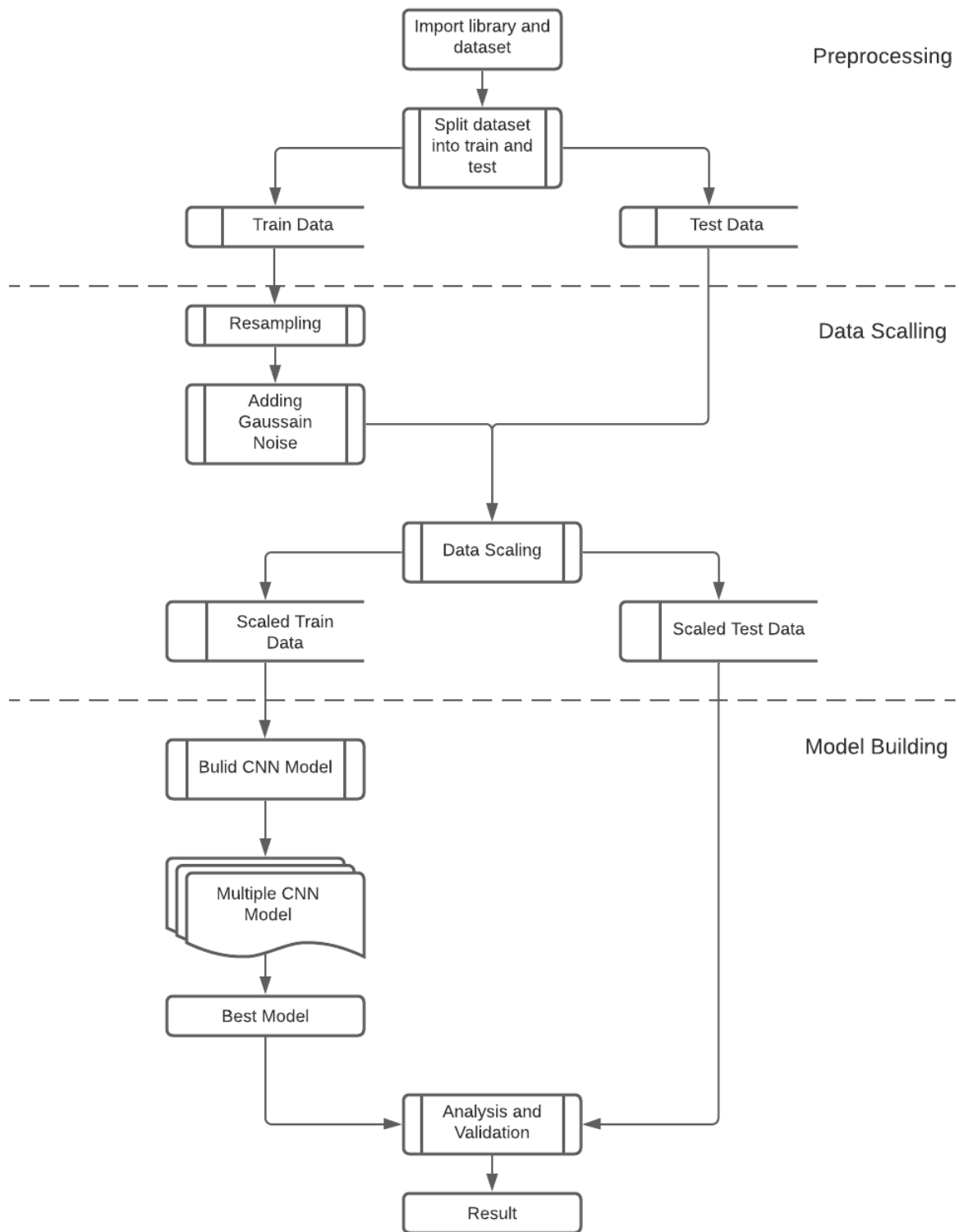


Figure 4.4: DFD - Level 2

4.3.4 Use Case Diagram

The use case diagram of the system is represented in fig 4.6. The diagram describes different entities present in the system and what all tasks are performed by each task. The use case diagram consists of two entity namely system and user. The task performed by the system is importing, preprocessing, data scaling, CNN model generation, analysis and validation. The task performed by the user is to provide the system with the required training and testing dataset.

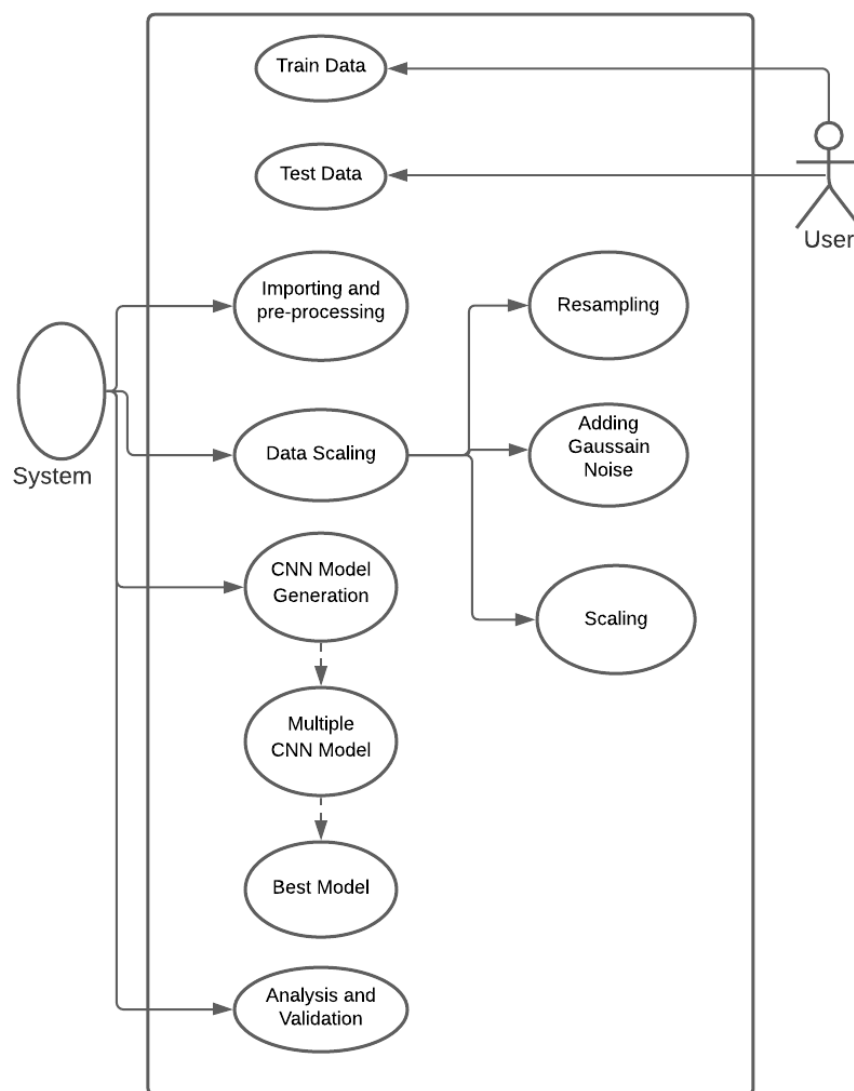


Figure 4.5: Use Case Diagram

4.3.5 UML Activity Diagram

Activity diagram is also called behavioral diagram. The behavioral diagram of the system is shown in fig 4.5. The diagram describes how the system responds on different paths during the execution of the model i.e. from starting point till the end point.

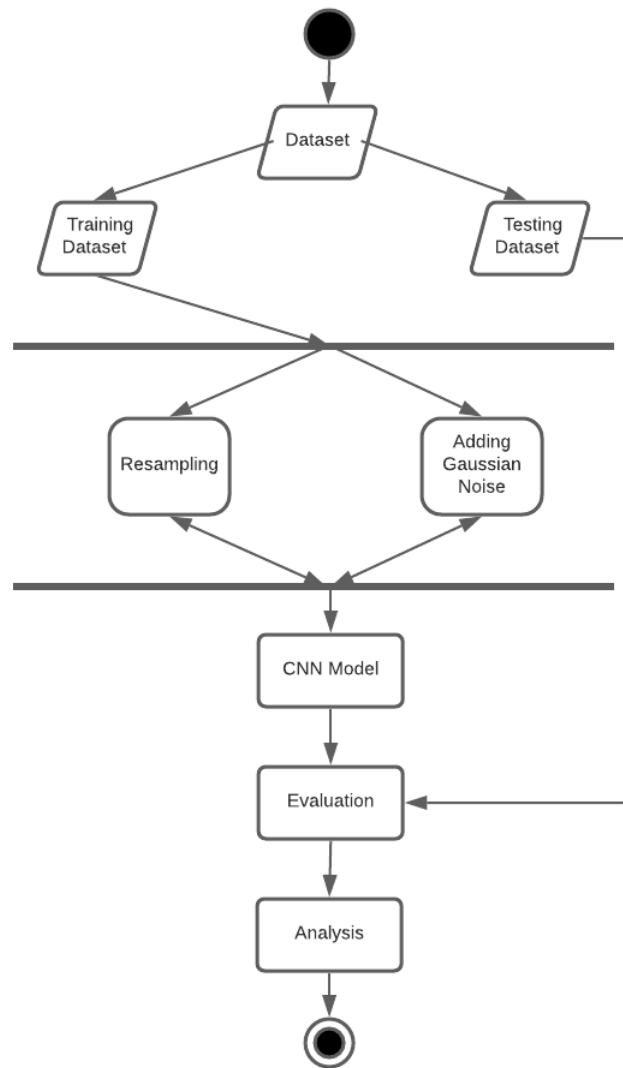


Figure 4.6: UML Activity Diagram

4.3.6 Summary

The above chapter deals with the system design where it explains about the system architecture that represent the authentication system and how different classes are classified, different processes or steps that is performed by the system which represent the behaviour of the system that represented by the UML activity diagram, different entities and operation performed by each entities that are involved in the system is represented using the use case diagram and the data flow of the system which is represented by different level of flow diagram.

Chapter 5

IMPLEMENTATION

This chapter illustrates the implementation part of the neural network i.e. convolutional neural network that can recognize whether the multimedia device within the wireless medium are legitimate devices or not. Building the system can be divided into three phases known as preprocessing, data scaling and model building. Each phase has a set of processes.

5.1 Programming Language and Platform Selection

The programming language that was used for the CNN model was Python 3.9 along with the libraries such as sklearn, tensorflow and keras. The IDE that was used to build the model was Anaconda Navigator which provided the platform Jupyter Notebook which is an open format document which was based on JSON. Visual Studio Code was used to integrate the application with the front end and the front end was built using HTML and CSS.

5.2 Process involved in implementation

The first step to build the model is to determine the right set of dataset to train the model. The unstructured time series dataset was obtained from the NIST. The website of NIST provided scenario and information of smart industries and smart systems. The dataset consisted of information about 47 sensors or devices which includes the physical

layer attributes of the devices namely RSS, DAS, PCC and CIR. The size of the dataset is very large, so working with it is intriguing. The steps performed in the implementation are:

5.2.1 Splitting

The dataset which was obtained was in two part, namely training dataset and testing dataset, figure 5.1. The shape of the training dataset consisted of 87,554 rows and 188 columns which had a total size of 1,64,60,152 and the shape of the testing dataset consisted of 21,892 rows and 188 columns. The dimension of both the dataset of two-dimensional.

```
1 train_df=pd.read_csv('Arr_train.csv',header=None)
2 test_df=pd.read_csv('Arr_test.csv',header=None)
```

Figure 5.1: Dataset Splitting

5.2.2 Preprocessing

This step involves processing the dataset that involves filling the missing data or removing them and getting the dataset clean for building. But the dataset obtained was already preprocessed and did require any further preprocessing represented in figure 5.2.

```
1 train_df[187]=train_df[187].astype(int)
2 equal=train_df[187].value_counts()
3 print(equal)

0    72471
4     6431
2     5788
1     2223
3      641
Name: 187, dtype: int64
```

Figure 5.2: Preprocessing of Dataset

5.2.3 Data Modelling

The dataset obtained was needed to be modified to obtain the required the results. First the dataset was needed to resampled. This was done using the pandas dataframe called `resample()` which is mainly used in time series dataset which is represented in figure 5.3. It is used to generate unique distribution of samples based on the actual dataset. It is a very important step in the analysis because it increases the accuracy and balance the uncertainty of the dataset. It is also a important step because while inducing noise in the dataset, the noise needs to be distributed equally within every class.

Once the dataset is resampled, noise can be appended equally within every class. The noise induced is Gaussian noise. It is generated using NumPy random normal function, which takes three argument, namely mean, standard deviation and shape of the input array. Based on the value given to the standard deviation, noise can be induced in the dataset.

```
1 from sklearn.utils import resample
2 df_1=train_df[train_df[187]==1]
3 df_2=train_df[train_df[187]==2]
4 df_3=train_df[train_df[187]==3]
5 df_4=train_df[train_df[187]==4]
6 df_0=(train_df[train_df[187]==0]).sample(n=20000,random_state=42)
7
8 df_1_upsample=resample(df_1,replace=True,n_samples=20000,random_state=123)
9 df_2_upsample=resample(df_2,replace=True,n_samples=20000,random_state=124)
10 df_3_upsample=resample(df_3,replace=True,n_samples=20000,random_state=125)
11 df_4_upsample=resample(df_4,replace=True,n_samples=20000,random_state=126)
12
13 train_df=pd.concat([df_0,df_1_upsample,df_2_upsample,df_3_upsample,df_4_upsa
```

Figure 5.3: Resampling of dataset

5.2.4 Feature Extraction

In order to extract the features present CNN model is build. The model is build on different layers. The main layer are the Convolution layer, Max Pooling layer, Flatten layer and Fully Connected Layer which is represented in figure 5.4.

The Convolution layer is mainly made of filter/kernel which is matrix that convolve

over the training dataset. It is a linear function that performs dot product between the input array and the weight array which results in activation function. The filter convolve around the complete data multiple time which results in feature map. Any number of convolution layer can be added. Three layer is added to the system with varying filter size. The feature map obtained by the layer is the input to the next layer called the Max Pooling layer.

The Max Pooling layer also know as down sampling layer is a generalization process used to reduce overfitting. This layer reduce the size of the size of the feature map obtained, in other words it can also be called as optimization because the max pooling filter convolve through the feature map and selects only the highly weighted features present in the feature map, removing the least once thereby reducing the size of the feature map. The down sampled feature map is given to next layer called flatten.

Flatten layer is a layer between the convolution filters and the fully connected layer. It is function that creates a copy of the input array into one dimensional. This one dimensional array of the feature map is given as input to the fully connected layer.

The fully connected layer is simple feed forward neural network that propagates the input forward the hidden layer where dot product between the weights and input array takes place and the results are forwarded to the output layer. This networks does not contain any loops.

```
conv1_1=Convolution1D(64, (6), activation='relu', input_shape=im_shape)(inputs_cnn)
conv1_1=BatchNormalization()(conv1_1)
pool1=MaxPool1D(pool_size=(3), strides=(2), padding="same")(conv1_1)
conv2_1=Convolution1D(128, (3), activation='relu', input_shape=im_shape)(pool1)
conv2_1=BatchNormalization()(conv2_1)
pool2=MaxPool1D(pool_size=(2), strides=(2), padding="same")(conv2_1)
conv3_1=Convolution1D(128, (3), activation='relu', input_shape=im_shape)(pool2)
conv3_1=BatchNormalization()(conv3_1)
pool3=MaxPool1D(pool_size=(2), strides=(2), padding="same")(conv3_1)
```

Figure 5.4: Feature Extraction

5.2.5 Classifiers

- **keras.layers.Convolution1D** `keras.layers.Convolution1D` This layer creates a convolutional filter/kernel that convolves with the input layer which is a single dimension to produce a output. Parameters used are:
 - `filters`: Integer, specifying the number of output filters.
 - `kernel_size`: Integer, specifying the size of the 1D kernel/filter.
 - `padding`: "same" which results in padding with zero's to the left/right or up/down of the input array so that the output array has the same dimension as that of the input.

```
conv1_1=Convolution1D(64, (6), activation='relu', input_shape=im_shape)(inputs_cnn)
```

Figure 5.5: Convolution 1D

- **keras.layers.BatchNormalization()** This layer transforms the mean output close to zero and standard deviation output close to one. Parameters used are:
 - `axis`: Integer, column which needs to be normalized.
 - `momentum`: assign momentum to the average.
 - `beta_initializer`: initial the beta weight.
 - `gamma_initializer`: initial the gamma weight.

```
conv1_1=BatchNormalization()(conv1_1)
```

Figure 5.6: BatchNormalization()

- **keras.layers.MaxPooling()** Downsamples the input within the dimension by taking the maximum value within the input channel. The filter is shifted by the parameter called `strides`. Parameters used are:
 - `pool_size`: Integer, that specify the filter size which takes the maximum size from the input array.

- `strides`: Integer, it specifies the number of moves the filter as to move for each pooling step.
- `padding`: "same" which results in padding with zero's to the left/right or up/down of the input array so that the output array has the same dimension as that of the input.

```
pool1=MaxPool1D(pool_size=(3), strides=(2), padding="same")(conv1_1)
```

Figure 5.7: MaxPooling()

- **keras.layers.Flatten()** Flatten the feature map i.e. the input from the convolutional layer. Parameters used are:
 - `data_format`: String, which specify the order of input dimension.

```
flatten=Flatten()(pool3)
```

Figure 5.8: Flatten()

- **keras.layers.Dense()** It is used to implement the operation: $output = activation(dot(input, kernel) + bias)$, where *activation* is the activation function that is performed, *bias* is the bias vector that is created and *kernel* is the weight matrix created. Parameters used are:
 - `units`: Integer, dimension of the output.
 - `activation`: specify the activation function that needs to be used.
 - `use_bias`: Boolean, determines whether to use bias vector.

```

dense_end1 = Dense(128, activation='relu')(flatten)
dropout_1 = Dropout(0.5)(dense_end1)
dense_end2 = Dense(64, activation='relu')(dropout_1)
dropout_2 = Dropout(0.5)(dense_end2)
dense_end3 = Dense(32, activation='relu')(dropout_2)
dropout_3 = Dropout(0.5)(dense_end3)
dense_end4 = Dense(16, activation='relu')(dropout_3)
dropout_4 = Dropout(0.5)(dense_end4)
dense_end5 = Dense(8, activation='relu')(dropout_4)

```

Figure 5.9: Dense()

5.3 Implementation Challenges

The challenges faced while implementation are:

- The dataset is extremely very large that consisted around 2 crore attributes. So to handle this large amount data came with its own challenges. The first challenge was how to balance the dataset. Since the dataset contained data of different classes, each classes required a balanced set of data so when operations are performed on them they are equally affected on all the values so the that the neural network get a whole idea of the multimedia network it working on.

The second problem was the batch size that need to to specified during the model building, the neural network is trained in batch's, where each batch has a required size and this size matter a lot. So specifying the size of the batch was very important because it affect the reckoning rate and the accuracy.

- Since the model was trained with Gaussian noise with different standard deviation as well, it needed to be ensured that the noise is distributed equally through the dataset. So in order to do that different approach was required to be tested to distribute the noise within every class was important. The best way was to use the resample method, which creates copies of the attributes of a particular class and makes each classes balanced.
- Building the model architecture was difficult, because neural network is build is using different layer and each layer as to perform its own operation. So adding and

removing layers in the architecture will bring changes in the accuracy. So different trail and error methods needs to performed to build the model to get the required accuracy and based on that a particular implementation is selected.

5.4 Summary

This chapters summaries about the implementation of the neural network(CNN). It generally talks about the layers and steps involved in the implementation. Its explains the operations performed on each step and based on the operation how the implementation can be effected. This chapter also explains about the different steps that is involved in the implementation and what are operations performed on each step.

Chapter 6

SOFTWARE TESTING

In this chapter, working of the proposed system will be tested and compare the different result with the varying noise induced. It is also a process of validating the system.

6.1 Software Testing

A total of six models will be build which includes one model which is trained without noise and model which will be trained with Gaussian noise of different standard deviation. Once the model is trained, the model will be validated using the validation dataset and result will be analyzed.

6.2 Unit Test of CNN model without noise

The table 6.1 describes about the testing of CNN authentication system which is trained without any Gaussian noise, which is then tested and validated and its corresponding output i.e., the result of analysis which gives the model accuracy.

Table 6.1: Testing of CNN without noise

Test Case	01
Test Name	”Testing CNN without noise”
Input	CNN Model generated using the training dataset without noise and validating using the test dataset.
Output	Model Accuracy: 97.36
Remark	Success%

6.3 Unit Test of CNN model with Noise (Standard deviation: 0.1)

The table 6.2 describes about the testing of CNN authentication system which is trained with Gaussian noise of standard deviation 0.1, which is then tested and validated and its corresponding output i.e., the result of analysis which gives the model accuracy is represented.

Table 6.2: Testing of CNN with noise 0.1

Test Case	02
Test Name	”Testing CNN with noise of Standard deviation: 0.1”
Input	CNN Model generated using the training dataset with noise of Standard deviation 0.1 and validating using the test dataset.
Output	Model Accuracy: 94.21
Remark	Success%

6.4 Unit Test of CNN model with Noise (Standard deviation: 0.2)

The table 6.3 describes about the testing of CNN authentication system which is trained with Gaussian noise of standard deviation 0.2, which is then tested and validated and its corresponding output i.e., the result of analysis which gives the model accuracy is represented.

Table 6.3: Testing of CNN with noise 0.2

Test Case	03
Test Name	"Testing CNN with noise of Standard deviation: 0.2"
Input	CNN Model generated using the training dataset with noise of Standard deviation 0.2 and validating using the test dataset.
Output	Model Accuracy: 92.83
Remark	Success%

6.5 Unit Test of CNN model with Noise (Standard deviation: 0.3)

The table 6.4 describes about the testing of CNN authentication system which is trained with Gaussian noise of standard deviation 0.3, which is then tested and validated and its corresponding output i.e., the result of analysis which gives the model accuracy is represented.

Table 6.4: Testing of CNN with noise 0.3

Test Case	04
Test Name	”Testing CNN with noise of Standard deviation: 0.3”
Input	CNN Model generated using the training dataset with noise of Standard deviation 0.3 and validating using the test dataset.
Output	Model Accuracy: 89.74
Remark	Success%

6.6 Unit Test of CNN model with Noise (Standard deviation: 0.4)

The table 6.5 describes about the testing of CNN authentication system which is trained with Gaussian noise of standard deviation 0.4, which is then tested and validated and its corresponding output i.e., the result of analysis which gives the model accuracy is represented.

Table 6.5: Testing of CNN with noise 0.4

Test Case	05
Test Name	”Testing CNN with noise of Standard deviation: 0.4”
Input	CNN Model generated using the training dataset with noise of Standard deviation 0.4 and validating using the test dataset.
Output	Model Accuracy: 83.69
Remark	Success%

6.7 Unit Test of CNN model with Noise (Standard deviation: 0.5)

The table 6.6 describes about the testing of CNN authentication system which is trained with Gaussian noise of standard deviation 0.5, which is then tested and validated and its corresponding output i.e., the result of analysis which gives the model accuracy is represented.

Table 6.6: Testing of CNN with noise 0.5

Test Case	06
Test Name	”Testing CNN with noise of Standard deviation: 0.5”
Input	CNN Model generated using the training dataset with noise of Standard deviation 0.5 and validating using the test dataset.
Output	Model Accuracy: 82.72
Remark	Success%

6.8 Integration Testing

Integration testing is approach where the different units or modules are integrated and tested whether they perform the required functionality or not. It includes different modules to be tested.

The first test to be performed is the import module testing, where all the necessary modules are import from the predefined libraries based on the requirement and run it. 6.7 represent the operation of importing the modules.

Table 6.7: Import modules

Test Case	07
Test Name	"Importing Modules"
Input	Import "modules" statements
Expected Output	The module have imported%
Actual Output	The modules has imported and ready to use
Remark	Success

6.9 Importing Dataset

The first step to train the neural network is to import the required dataset. Importing dataset is done using the pandas python libraries.

Table 6.8: Import dataset

Test Case	08
Test Name	"Importing Dataset"
Input	Import "dataset" statements
Expected Output	The dataset have imported%
Actual Output	The dataset has imported and ready to use
Remark	Success

6.10 Importing user defined function

As mentioned in the non-functional requirement, the building of neural network as maintained modularity. So, the model is build using user defined method each performing

its own operations which takes the required arguments or parameters and perform the required task.

Table 6.9: Import user defined function

Test Case	09
Test Name	"Importing user defined function"
Input	Import "user defined function" statements
Expected Output	The user defined function have imported%
Actual Output	The user defined function has imported and ready to use
Remark	Success

6.11 System Testing

System testing is a approach that checks the complete model or the authentication system. The checks include whether system is performing to the required accuracy and is able to recognise the objective of the system.

Table 6.10: System Testing

Test Case	10
Test Name	"System Testing"
Input	Training dataset used to build the CNN Model
Expected Output	Model analysis resulting in less model lose.%
Actual Output	Model analysis with good accuracy and low model loss
Remark	Success

Chapter 7

EXPERIMENTAL ANALYSIS AND RESULTS

In this chapter, the neural network execution and results will be discussed.

7.1 Experimental Analysis

Confusion Matrix (CM) describes the graphical summary of the correct and incorrect predictions based on the different classes of predication. It also determine the performance of each class. CM of each CNN model will described below with includes CNN model trained using noise and without noise.

7.1.1 Confusion Matrix of CNN Model without Noise

Figure 7.1 represent the confusion matrix of the convolutional neural network which is trained without the Gaussian noise. CM represent the performance matrix for each class in terms of predicted value with respect to the actual value.

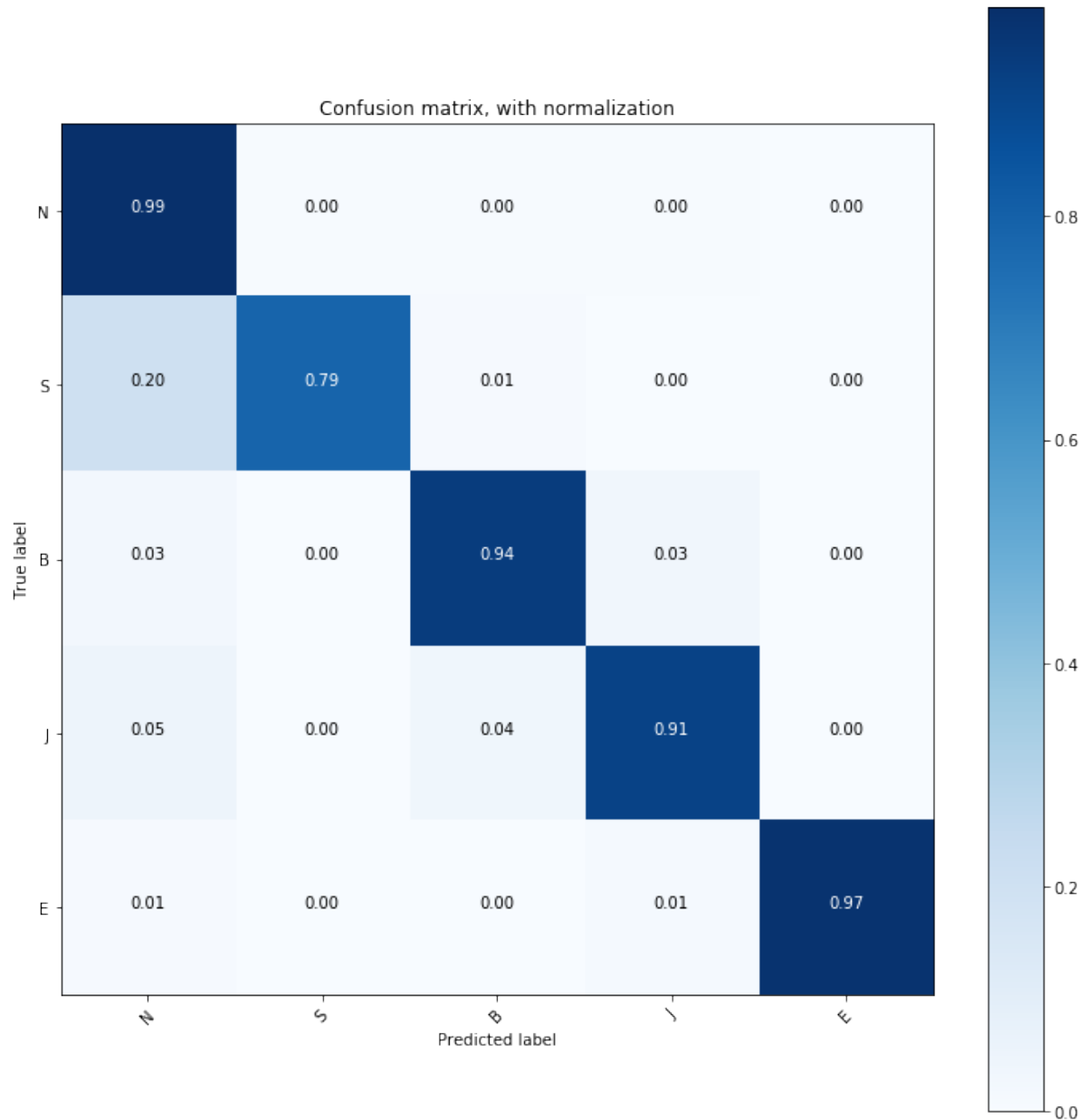


Figure 7.1: CNN without Noise - Confusion Matrix

7.1.2 Confusion Matrix of CNN Model with Noise 0.1

Figure 7.2 represent the confusion matrix of the convolutional neural network which is trained with the Gaussian noise of standard deviation 0.1. CM represent the performance matrix for each class in terms of predicted value with respect to the actual value.

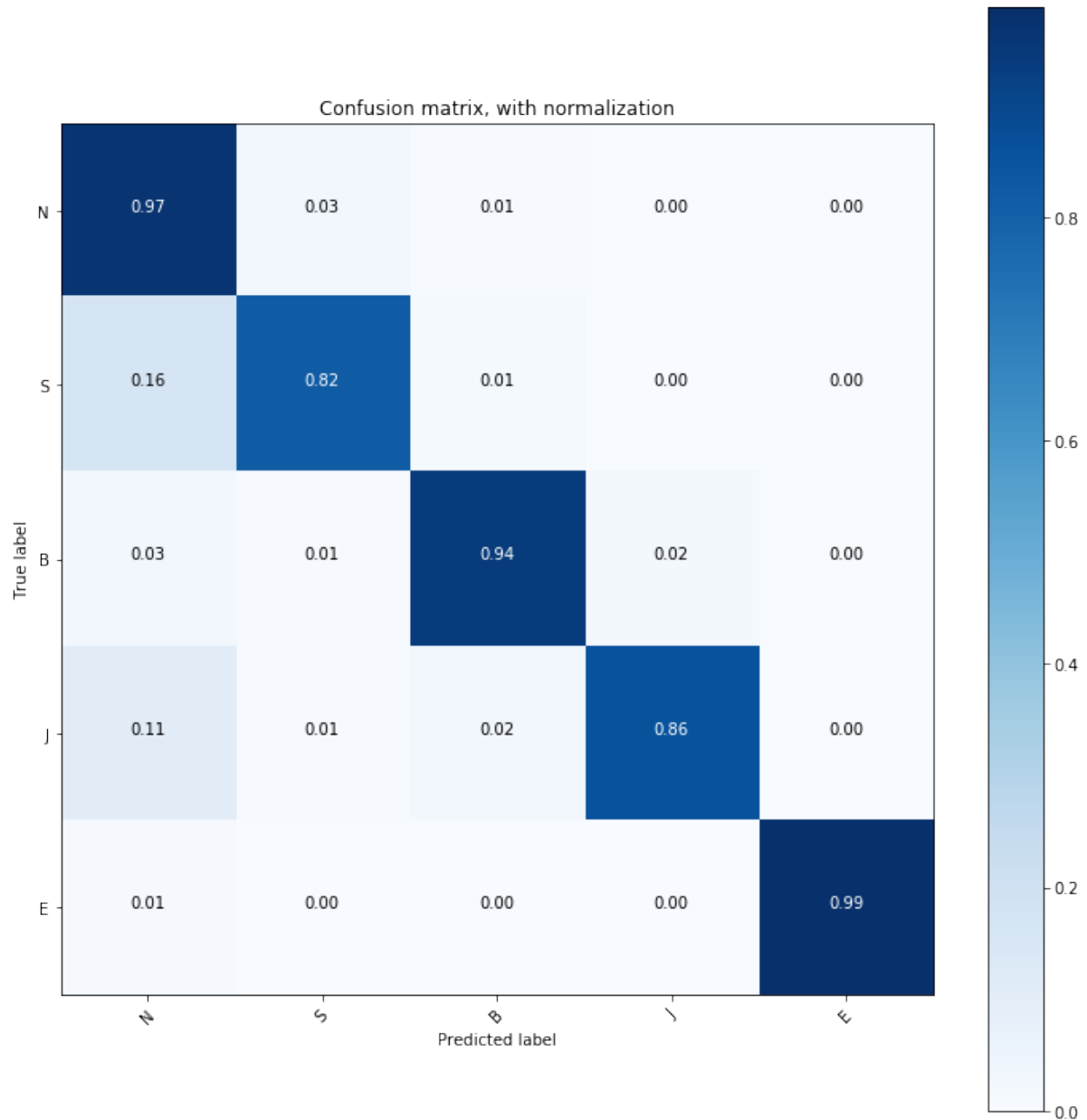


Figure 7.2: CNN with Noise 0.1 - Confusion Matrix

7.1.3 Confusion Matrix of CNN Model with Noise 0.2

Figure 7.3 represent the confusion matrix of the convolutional neural network which is trained with the Gaussian noise of standard deviation 0.2. CM represent the performance matrix for each class in terms of predicted value with respect to the actual value.

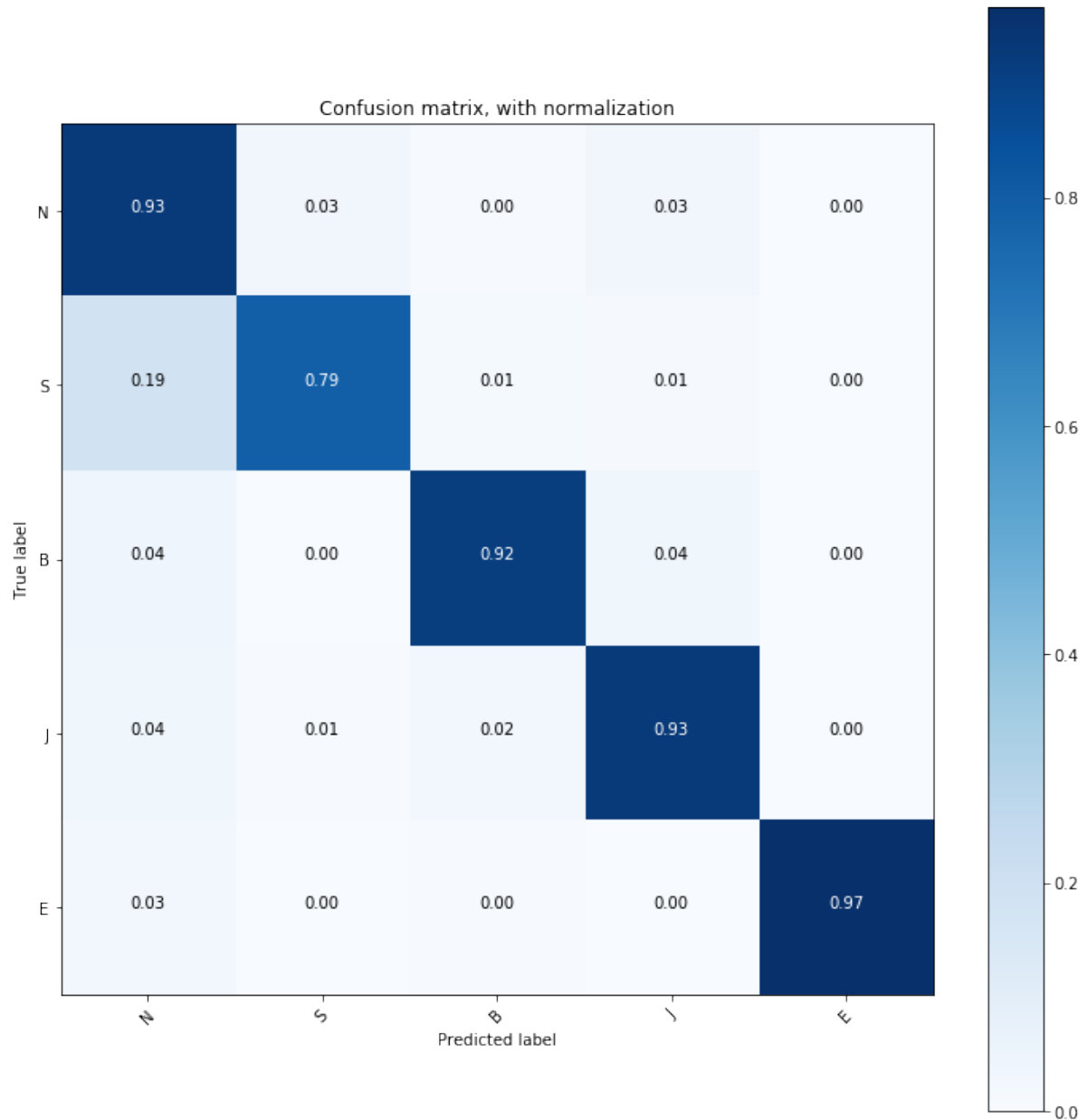


Figure 7.3: CNN with Noise 0.2 - Confusion Matrix

7.1.4 Confusion Matrix of CNN Model with Noise 0.3

Figure 7.4 represent the confusion matrix of the convolutional neural network which is trained with the Gaussian noise of standard deviation 0.3. CM represent the performance matrix for each class in terms of predicted value with respect to the actual value.

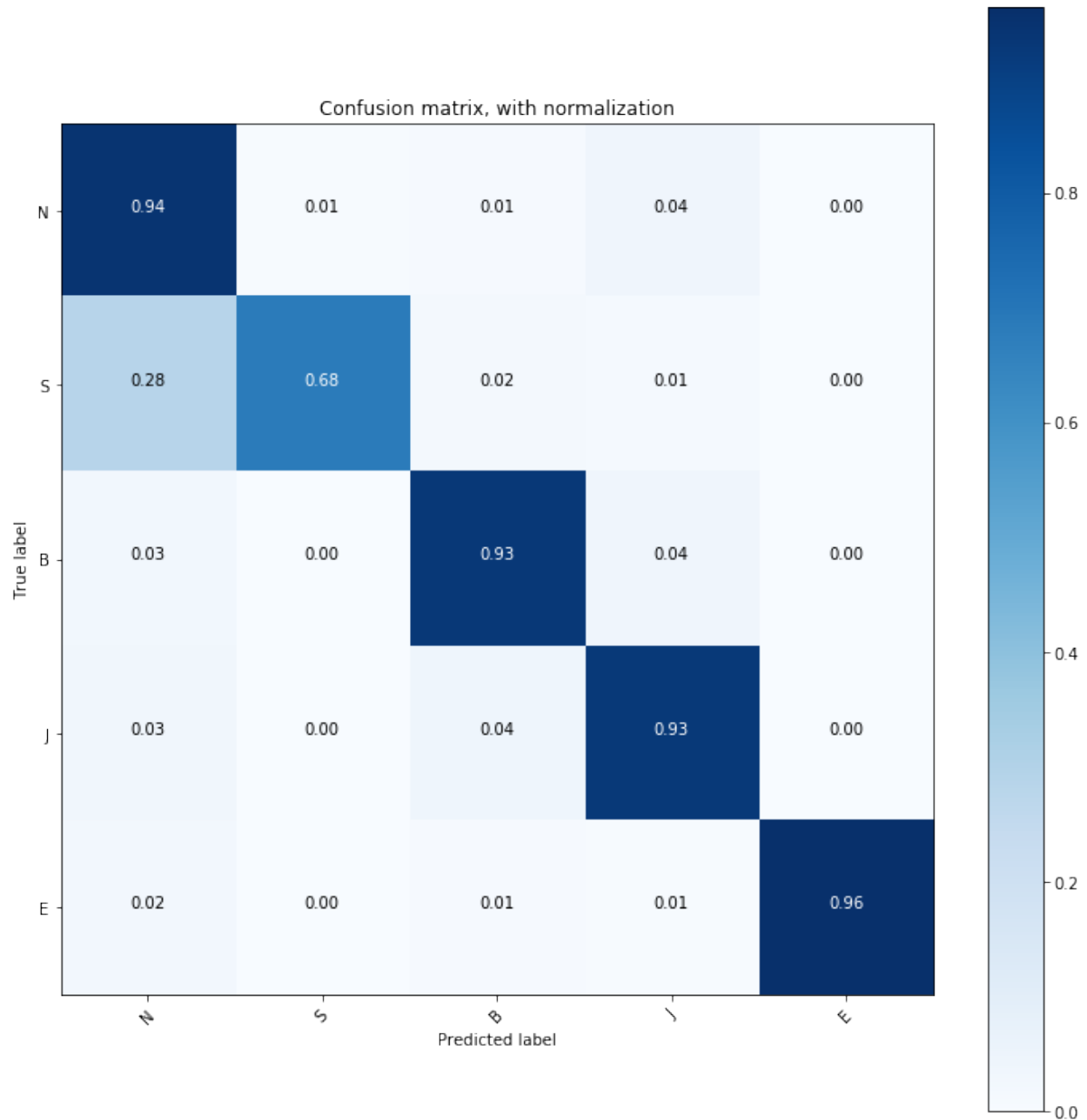


Figure 7.4: CNN with Noise 0.3 - Confusion Matrix

7.1.5 Confusion Matrix of CNN Model with Noise 0.4

Figure 7.5 represent the confusion matrix of the convolutional neural network which is trained with the Gaussian noise of standard deviation 0.4. CM represent the performance matrix for each class in terms of predicted value with respect to the actual value.

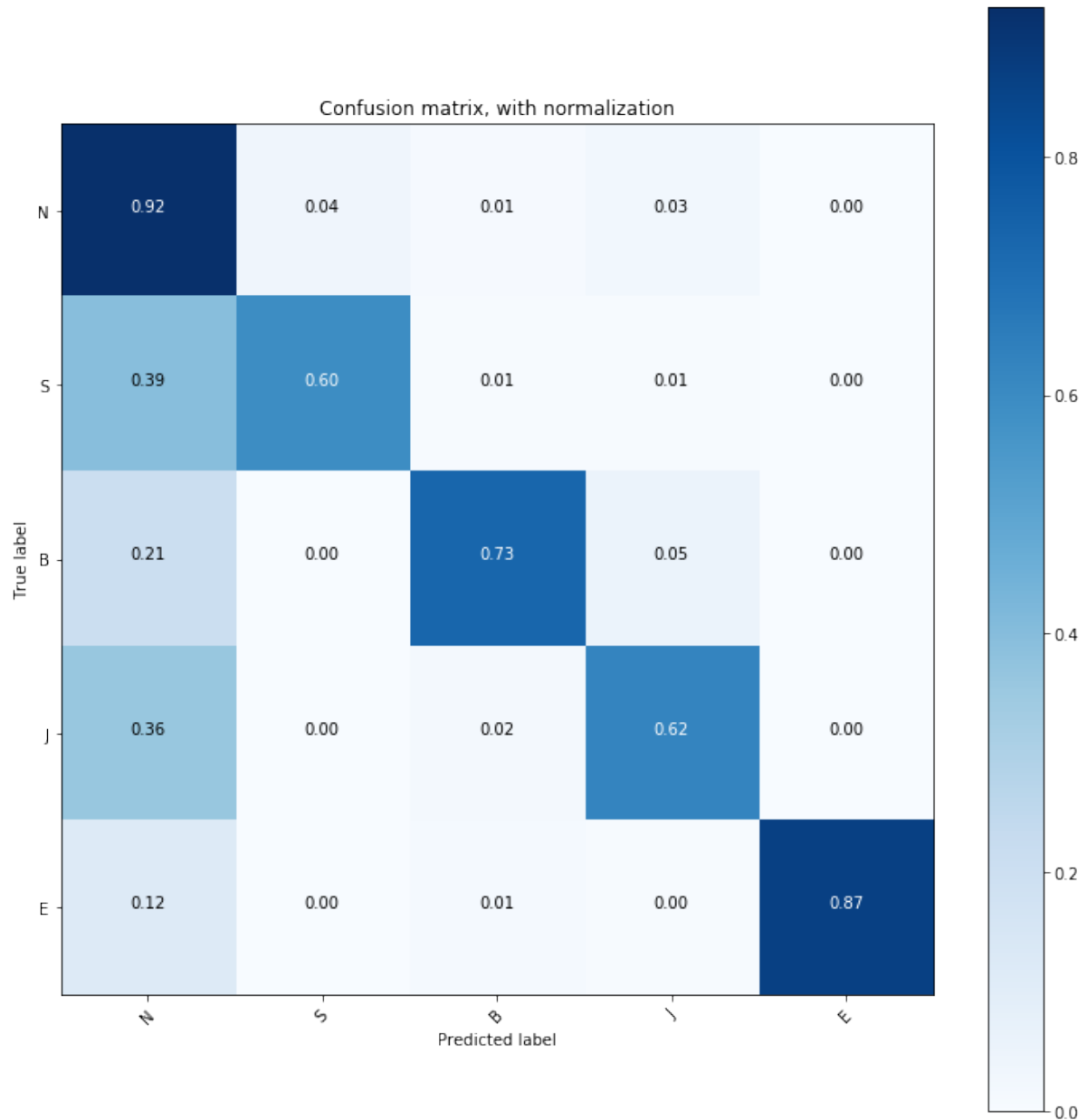


Figure 7.5: CNN with Noise 0.4 - Confusion Matrix

7.1.6 Confusion Matrix of CNN Model with Noise 0.5

Figure 7.6 represent the confusion matrix of the convolutional neural network which is trained with the Gaussian noise of standard deviation 0.5. CM represent the performance matrix for each class in terms of predicted value with respect to the actual value.

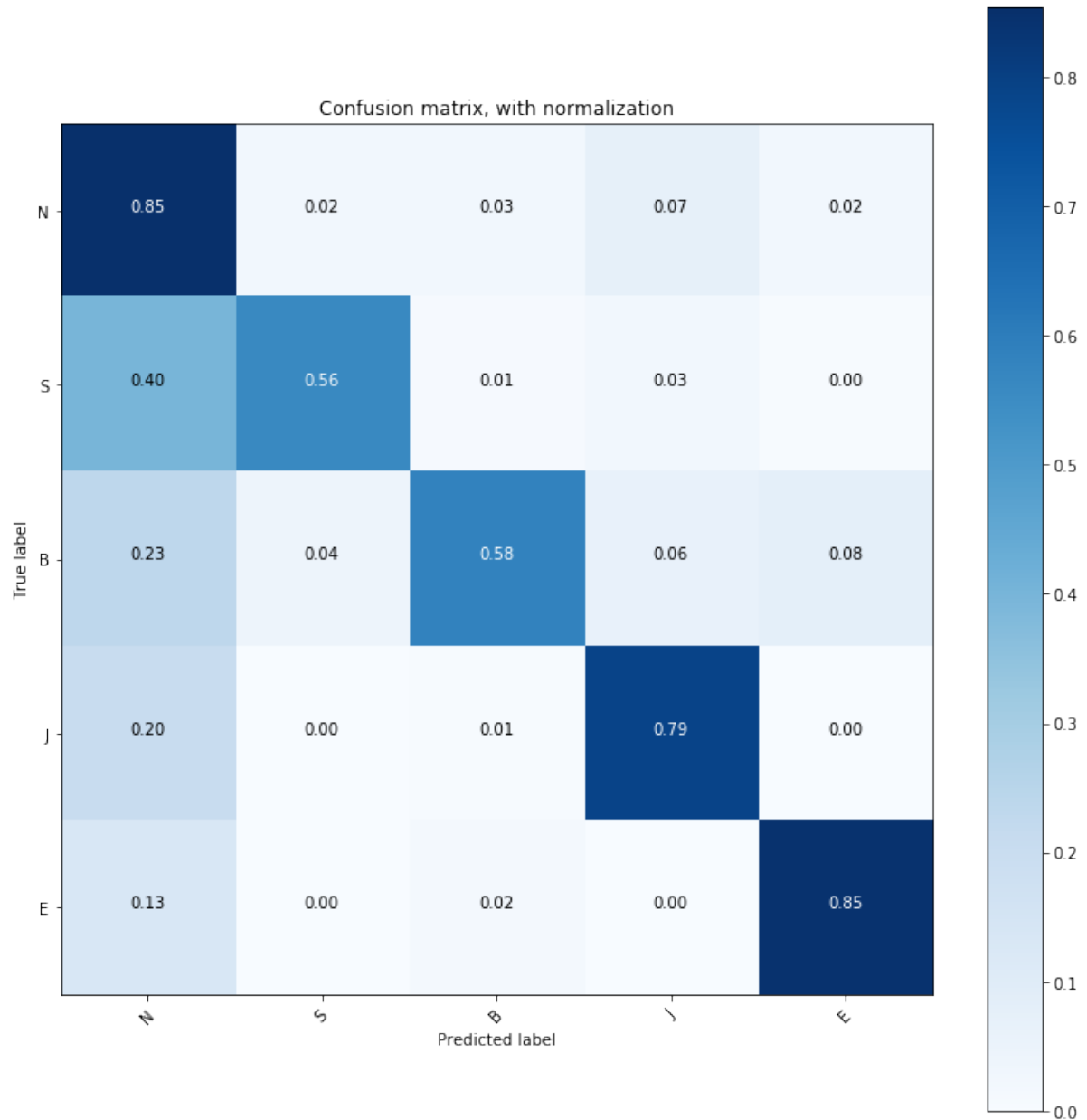


Figure 7.6: CNN with Noise 0.5 - Confusion Matrix

7.2 Analysis and Validation

This section discuss about the model accuracy and model loss, where validation is performed between the trained model and the validation dataset.

7.3 Model Accuracy of CNN Model without Noise

Figure 7.7 represent the accuracy graph of the neural network model that is trained without any Gaussian noise. The graph represents a relation between the trained model and validation model. It helps to analysis the difference in the accuracy model between the training and the validation. It gives a better understanding of the model and also helps to understand is the model build right.

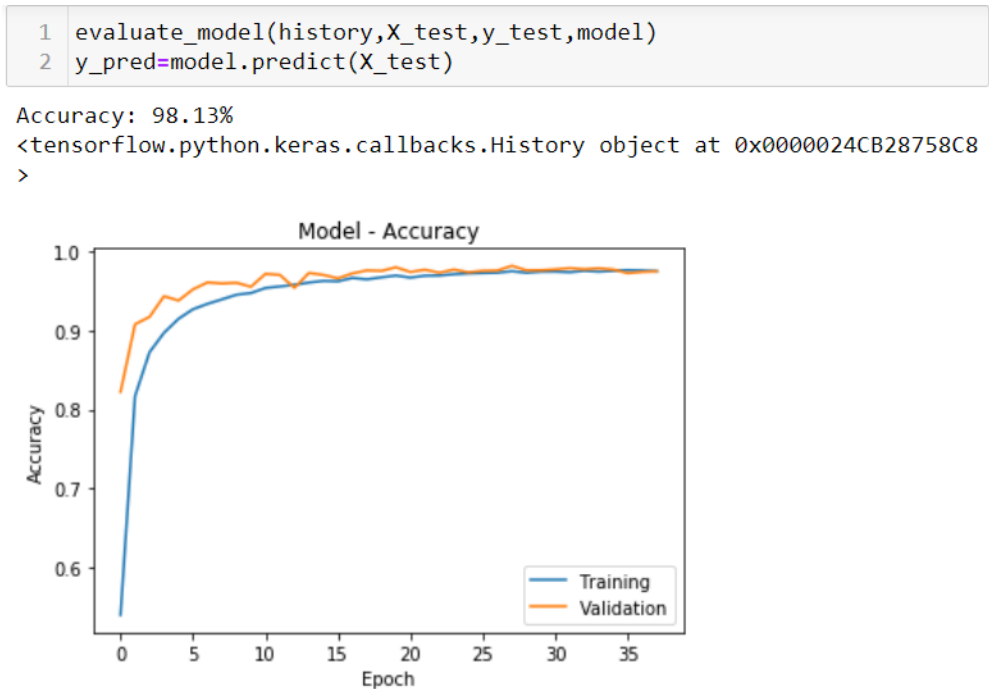


Figure 7.7: CNN without Noise - Model Accuracy

7.4 Model Accuracy of CNN Model with Noise 0.1

Figure 7.8 represent the accuracy graph of the neural network model that is trained with Gaussian noise of standard deviation 0.1. The graph represents a relation between the trained model and validation model. It helps to analysis the difference in the accuracy model between the training and the validation. It gives a better understanding of the model and also helps to understand is the model build right.

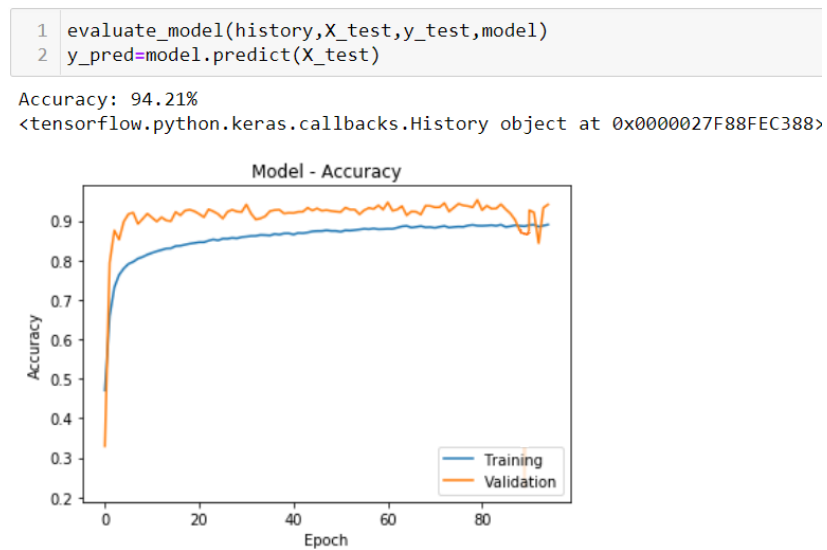


Figure 7.8: CNN with Noise 0.1 - Model Accuracy

7.5 Model Accuracy of CNN Model with Noise 0.2

Figure 7.9 represent the accuracy graph of the neural network model that is trained with Gaussian noise of standard deviation 0.2. The graph represents a relation between the trained model and validation model. It helps to analysis the difference in the accuracy model between the training and the validation. It gives a better understanding of the model and also helps to understand is the model build right.

```
1 evaluate_model(history,X_test,y_test,model)
2 y_pred=model.predict(X_test)
```

Accuracy: 92.83%

<tensorflow.python.keras.callbacks.History object at 0x0000027F9948DB88>

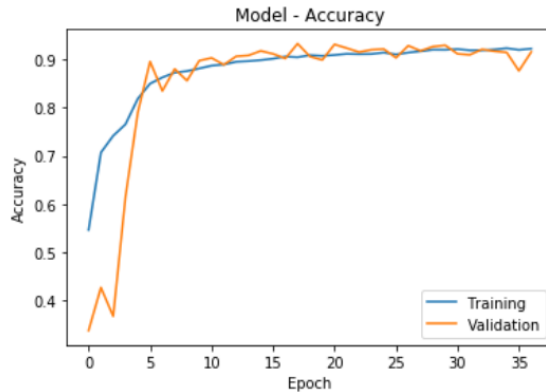


Figure 7.9: CNN with Noise 0.2 - Model Accuracy

7.6 Model Accuracy of CNN Model with Noise 0.3

Figure 7.10 represent the accuracy graph of the neural network model that is trained with Gaussian noise of standard deviation 0.3. The graph represents a relation between the trained model and validation model. It helps to analysis the difference in the accuracy model between the training and the validation. It gives a better understanding of the model and also helps to understand is the model build right.

```
1 evaluate_model(history,X_test,y_test,model)
2 y_pred=model.predict(X_test)
```

Accuracy: 89.74%

<tensorflow.python.keras.callbacks.History object at 0x0000027F89654DC8>

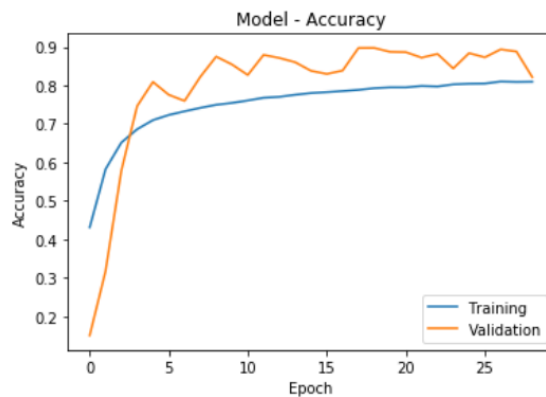


Figure 7.10: CNN with Noise 0.3 - Model Accuracy

7.7 Model Accuracy of CNN Model with Noise 0.4

Figure 7.11 represent the accuracy graph of the neural network model that is trained with Gaussian noise of standard deviation 0.4. The graph represents a relation between the trained model and validation model. It helps to analysis the difference in the accuracy model between the training and the validation. It gives a better understanding of the model and also helps to understand is the model build right.

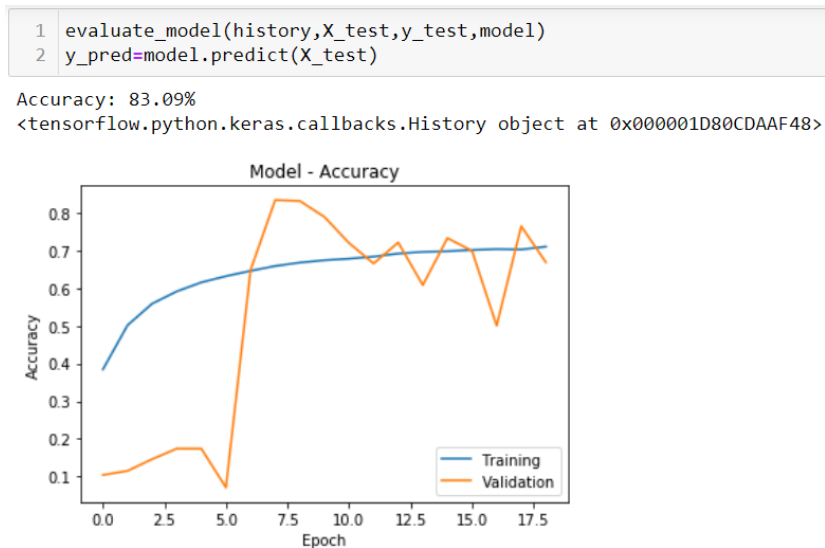


Figure 7.11: CNN with Noise 0.4 - Model Accuracy

7.8 Model Accuracy of CNN Model with Noise 0.5

Figure 7.12 represent the accuracy graph of the neural network model that is trained with Gaussian noise of standard deviation 0.5. The graph represents a relation between the trained model and validation model. It helps to analysis the difference in the accuracy model between the training and the validation. It gives a better understanding of the model and also helps to understand is the model build right.

```
1 evaluate_model(history,X_test,y_test,model)
2 y_pred=model.predict(X_test)
```

Accuracy: 82.72%
<tensorflow.python.keras.callbacks.History object at 0x000001D81D2B7F48>

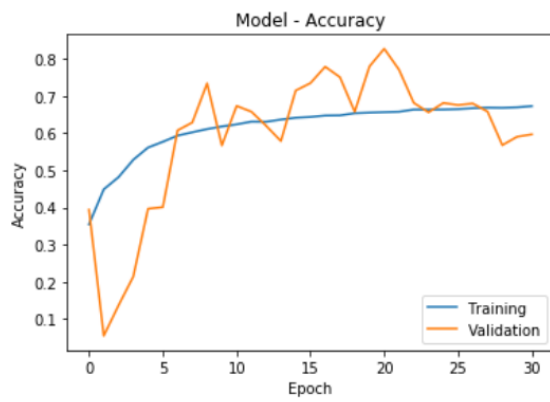


Figure 7.12: CNN with Noise 0.5 - Model Accuracy

Chapter 8

CONCLUSION

The neural network based security authentication for wireless multimedia devices proposed in this paper has significant practical importance. The neural network not only ensures the privacy of the multimedia devices but also ensures a light weight authentication system. The experimental analysis prove that the system build effectively learns the physical layer attributes of the devices, which are manually selected by the algorithm. In short, the neural network has good detection performance and less leading the lesser communication latency.

8.1 Limitations

The neural network based security authentication which is based on the physical layer attributes of the multimedia devices can achieve better accuracy if the dimension of the features becomes large. So, if the dimension of the security authentication can be increased the neural network will have bigger accuracy rate of the wireless multimedia device.

8.2 Future Scope

The section I discusses about the challenges of traditional authentication system and discussed some authentication mechanisms for wireless multimedia environments. A promis-

ing general idea is to test the wireless physical layer attributes and use computing to boost detection accuracy. Common to any or all algorithms that rely on statistical methods is that the challenge of feature selection. Especially in an adversarial scenario, the estimated signal preprocessing is an important aspect of these authentication schemes.

Deep learning-driven signal processing is additionally a possible method for selecting physical layer features. Deep learning is a good thanks to solve the uncertainty of wireless networks. It provides data-centric channel feature mining and deep feature mapping which might be used for secure authentication modeling. particularly, the deep learning tool provides multiple operators that transform a model-based authentication scheme into an information analytic-centric security technology. Optimally, the security should depend on the knowledge, not the authentication model.

In most cases, the quality authentication approach works well during a perfect communication environment, but its detection accuracy is significantly reduced when realistic environment interference is introduced. regardless of the ideal situation, we'd wish to style an appropriate adaptive authenticator from a practical perspective. Different assumptions and communication environments make it difficult to spice up the adaptability of the authentication model. Thus, solving these questions is far more interesting direction for future research.

DISSERTATION BASED PUBLICATIONS

- Anjan K Koundinya and Gautham S K, “*CNN Based Security Authentication for Wireless Multimedia Network*” accepted at International Journal of Wireless and Microwave Technologies (IJWMT) in July 2021.
- Anjan K Koundinya and Gautham S K , “*Two-Layer Encryption based on Paillier and ElGamal Cryptosystem for Privacy Violation*” accepted at International Journal of Wireless and Microwave Technologies IJWMT Vol. 11, No. 3, Jun. 2021.
- Anjan K Koundinya and Gautham S K, “*Machine Learning Based Security Authentication for Wireless Multimedia Network*” accepted at Information and Communication Technology For Competitive Strategies Fifth International Conference (ICTCS 2020) held during December 11-12, 2020.

Bibliography

- [1] D. Wu, Zhihao Zhang, Shaoen Wu, J. Yang, and Ruyang Wang. Biologically inspired resource allocation for network slices in 5g-enabled internet of things. *IEEE Internet of Things Journal*, 6:9266–9279, 2019.
- [2] Puning Zhang, Xuyuan Kang, Xuefang Li, Yuzhe Liu, Dapeng Wu, and Ruyan Wang. Overlapping community deep exploring-based relay selection method toward multi-hop d2d communication. *IEEE Wireless Communications Letters*, 8(5):1357–1360, 2019.
- [3] Zufan Zhang, Chun Wang, Chenquan Gan, Shaohui Sun, and M. Wang. Automatic modulation classification using convolutional neural network with features fusion of spwvd and bjd. *IEEE Transactions on Signal and Information Processing over Networks*, 5:469–478, 2019.
- [4] Zhidu Li, Hailiang Liu, and Ruyan Wang. Service benefit aware multi-task assignment strategy for mobile crowd sensing. *Sensors*, 19(21), 2019.
- [5] Zhidu Li, Yuming Jiang, Yuehong Gao, Lin Sang, and Dacheng Yang. On buffer-constrained throughput of a wireless-powered communication system. *IEEE Journal on Selected Areas in Communications*, 37(2):283–297, 2019.
- [6] Dapeng Wu, Hang Shi, Honggang Wang, Ruyan Wang, and Hua Fang. A feature-based learning system for internet of things applications. *IEEE Internet of Things Journal*, 6(2):1928–1937, 2019.

- [7] Puning Zhang, Xuyuan Kang, Dapeng Wu, and Ruyan Wang. High-accuracy entity state prediction method based on deep belief network toward iot search. *IEEE Wireless Communications Letters*, 8(2):492–495, 2019.
- [8] Dapeng Wu, Shushan Si, Shaoen Wu, and Ruyan Wang. Dynamic trust relationships aware data privacy protection in mobile crowd-sensing. *IEEE Internet of Things Journal*, 5(4):2958–2970, 2018.
- [9] Dapeng Wu, Lingli Deng, Honggang Wang, Keyu Liu, and Ruyan Wang. Similarity aware safety multimedia data transmission mechanism for internet of vehicles. *Future Generation Computer Systems*, 99:609–623, 2019.
- [10] He Fang, Angie Qi, and Xianbin Wang. Fast authentication and progressive authorization in large-scale iot: How to leverage AI for security enhancement? *CoRR*, abs/1907.12092, 2019.
- [11] Mian Ahmad Jan, Muhammad Usman, Xiangjian He, and Ateeq Ur Rehman. Sams: A seamless and authorized multimedia streaming framework for wmsn-based iomt. *IEEE Internet of Things Journal*, 6(2):1576–1583, 2019.
- [12] Xiaoying Qiu, Ting Jiang, Sheng Wu, and Monson Hayes. Physical layer authentication enhancement using a gaussian mixture model. *IEEE Access*, 6:53583–53592, 2018.
- [13] Ning Xie and Changsheng Chen. Slope authentication at the physical layer. *IEEE Transactions on Information Forensics and Security*, 13(6):1579–1594, 2018.
- [14] E. Jorswieck, S. Tomasin, and A. Sezgin. Broadcasting into the uncertainty: Authentication and confidentiality by physical-layer processing. *Proceedings of the IEEE*, 103:1702–1724, 2015.
- [15] Ning Xie and Shengli Zhang. Blind authentication at the physical layer under time-varying fading channels. *IEEE Journal on Selected Areas in Communications*, 36(7):1465–1479, 2018.

- [16] Ning Wang, Ting Jiang, Shichao Lv, and Liang Xiao. Physical-layer authentication based on extreme learning machine. *IEEE Communications Letters*, 21(7):1557–1560, 2017.
- [17] Fei Pan, Zhibo Pang, Hong Wen, Michele Luvisotto, Ming Xiao, Run-Fa Liao, and Jie Chen. Threshold-free physical layer authentication based on machine learning for industrial wireless cps. *IEEE Transactions on Industrial Informatics*, 15(12):6481–6491, 2019.
- [18] Run-Fa Liao, Hong Wen, Jinsong Wu, Fei Pan, Aidong Xu, Yixin Jiang, Feiyi Xie, and Mingguai Cao. Deep-learning-based physical layer authentication for industrial wireless sensor networks. *Sensors*, 19(11), 2019.
- [19] Rick Fritschek, Rafael F. Schaefer, and Gerhard Wunder. Deep learning for the gaussian wiretap channel. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6, 2019.
- [20] Haitao Gan, Zhenhua Li, Yingle Fan, and Zhizeng Luo. Dual learning-based safe semi-supervised learning. *IEEE Access*, 6:2615–2621, 2017.
- [21] Baibhab Chatterjee, Debayan Das, Shovan Maity, and Shreyas Sen. Rf-puf: Enhancing iot security through authentication of wireless nodes using in-situ machine learning. *IEEE Internet of Things Journal*, 6(1):388–398, 2019.
- [22] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017.
- [23] Chunxiao Jiang, Haijun Zhang, Yong Ren, Zhu Han, Kwang-Cheng Chen, and Lajos Hanzo. Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Communications*, 24(2):98–105, 2017.
- [24] Aidin Ferdowsi and Walid Saad. Deep learning for signal authentication and security in massive internet-of-things systems. *IEEE Transactions on Communications*, 67(2):1371–1387, 2019.

- [25] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Fred Juang. Power of deep learning for channel estimation and signal detection in ofdm systems. 2017.
- [26] Liang Xiao, Xiaoyue Wan, Xiaozhen Lu, Yanyong Zhang, and Di Wu. Iot security techniques based on machine learning: How do iot devices use ai to enhance security? *IEEE Signal Processing Magazine*, 35(5):41–49, 2018.
- [27] Qian Mao, Fei Hu, and Qi Hao. Deep learning for intelligent wireless networks: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 20(4):2595–2621, 2018.
- [28] Xiaoying Qiu, Ting Jiang, and Weixia Zou. Physical layer security in simultaneous wireless information and power transfer networks. In *2017 17th International Symposium on Communications and Information Technologies (ISCIT)*, pages 1–4, 2017.
- [29] Xiaoying Qiu and Ting Jiang. Safeguarding multiuser communication using full-duplex jamming receivers. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5, 2017.
- [30] Timothy O’Shea and Jakob Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.
- [31] Olakunle Ibitoye, Omair Shafiq, and Ashraf Matrawy. Analyzing adversarial attacks against deep learning for intrusion detection in iot networks, 2019.
- [32] Xiaoying Qiu. *IET Communications*, 12:1805–1811(6), September 2018.
- [33] Xinlei Wang, Wei Cheng, Prasant Mohapatra, and Tarek Abdelzaher. Enabling reputation and trust in privacy-preserving mobile sensing. *IEEE Transactions on Mobile Computing*, 13(12):2777–2790, 2014.
- [34] Sebastian Henningsen, Björn Scheuermann, and Stefan Dietzel. Challenges of misbehavior detection in industrial wireless networks. In *Ad Hoc Networks*, pages 37–46, 2018.

- [35] He Fang, Xianbin Wang, and Lajos Hanzo. Learning-aided physical layer authentication as an intelligent process. *IEEE Transactions on Communications*, 67(3):2260–2273, 2019.

Appendices

Appendix A

Importing Dataset & Dataset Description

```
In [2]: 1 train_df=pd.read_csv('Arr_train.csv',header=None)
        2 test_df=pd.read_csv('Arr_test.csv',header=None)
```

```
In [3]: 1 # dataframe.size
        2 size = train_df.size
        3 size_test = test_df.size
        4
        5 # dataframe.shape
        6 shape = train_df.shape
        7 shape_test = test_df.shape
        8
        9 # dataframe.ndim
       10 df_ndim = train_df.ndim
       11 df_test_ndim = test_df.ndim
       12
       13 # printing size and shape
       14 print("Train Size = {}\nTrain Shape = {}".
       15       format(size, shape))
       16
       17 print("Test Size = {}\nTest Shape = {}".
       18       format(size_test, shape_test))
       19
       20 # printing ndim
       21 print("ndim of dataframe of train = {}".
       22       format(df_ndim))
       23
       24 print("ndim of dataframe of test = {}".
       25       format(df_test_ndim))
```

```
Train Size = 16460152
Train Shape = (87554, 188)
Test Size = 4115696
Test Shape = (21892, 188)
```

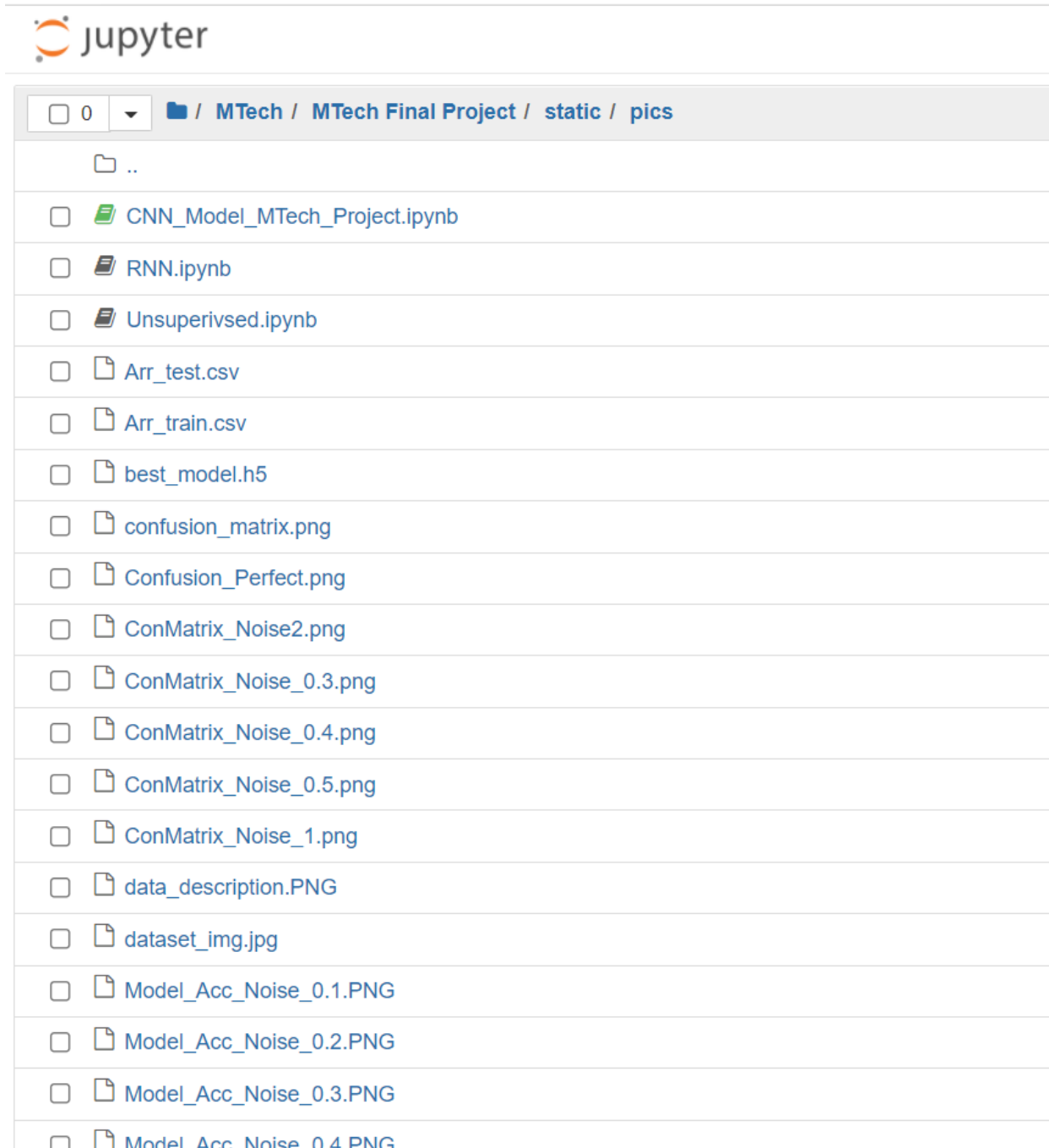
Re-sampling Dataset

In [6]:

```
1 from sklearn.utils import resample
2 df_1=train_df[train_df[187]==1]
3 df_2=train_df[train_df[187]==2]
4 df_3=train_df[train_df[187]==3]
5 df_4=train_df[train_df[187]==4]
6 df_0=(train_df[train_df[187]==0]).sample(n=20000,random_state=42)
7
8 df_1_upsample=resample(df_1,replace=True,n_samples=20000,random_state=123)
9 df_2_upsample=resample(df_2,replace=True,n_samples=20000,random_state=124)
10 df_3_upsample=resample(df_3,replace=True,n_samples=20000,random_state=125)
11 df_4_upsample=resample(df_4,replace=True,n_samples=20000,random_state=126)
12
13 train_df=pd.concat([df_0,df_1_upsample,df_2_upsample,df_3_upsample,df_4_upsample])
```

Appendix B

Anaconda IDE (Jupyter



The screenshot displays the Jupyter IDE interface. At the top left, the Jupyter logo is visible. Below it, a breadcrumb navigation path shows the current directory: `/ MTech / MTech Final Project / static / pics`. The main area contains a list of files and folders, each with a checkbox on the left for selection. The files listed are:

- ..
- CNN_Model_MTech_Project.ipynb
- RNN.ipynb
- Unsuperivsed.ipynb
- Arr_test.csv
- Arr_train.csv
- best_model.h5
- confusion_matrix.png
- Confusion_Perfect.png
- ConMatrix_Noise2.png
- ConMatrix_Noise_0.3.png
- ConMatrix_Noise_0.4.png
- ConMatrix_Noise_0.5.png
- ConMatrix_Noise_1.png
- data_description.PNG
- dataset_img.jpg
- Model_Acc_Noise_0.1.PNG
- Model_Acc_Noise_0.2.PNG
- Model_Acc_Noise_0.3.PNG
- Model Acc Noise 0.4.PNG

Flask Web Framework

```
File Edit Selection View Go Run Terminal Help app.py - MTECH Final Project - Visual Studio Code

EXPLORER
MTECH FINAL PROJECT
  .idea
  .ipynb_checkpoints
  static
  templates
    index.html
    style.css
  app.py
  Arr_K-Means.ipynb
  Arr_Working_test...
  best_model.h5
  cifar-10-python.ta...
  CNN_Regression.i...
  ConvNet Tutorial ...
  data.csv
  Mini Project train...
  Mirai_dataset.csv
  Mirai_Test_Main.i...
  mirai.csv
  Test_Mirai.ipynb
  Testing Mini Proje...
  Working Model.ip...

app.py
1 from flask import Flask, render_template
2 import os
3
4 app = Flask(__name__)
5
6 picFolder = os.path.join('static', 'pics')
7 print(picFolder)
8 app.config['UPLOAD_FOLDER'] = picFolder
9
10
11 @app.route("/")
12 def index():
13     pic1 = os.path.join(app.config['UPLOAD_FOLDER'], 'System Arch.png')
14     pic2 = os.path.join(app.config['UPLOAD_FOLDER'], 'data_description.PNG')
15     pic3 = os.path.join(app.config['UPLOAD_FOLDER'], 'dataset_img.jpg')
16     pic4 = os.path.join(app.config['UPLOAD_FOLDER'], 'resample_dataset.png')
17     pic5 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Summary_1.png')
18     pic6 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Summary_2.png')
19
20     pic7 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Acc_Perfect.png')
21     pic8 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Lose_Perfect.png')
22     pic9 = os.path.join(app.config['UPLOAD_FOLDER'], 'Confusion_Perfect.png')
23
24     pic10 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Acc_Noise_0.1.png')
25     pic11 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Loss_Noise1.jpg')
26     pic12 = os.path.join(app.config['UPLOAD_FOLDER'], 'ConMatrix_Noise_1.png')
27
28     pic13 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Acc_Noise_0.2.png')
29     pic14 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Lose_Noise_0.2.png')
30     pic15 = os.path.join(app.config['UPLOAD_FOLDER'], 'ConMatrix_Noise2.png')
31
32     pic16 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Acc_Noise_0.3.png')
33     pic17 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Lose_Noise_0.3.png')
34     pic18 = os.path.join(app.config['UPLOAD_FOLDER'], 'ConMatrix_Noise_0.3.png')
35
36     pic19 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Acc_Noise_0.4.png')
37     pic20 = os.path.join(app.config['UPLOAD_FOLDER'], 'Model_Lose_Noise_0.4.png')
38     pic21 = os.path.join(app.config['UPLOAD_FOLDER'], 'ConMatrix_Noise_0.4.png')
```

Python 3.9.1 64-bit