Encrypted Network Traffic Classification using Ensemble Learning Techniques

By

ThankGod C. Obasi

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Information Technology

In

Digital Media with specialization in Data Science

Carleton University
Ottawa, Ontario

Abstract

There is a continuous evolution of technological devices leading to a huge amount of traffic data on the internet. This presents Internet Service Providers with changes in the Quality of Service being provided and network security. The classification of these network traffic data promotes a better QoS, and management of the encrypted network. The major concern of the ISPs is protecting users' privacy, thereby generating network traffic data that are encrypted.

In this thesis, we determine the best techniques as well as the relevant statistical features suitable for the classification of the non-VPN encrypted network traffic data. We utilize the opensource UNB and the Solana Networks encrypted network traffic datasets. We performed multiple experiments that led to developing an ensemble learning model with the stacking technique using the deep learning and machine learning classification algorithms with the best performances for the classification of the non-VPN encrypted network traffic data.

Acknowledgments

My profound gratitude goes to God Almighty for His estimable and inexplicable love and cares all these years. The completion of this work could not have been possible without His infinite mercies, strength, and wisdom. Thus, I will ever remain grateful to Him.

My special thanks and appreciation go to my amiable and indefatigable supervisor, Professor Omair M. Shafiq for all his fatherly role in making me achieve this feat. His relentless guide and support even before my arrival at Carleton University were wonderful. His countless checkups, feedback, and encouragement when I took his course ITEC 5920 (selected topics in digital media) during the 2019 Winter term were awesome. I thank him especially for sacrificing his time, professional knowledge and advice to mold me into the data scientist I am today. His guide in the selection of this project topic, his regular feedback all through the experiment as well as the design of the proposed solution are unforgettable. His favorable disposition, support, leadership, and more especially painstakingly reading through this work was specifically the secret behind the completion of this project in the record time. I will ever remain appreciative of your love for me, sir.

I also wish to acknowledge the high degree of support and encouragement received from various quarters following the decision to work on this project topic. I am particularly indebted to the incredible professors at the School of Information Technology especially Dr. Ali Arya, Dr. Chris Joslin, Dr. Audrey Girouard, Dr. Wei Shi as well as other esteemed professors in the Department and the Faculty at large for all their academic guidance. My immeasurable thanks also go to the administrative staff of the School especially Ms. Erenia Oliver and Ms. Ria Akaiwa for their relentless support and encouragement each step of the way. Your show of love was amazing. I am

grateful to you all. I equally appreciate the University of Carleton management for providing a very serene, comfortable, and good study environment for all of us at Carleton.

I am deeply grateful to the management of Solana networks for granting me the wonderful opportunity and rare privilege to leverage on their wealth of experience and knowledge to work with a real-world encrypted network traffic data that was captured by their experienced and outstanding team. The opportunity offered by them to apply data science in solving a real-world problem was a clear demonstration of their love and equally an A-ha moment for me. Also, I thank them for sponsoring this research to the end.

My unquantifiable appreciation goes to my precious and lovely parents for their moral, spiritual, and financial support all through my stay in Carleton. They sacrificed everything to give me good education not minding the pains. I equally appreciate my siblings Onyi, Nonso, and Emeka for their wonderful prayers all the time. I will ever remain grateful to all of you.

I am also deeply grateful to all my colleagues who in various ways contributed to my actualizing this work. Specifically, I wish to appreciate Leroy Ngene and Matt. Ma. for their wonderful suggestions and technical support. The experience shared with you helped in enhancing the value of this thesis. Thank you so much. This work will be inconclusive without due acknowledgment of my mentor Bishop Bryan Williams and family who stopped at nothing to make me feel at home in Ottawa. I thank you for all the support, advice, and encouragement. It has been a very rewarding experience working with you all.

Table of Contents

Abstract ..... **ii**

Acknowledgments ..... **iii**

Table of Contents ..... **v**

List of Tables ..... **viii**

List of Figures ..... **xiii**

Chapter 1: Introduction **1**

1.1 Introduction and background 1

1.2 Motivation 3

1.3 Summary of gap analysis and problem statement 4

1.4 Summary of contributions 6

1.5 Structure of the thesis 7

Chapter 2: Literature Review **8**

2.1 Deep Learning Aided Network Traffic Classification 8

2.2 Machine Learning Aided Network Traffic Classification 25

2.3 Gap Analysis 33

2.4 Theory of a Network 37

2.4.1 The OSI Model 37

2.4.2 Ethernet Frame 39

2.4.3 Internet Protocol (IP) 39

2.4.4 Transmission Control Protocol (TCP) 40

2.4.5 User Datagram Protocol (UDP) 41

2.4.6 Encryption 41

2.4.7 Flow and Session 42

Chapter 3: Dataset Description **43**

3.1 Public dataset 43
3.1.1 Generation of Solana Networks network traffic dataset 46
3.1.2 Processing of Solana Networks captured network traffic 47
3.1.3 Data formatting 48
Chapter 4: Gap Analysis and Problem Description 50
4.1 Research Questions 50
4.2 Objectives 50
Chapter 5: Experimentation 53
5.1 Experiments on the public dataset 53
5.1.1 Artificial Neural Network (ANN) 53
5.1.2 Convolutional Neural Networks (CNN) 58
5.1.3 Recurrent Neural Network using Long Short-Term Memory (LSTM) Concept 63
5.1.4 Capsule Neural Network (CapsNet) 66
5.1.5 Ensemble Learning Technique 69
5.1.6 Machine Learning- Decision Tree (DT) Technique 73
5.1.7 Random Forest (RF) Technique 78
5.2 Experiments on the Solana Networks dataset 80
5.2.1 Artificial Neural Network (ANN) 80
5.2.2 Convolutional Neural Network (CNN) 85
5.2.3 Recurrent Neural Network using Long Short-Term Memory (LSTM) Concept 89
5.2.4 Capsule Neural Networks (CapsNet) 92
5.2.5 Ensemble Learning Technique 94
5.2.6 Machine Learning- Decision Tree Technique 97
5.2.7 Random Forest (RF) Technique 102
5.3 Comparison Analysis of the different experiments performed on the public dataset 104

5.4 Comparison Analysis of the different experiments performed on the Solana Networks dataset 112

Chapter 6: Proposed Solution 119

6.1 System Model 119

6.1.1 Conceptual Architecture 119

6.1.2 Technical Architecture 121

6.1.3 Statistical Features Extracted from the dataset 124

6.2 The Building of The Proposed Ensemble Model 131

6.3 Design Discussions 139

Chapter 7: Evaluation 142

7.1 Functional Evaluation Using the Public Dataset 142

7.2 Functional Evaluation Using the Solana Networks Dataset 148

7.3 Efficiency Evaluation 153

7.4 Complexity Analysis 155

7.5 Comparison of results 156

Chapter 8: Conclusion and Future work 162

8.1 Conclusion 162

8.2 Future Work 164

References 165

List of Tables

Table 2.1 Comparison Table of The Related Works 29

Table 3.1 List of captured traffic and applications for the UNB public dataset 43

Table 3.2 List of Solana Networks captured traffic and applications 48

Table 3.3 List of independent variables for the UNB public dataset 48

Table 3.4 List of independent variables for the Solana Networks dataset 49

Table 5.1 Conversion of Labels to one-hot encodings for the public dataset 55

Table 5.2 ANN Experiment Result for each category (Public Dataset) 56

Table 5.3 ANN Experiment Confusion Matrix (Public Dataset) 57

Table 5.4 ANN + XGBoost Experiment Result for each category (Public Dataset) 58

Table 5.5 ANN + XGBoost Experiment Confusion Matrix (Public Dataset) 58

Table 5.6 1D-CNN Experiment Result for each category (Public Dataset) 62

Table 5.7 1D-CNN Experiment Confusion Matrix (Public Dataset) 62

Table 5.8 1D-CNN + XGBoost Experiment Result for each category (Public Dataset) 63

Table 5.9 1D-CNN + XGBoost Experiment Confusion Matrix (Public Dataset) 63

Table 5.10 RNN-LSTM Experiment Result for each category (Public Dataset) 65

Table 5.11 RNN-LSTM Experiment Confusion Matrix (Public Dataset) 65

Table 5.12 RNN-LSTM + XGBoost Experiment Result for each category (Public Dataset) 66

Table 5.13 RNN-LSTM + XGBoost Experiment Confusion Matrix (Public Dataset) 66

Table 5.14 CapsNet Experiment Result for each category (Public Dataset) 68

Table 5.15 CapsNet Experiment Confusion Matrix (Public Dataset) 68

Table 5.16 CapsNet + XGBoost Experiment Result for each category (Public Dataset) 69

Table 5.17 CapsNet + XGBoost Experiment Confusion Matrix (Public Dataset) 69

Table 5.18 Ensemble Model Experiment Result for each category (Public Dataset)..... 72

Table 5.19 Ensemble Model Experiment Confusion Matrix (Public Dataset) 72

Table 5.20 Ensemble Model + XGBoost Experiment Result for each category (Public Dataset) 73

Table 5.21 Ensemble Model + XGBoost Experiment Confusion Matrix (Public Dataset)..... 73

Table 5.22 Decision Tree Experiment Result for each category (Public Dataset) 74

Table 5.23 Decision Tree Experiment Confusion Matrix (Public Dataset)..... 74

Table 5.24 Decision Tree + XGBoost Experiment Result for each category (Public Dataset) 75

Table 5.25 Decision Tree + XGBoost Experiment Confusion Matrix (Public Dataset) 75

Table 5.26 Decision Tree + AdaBoost Experiment Result for each category (Public Dataset) ... 76

Table 5.27 Decision Tree + AdaBoost Experiment Confusion Matrix (Public Dataset)..... 76

Table 5.28Gradient Boost Experiment Result for each category (Public Dataset)..... 77

Table 5.29 Gradient Boost Experiment Confusion Matrix (Public Dataset) 77

Table 5.30 Decision Tree + Bagging Technique Experiment Result for each category (Public Dataset) 78

Table 5.31 Decision Tree + Bagging Technique Experiment Confusion Matrix (Public Dataset) 78

Table 5.32 Random Forest Technique Experiment Result for each category (Public Dataset) ... 79

Table 5.33 Random Forest Technique Experiment Confusion Matrix (Public Dataset)..... 79

Table 5.34 Random Forest + XGBoost Experiment Result for each category (Public Dataset).. 80

Table 5.35 Random Forest + XGBoost Experiment Confusion Matrix (Public Dataset) 80

Table 5.36 Conversion of Labels to one-hot encodings for the Solana Networks Dataset 82

Table 5.37 ANN Experiment Result for each category (Solana Networks Dataset) 83

Table 5.38 ANN Experiment Confusion Matrix (Solana Networks Dataset) 84
Table 5.39 ANN + XGBoost Experiment Result for each category (Solana Networks Dataset). 84
Table 5.40 ANN + XGBoost Experiment Confusion Matrix (Solana Networks Dataset) 85
Table 5.41 1D-CNN Experiment Result for each category (Solana Networks Dataset) 88
Table 5.42 1D-CNN Experiment Confusion Matrix (Solana Networks Dataset) 88
Table 5.43 1D-CNN + XGBoost Experiment Result for each category (Solana Networks Dataset) 89
Table 5.44 1D-CNN + XGBoost Experiment Confusion Matrix (Solana Networks Dataset) 89
Table 5.45 RNN-LSTM Experiment Result for each category (Solana Networks Dataset) 90
Table 5.46 RNN-LSTM Experiment Confusion Matrix (Solana Networks Dataset)..... 90
Table 5.47 RNN-LSTM + XGBoost Experiment Result for each category (Solana Networks Dataset) 91
Table 5.48 RNN-LSTM + XGBoost Experiment Confusion Matrix (Solana Networks Dataset) 91
Table 5.49 CapsNet Experiment Result for each category (Solana Networks Dataset) 92
Table 5.50 CapsNet Experiment Confusion Matrix (Solana Networks Dataset) 93
Table 5.51 CapsNet + XGBoost Experiment Result for each category (Solana Networks Dataset) 94
Table 5.52 CapsNet + XGBoost Experiment Confusion Matrix (Solana Networks Dataset) 94
Table 5.53 Ensemble Model Experiment Result for each category (Solana Networks Dataset) . 95
Table 5.54 Ensemble Model Experiment Confusion Matrix (Solana Networks Dataset)..... 96
Table 5.55 Ensemble Model + XGBoost Experiment Result for each category (Solana Networks Dataset) 96

Table 5.56 Ensemble Model + XGBoost Experiment Confusion Matrix (Solana Networks Dataset) 97
Table 5.57 Decision Tree Experiment Result for each category (Solana Networks Dataset) 98
Table 5.58 Decision Tree Experiment Confusion Matrix (Solana Networks Dataset)..... 98
Table 5.59 Decision Tree + XGBoost Experiment Result for each category (Solana Networks Dataset) 99
Table 5.60 Decision Tree + XGBoost Experiment Confusion Matrix (Solana Networks Dataset) 99
Table 5.61 Decision Tree + AdaBoost Experiment Result for each category (Solana Networks Dataset) 100
Table 5.62 Decision Tree + AdaBoost Experiment Confusion Matrix (Solana Networks Dataset) 100
Table 5.63 Gradient Boost Experiment Result for each category (Solana Networks Dataset) .. 101
Table 5.64 Gradient Boost Experiment Confusion Matrix (Solana Networks Dataset)..... 101
Table 5.65 Decision Tree + Bagging Technique Experiment Result for each category (Solana Networks Dataset)..... 102
Table 5.66 Decision Tree + Bagging Technique Experiment Confusion Matrix (Solana Networks Dataset) 102
Table 5.67 Random Forest Technique Experiment Result for each category (Solana Networks Dataset) 103
Table 5.68 Random Forest Technique Experiment Confusion Matrix (Solana Networks Dataset) 103

Table 5.69 Random Forest + XGBoost Experiment Result for each category (Solana Networks Dataset) 104

Table 5.70 Random Forest + XGBoost Experiment Confusion Matrix (Solana Networks Dataset) 104

Table 5.71 Comparison of The Results of The Experiments Performed on The Public Dataset 106

Table 5.72 Comparison of The Results of The Experiments Performed on The Solana Networks Dataset..... 113

Table 6.1 Proposed Ensemble Model Experiment Result (public dataset)..... 137

Table 6.2 Proposed Ensemble Model Experiment Result for each category (public dataset).... 137

Table 6.3 Proposed Ensemble Model Experiment Confusion Matrix (public dataset) 138

Table 6.4 Proposed Ensemble Model Experiment Result (Solana Networks dataset) 138

Table 6.5 Proposed Ensemble Model Experiment Result for each category (Solana Networks dataset) 138

Table 6.6 Proposed Ensemble Model Experiment Confusion Matrix (Solana Networks dataset) 139

Table 7.1 Proposed Ensemble Model Functional Result for each category (Public dataset) 146

Table 7.2 Proposed Ensemble Model Functional Result for each category (Solana Networks Dataset) 151

Table 7.3 Proposed Ensemble Model Efficiency Evaluation (Public dataset) 154

Table 7.4 Proposed Ensemble Model Efficiency Evaluation (Solana Networks dataset) 155

Table 7.5 Comparison of our model and other studies 159

Table 7. 6 Comparison of our model and other studies in terms of performance..... 159

List of Figures

Figure 3.1 Public Dataset visualization 45

Figure 3.2 Public Dataset visualization of the categorised traffic 45

Figure 5. 1 ANN Architecture (source: TensorFlow, 2020) for the UNB public dataset..... 56

Figure 5.2 1D-CNN Architecture for the public dataset..... 60

Figure 5.3 Ensemble Structure for the public dataset 70

Figure 5.4 Illustration of Bagging Technique for the public datase*t* 71

Figure 5.5 Traffic categories considered for the Solana Networks dataset 81

Figure 5.6 ANN Architecture (source: TensorFlow, 2020) for Solana Networks Dataset..... 83

Figure 5.7 1D-CNN Model Architecture for Solana Networks Dataset..... 87

Figure 5.8 Ensemble Structure for the Solana Networks dataset..... 95

Figure 5.9 Overall Accuracies of the models designed for the Public Dataset..... 108

Figure 5.10 Overall Accuracies of the boosted models designed for the Public Dataset 108

Figure 5.11 Weighted Precision of the models designed for the Public Dataset..... 109

Figure 5.12 Weighted Precision of the boosted models designed for the Public Dataset 109

Figure 5.13 Weighted Recall of the models designed for the Public Dataset..... 110

Figure 5.14 Weighted Recall of the boosted models designed for the Public Dataset 110

Figure 5.15 Weighted F1-Score of the models designed for the Public Dataset 111

Figure 5.16 Weighted F1-Score of the boosted models designed for the Public Dataset..... 111

Figure 5.17 Overall Accuracies of the models designed for the Solana Networks Dataset 114

Figure 5.18 Overall Accuracies of the boosted models designed for the Solana Networks Dataset 115

Figure 5.19 Weighted Precision of the models designed for the Solana Networks Dataset..... 115

Figure 5.20 Weighted Precision of the boosted models designed for the Solana Networks Dataset 116
Figure 5.21 Weighted Recall of the models designed for the Solana Networks Dataset 116
Figure 5.22 Weighted Recall of the boosted models designed for the Solana Networks Dataset 117
Figure 5.23 Weighted F1-score of the models designed for the Solana Networks Dataset..... 117
Figure 5.24 Weighted F1-score of the boosted models designed for the Solana Networks Dataset 118

Figure 6.1 Conceptual Architecture 120
Figure 6.2 Proposed System Design and Implementation 122

Figure 7.1 Proposed System ROC curves for the network traffic categories (public dataset) ... 144
Figure 7.2 Proposed System confusion matrix for the network traffic categories (public dataset) 145
Figure 7.3 Proposed System correlation matrix (public dataset) 145
Figure 7.4 Proposed System ROC curves for the network traffic categories (Solana Networks dataset) 149
Figure 7.5 Proposed System confusion matrix for the network traffic categories (Solana Networks Dataset) 149
Figure 7.6 Proposed System correlation matrix (Solana Networks Dataset)..... 150

Chapter 1: Introduction

In this chapter, we will give an introduction and background of our work, the motivation, the summary of the gap analysis of the reviewed related works, and the problem statement. Then we will list out the key contributions of our work in detail and lastly the structure of the thesis.

1.1 Introduction and background

Classification of network traffic has become relevant in recent years due to the breakthroughs in technology as well as the speedy growth of high-speed internet traffic demands to help manage network resources appropriately. It is very vital in managing and securing the network (Bagui et. al., 2017). Classifying these network traffic accurately can ensure Quality of Service (QoS) of the network that is being provided to the customers as well as managing it properly. With the recent use of encryption techniques such as the security protocols like HTTPS (Hypertext Transfer Protocol Secure), SSH (Secure Shell), SSL (Secure Sockets Layer) and tools such as Tor, TrueCrypt to mention just a few, in-network applications to encrypt internet traffic to prevent the violation of users' privacy, thus, there has become a need to classify this network traffics generated by the applications to help manage the network, preventing it from malware intrusions and maximize the network intrusion detection and making it easier for ISPs (Internet Service Providers) in troubleshooting of the network. Classification of these encrypted network traffic has brought new challenges to that of the traditional ways used in classifying network traffic (Cao et. al., 2014). The major methods used in the classification of network traffic are port-based, Deep Packets Inspection-based (DPI), statistical-based, and behavioral-based (Biersack et. al., 2013). The port-based classification method has achieved a low accuracy in recent years due to the unsystematic use of port or port concealment while the DPI-based method, on the other hand, cannot be used in the classification of encrypted network traffic due to its ability to violate the

users' privacy. This has left researchers to currently focus on the statistical-based and behavioral-based approaches. These approaches are machine learning-based which has to do with engineering the features associated with the raw traffic flow manually, extracting those features, and classifying network traffic with those features (X. Zeng et. al., 2017). There is a disadvantage to the method which has to do with the manually generated features. This requires a professional in the field of networking who only does not have domain knowledge but also knows the statistical features of the flow-based network traffic to consider while building a classification model.

In this paper, we proposed classifying encrypted network traffic using ensemble learning techniques which is a combination of deep learning and machine learning classification algorithms. The deep learning algorithm is applied as a learning algorithm that automatically learns the features from the raw network traffic and outputs the classified labels directly (W. Wang et. al., 2017).

To be able to classify the encrypted network traffic data, it is important to use flow-based time-related features (Gil et. al., 2016). These time-related features are extracted from the raw pcap (packet capture) files and the port numbers and IP (Internet Protocol) address fields are masked to avoid overfitting of the model because using packets that contains all the headers from every 5 layers of the IP stack since the source and destination IP addresses are unique for each application can lead to the poor performance of the model (Lotfollahi et. al., 2017). Many statistical features can be derived or extracted from the raw pcap files such as the flow duration, forward packet length, backward packet length, flow interarrival time, etc. (Gil et. al., 2016). Different researchers have used different statistical features in designing their models which have led to the different overall accuracy of their models. In (McGaughey et al. 2018), authors used a total number of 44 statistical features extracted from the raw pcap file, authors in (Salman et al. 2018) used 4 statistical features while most of the other researchers like in (Lotfollahi et. al., 2017, Cui et. al., 2019 and

W. Wang wt. al., 2017) did not mention or provide the number of statistical features they used. Selecting effective and relevant statistical features for the classification of the encrypted network traffic data is also another challenge (Gil et. al., 2016). Therefore, there is a need to extract and include relevant statistical features in the design of a model for the proper classification of encrypted network traffic to its appropriate category using the standard MTU (Maximum Transmission Unit) of 1500 bytes.

Maximum Transmission Unit (MTU)

The Maximum Transmission Unit is the largest or maximum number of bytes that can be transmitted over a network by an individual. The standard unit is 1500bytes. When reduced, some applications might no work well thereby leading to failure while performing path MTU discovery. It is important to use the standard unit (Thathapudi et. al., 2012).

1.2 Motivation

Looking at preventing a network from intrusions and violation of users' privacy with the fast-growing high-speed internet traffic, we propose a solution to explore the possibilities of classifying those encrypted network traffic (non-VPN) by use of an ensemble learning method. Specifically, we aim at classifying these encrypted network traffic to enable ISPs to provide Quality of Service to the users as well as troubleshooting and managing the network while protecting the customers' information. As the network traffic makes use of some security protocols such as HTTPS which uses encryption, our proposed system needs to predict the class category of the network traffic data using the hidden patterns identified in the transmitted packets. In general, this can make a huge difference in user experience for critical and time-sensitive applications such as giving the transaction application traffic more priority to other types of traffic thereby optimizing the available bandwidth to enable the network to perform better.

1.3 Summary of gap analysis and problem statement

With different researchers designing and implementing different models using deep learning and machine learning algorithms to classify encrypted network traffic data and achieving different accuracies using different statistical features, there is still need to determine the best classification technique and algorithm that can properly do so while considering the relevant statistical features and also adhering to users' privacy.

Some of the researchers used the well-known University of New Brunswick (UNB) public available network traffic datasets for the design and development of their classification model to classify those encrypted network traffic data. Researchers in (Salman et. al., 2018) considered 4 statistical features for the design of their model and achieved an accuracy of 95%, authors in (Lopez et al., 2017) considered 6 statistical features and achieved an accuracy of 96% while in (McGaughey et. al., 2018), the authors considered 44 features and achieved an accuracy of 81%. Furthermore, authors in (Lopez et al., 2017) used the source port and destination port numbers as part of their statistical features considered for the classification of the encrypted network traffic data. The source port and destination port numbers are meant to be masked for proper classification and to prevent the model from overfitting. In (W. Wang et. al., 2017), the authors were able to classify VPN traffic data achieving an accuracy of 99.9% while their model performed poorly in the classification of non-VPN traffic data achieving an accuracy of 86.6%. Authors in (Y. Zeng et. al., 2019, Cui et. al., 2019, W. Wang et. al., 2017 and Zhang et. al., 2018) used fewer MTU (Maximum Transfer Unit) which might have led to the possibility of their system not learning more about the hidden patterns because the chosen MTU might have ignored the important packets in the network traffic. Researchers in (Lopez et al., 2017) performed series of experiments to determine the number of packets per flow that is suitable for the classification of the encrypted

network traffic data and found out that packet numbers between the range 5 – 20 can achieve better performance. In (Yao et. al., 2019), the authors considered the first 4 packets of each flow thereby ignoring packets they are relevant in the detection of hidden patterns in the dataset required for classification as stated in (Lopez et al., 2017). The authors in (Salman et. al., 2018 and Vu et. al., 2018), used the imbalanced network traffic dataset to design their system and this might have caused overfitting or underfitting of their model. Selecting effective and relevant statistical features for the classification of the encrypted network traffic data is a challenge (Gil et. al., 2016).

Having reviewed some of the related works in the classification of encrypted network traffic and the problems associated with it, there is need to extract and include relevant statistical features, use the standard MTU of 1500 bytes and all the packets of a flow in the design of a model for proper classification of the encrypted network traffic to its appropriate category while maintaining the users' privacy.

From the series of experiment we performed using different numbers of extracted statistical features and less number of MTU of 750 bytes shows different overall accuracies less than that of our proposed solution. Table 1 below shows a summary of the results.

Table 1: Summary of results of the experiment using different MTU and features for the UNB public dataset

Number of statistical features	MTU (Bytes)	Overall accuracy
12	750	84%
12	1500	88%
23	750	92%
63	750	90%
63	1500	94%

The key take-aways are as follows;

- Using more extracted statistical features by determining the importance of the features using the Scikit-learn chi-squared algorithm to include relevant statistical features will improve the performance of the classification model
- Setting the Maximum Transmission Unit to the standard of 1500 bytes and using all the packets of flow will also contribute to the better performance of the model

1.4 Summary of contributions

The contributions of our work are as follows.

- We performed a series of experiments to classify encrypted network traffic using different deep learning and machine learning algorithms to determine the best algorithms suitable for the encrypted network traffic classification using the number of statistical features, MTU, and categories we considered in our work.
- We determined the best statistical features for the flow-based network traffic data that should be considered in the classification of encrypted network traffic by setting a threshold for the determined feature importance since most researchers used different statistical features.
- We designed an ensemble model with a stacking technique using the algorithms (deep learning and machine learning) with the best performances and the relevant statistical features which are flow-based for the classification of the non-VPN traffic data into 7 network categories.
- We tested our model on a real-world encrypted network traffic dataset which is the Solana Networks encrypted network traffic data and it performed very well in the classification of

non-VPN traffic data by achieving an overall accuracy of 90% and AUC of 94% using a total of 27 extracted statistical features.

1.5 Structure of the thesis

This section outlines the way the thesis is structured, and, in each chapter, there is a brief introduction to the content. The structure is as follows.

- Chapter 2: Literature Review of the related works done by some researchers in the classification of encrypted network traffic data (non-VPN and VPN).
- Chapter 3: Dataset description of both the public UNB and Solana Networks network traffic dataset used in performing a series of experiments and in the building of our proposed solution. We also gave a summary of the network theory needed to gain a better understanding of the network traffic aspect.
- Chapter 4: Gap analysis and problem description. We also outlined our research questions and the approach we intend to use to solve them.
- Chapter 5: Experimentations. This covers the series of experiments performed in the classification of encrypted network traffic data using the UNB public dataset and Solana Networks dataset, its evaluations, and comparisons.
- Chapter 6: Proposed solution. Here, we described and implemented our proposed system to solve the encrypted network traffic classification problem.
- Chapter 7: Evaluation of the proposed system outlines the evaluation of the system using the UNB and Solana Networks dataset in terms of its functionality, time complexity, and efficiency.
- Chapter 8: Conclusion and Future work outline the key take away as well as the general conclusion of the thesis and suggestion on future work.

Chapter 2: Literature Review

In this section, we review and describe some of the related works done by some researchers in the field of network traffic classification and analyze their strengths and weaknesses in the encrypted network traffic classification scenario. Most of the researchers worked with the University of New Brunswick (UNB) network flow dataset and some statistical features derived from those flows.

2.1 Deep Learning Aided Network Traffic Classification

Overview: Authors in (Salman et. al., 2018) proposed a multi-level classification framework using deep learning (Convolutional Neural Network) method to classify internet traffic data based on distinct network requirements in terms of QoS and security. The traffic was classified hierarchically into four classes: interactive, bulk data transfer, streaming, and transaction. They used an MTU of 784 bytes and 16 number of packets of each flow and sub-flow from different kinds of devices and applications for offline and online classification, respectively. Four features (packet size, interarrival time, flow direction, and protocol) were also extracted from the flows for the new data representation method (RGBA images) that is based on traffic-related characteristics. To test their system using encrypted and encapsulated data, TOR (UNB, 2017) and VPN (UNB, 2016) public datasets were considered. The results of the experiment showed that the proposed method RGBA 28*28 and RGBA 4*4 achieved a level 1 accuracy of 95.84% and 93.49% respectively, in classifying the encrypted network traffic data into 4 classes (interactive, streaming, bulk data transfer, and transaction).

Strength: This method can be used to classify traffic in an online platform using the features of the first 16 packets of each flow having recorded up to 95% accuracy and it is not affected by data encryption and data anonymization. Also, it can be employed in distinct network environments' datasets because the features considered are flow-dependent.

Weakness: They did not consider the network bandwidth for the online classification which might be time-consuming. Also, the public datasets used were not balanced and might have affected the accuracy of the system.

Overview: In (M. Lotfollahi et. al., 2017), the authors presented a novel system called Deep Packet to identify and classify the end-user application(s) and distinguish the network traffic using deep learning techniques. They applied One-Dimensional Convolutional Neural Network (1D-CNN) and Stack Auto Encoding Neural Network (SAE NN) to the public ISCX VPN-nonVPN traffic datasets obtained from (UNB, 2016 and Gil et. al., 2016) to implement and test their proposed system. They identified 17 classes of applications (e.g.: Facebook, Netflix, Gmail, etc.) and 12 activities (e.g.: VoIP, VPN, streaming, etc.) generating the network traffic and classified them successfully while considering the header and payload of the flow. The results of the experiment showed that 1D-CNN achieved the best result in identifying and classifying the application and network traffic with precision of 98% and 93% respectively.

Strength: Their system outperformed the proposed system in (O. Salman et al., 2018) in terms of traffic classification using the same datasets. Also, they used the information that is contained in traffic flow to classify the network traffic. It was able to classify P2P applications which happened to be one of the hardest classes of applications (Vrána et. al., 2019). It uses advanced port obfuscation techniques by embedding their information in packet protocols that are known and making use of random ports to avoid ISPs controlling processes.

Weaknesses: They did not mention the features considered during their classification. Also, their system is not able to detect traffic from an unknown class(es) if it is not from any of the known class(es) identified during their training and testing phases. The system relies on the information contained in the packet header which makes it dependent on the data and network.

Overview: In (Y. Zeng et. al., 2019), the authors presented a novel framework called Deep Full Range (DPR) for the classification of encrypted network traffic and detection of malware intrusions using the deep learning techniques 1D-CNN, LSTM and SAE-NN. The dataset in (Gil et. al., 2016) was utilized to evaluate the efficacy of the proposed system in classifying encrypted network generated traffic accurately while the dataset in (Shiravi et. al., 2012) was used to evaluate the performance of DPR in terms of malware intrusion detection. The dataset in (Gil et. al., 2016) consists of 7 types of regular encrypted network traffic (web browsing, email, chat, streaming, file transfer, VoIP, and P2P) and 7 types of protocol encapsulated traffic. During the training process of the system, the hyperparameters were set as Epoch, Minbatch, LR, KeepP, N, and Lambda. They utilized CNN to learn the feature content of the encrypted network traffic; LSTM was employed in the extraction of time series features while SAE-NN was chosen to extract features from the text. The efficiency of the system was tested using the test datasets and the results showed that 1D-CNN attained the highest accuracy of 99.85% using L1 regularization for encrypted network traffic classification into the 7 classes of regular encrypted network traffic (Web Browsing, Email, Chat, Streaming, File Transfer, VoIP, and P2P) while LSTM achieved an accuracy of 99.41% for intrusion detection based on the payload information of each flow.

Strength: The system can study the network environment and the multi-type characteristics of the sequence. The system is expected to perform reasonably in an online real world. There was a drop in storage requirements.

Weakness: They did not discuss the statistical features and the number of packets per flow used as input for the classification of encrypted network traffic.

Overview: Authors in (Cui et al., 2019) proposed a novel classification system called SPCaps (Section Packets Capsule) that classifies encrypted network traffic using Capsule Neural Network

which uses vectors as a neuron instead of scalar for classification. They used the ISCX VPN-nonVPN dataset that is available in (Gil et. al., 2016) to implement by training and evaluating their system. The SPCaps was used to learn the spatial characteristics of the network traffic and as well the position of fixed strings and their order in the packets unlike (Zeng et. al., 2019). A two-stage segmentation procedure was used to split the raw data by session-packets to dilute the interference traffic and increase the traffic weight to show higher accuracy than the traditional ones. The results of the experiment performed on the system for evaluation showed that SPCaps scored the highest accuracy of 99.1% for classifying encrypted network traffic into 12 classes (chat, file transfer, P2P, etc.) compared to the three baseline methods (1D-CNN, CNN+LSTM, SAE-NN).

Strength: The pooling operation was not adopted in the system since it discards some mandatory information and its location while trying to reduce the connection of the parameters and refine the features. They considered balancing the raw traffic data before using it as input to their system.

Weakness: They disregarded by deleting the browsing and VPN browsing labels from their datasets which reduced the classes from 14 to 12 thereby might have influenced the classification accuracy of their system. Also, their system will not be able to classify browsing and VPN browsing labels when it is fed with new traffic data and might lead to underfitting of the system.

Overview: In (W. Wang et al., 2017), the authors proposed a system for an end-to-end encrypted network traffic classification using One-Dimensional Convolution Neural Networks (1D-CNN). Feature design, extraction, and selection were fused into their system to automatically learn and select the characteristics features of network traffic to classify traffic networks to their appropriate 12 classes (file transfer, P2P, streaming, VoIP, VPNChat, etc.) that were considered. Their dataset was obtained from (Gil et. al., 2016) which comprises 6 types of encrypted network traffic and 6 types of protocol encapsulated network traffic. It was then split into flow and session traffic for

representation. To evaluate their system, four different experiments were performed, and the result showed that 1D-CNN performed better using the session encrypted traffic data (VPN) with the highest accuracy of 99.9%. It was then compared with other results of the Machine Learning Algorithm (C4.5) and it outperformed those ML algorithm techniques.

Strength: Their system was able to classify end-to-end encrypted traffic data and determine the best representation type of encrypted traffic (session packet traffic).

Weakness: They did not balance the raw network traffic dataset to prevent overfitting. The result of their experiment also showed that the regular traffic (non-VPN) performed badly with an accuracy of 86.6% but did not specify the reason. Training their model with the addresses, during the testing stage, uses the features to classify the network traffic thereby making the system a simple classification model. They fed all the packets containing all headers from every five layers of the IP stack to their system, which has made the result of its efficiency questionable.

Overview: Authors in (Shapira and Shavitt, 2019) introduced a novel system approach called "FlowPic" for the classification of encrypted internet traffic by converting the network flow data (containing the packet sizes and arrival times) into an image and then using the CNN technique to classify the traffic flow into their proper classes and identify their applications. They utilized the flow-based dataset from (Gil et. al., 2016) and their own captured small packets to examine the accuracy of their system. They trained their model with the non-VPN traffic data and tested the model for classification of the VPN traffic data, results of a series of experiments performed showed that FlowPic was able to classify Non-VPN and VPN network traffic with an accuracy of 85% and 98.4% respectively. While the classification of TOR traffic achieved the worst result with an accuracy of 67.8% because of the difficulties encountered in classifying internet traffic generated from TOR correctly compared to (Gil et. al., 2016) which achieved an accuracy of 84.3%

in the classification of TOR network traffic by making use of time-series features, and unbalanced dataset. In general, the system achieved an average accuracy of 93.7% in classifying encrypted internet traffic into 5 categories (VoIP, video, file transfer, chat, and browsing) and 99.7% in identifying the specific applications.

Strength: The system can as well classify unknown applications that are not included during the training stage by learning the samples of other applications with the same traffic category not minding the poor accuracy achieved in classifying Non-VPN traffic data.

Weakness: Time series features were not considered and may improve the system's performance over non-VPN traffic data if considered as input during the implementation.

Overview: Authors in (Lopez et. al., 2017) presented a novel technique to classify encrypted network traffic for the Internet of Things using a hybrid Deep Learning technique (Convolutional Neural Network and Recurrent Neural Network). The network traffic dataset used to train and evaluate their system was captured from RedIRIS, a Spanish academic and research network; and consists of more than 250 thousand network flows which contained more than 100 distinct services. They used the first 20 packets of the flow and 6 features (source port, destination port, payload, window size, direction, and interarrival time) were extracted from the packets' header. They performed a series of experiments to evaluate the accuracy of their system with different models and features, the results show that CNN+RNN-2a achieved the best accuracy of 96.32%, F1-score, precision and recall of 95.74%, 95.43%, and 96.32% respectively. The suffix 'a' attached to the model means they changed the dropout percentage at the dropout layers which was instrumental in increasing the accuracy result.

Strength: They were able to determine the best packet size to use during network traffic classification by performing some experiments and found the packet size range of 5-20 to be the

best. They were able to boost their accuracy result by combining and using the time series features and header features with the CNN/LSTM architecture.

Weakness: During their experiment, it was shown that the feature timestamp (interarrival time) affects the result negatively when added to the extracted feature set. They did not consider the imbalance nature of the dataset which might cause overfitting of the system. Including the source and destination port numbers as among the features being considered for classification might cause overfitting of the model.

Overview: In (Yao et. al., 2019), authors proposed and modeled a system for the classification of time-series encrypted network traffic using Recurrent Neural Network (RNN) and as well introduced the attention mechanism to aid LSTM and Hierarchical Attention Network (HAN) in the classification. They used the dataset from (Gil et. al., 2016) as their input to train, test, and validate the performance of their system. The dataset consists of two-level classification tasks (protocol type and application type identification). Having focused on the protocol type for their classification using the Hadoop platform for preprocessing, during their experimental stage, they applied the dropout technique to avoid overfitting. After conducting series of experiments, the results showed that their proposed method achieved a high accuracy of 91.2% in classifying encrypted network traffic into 12 classes considered (chat, email, file transfer, streaming, torrent, VoIP, VPN-chat, VPN-email, VPN-file transfer, VPN-streaming, VPN-torrent, VPN-VoIP) compared to other methods.

Strength: They used the cost-sensitive learning method in reducing the negative impact the unbalanced dataset might have caused the performance of the system while using the attention mechanism to improve the accuracy of their model.

Weakness: They utilized the first four packets of the flow which might have impacted negatively on the performance of their system while neglecting the range of packet size to use proposed in (Lopez et. al., 2017), in classifying encrypted network traffic.

Overview: In (Zhang et. al., 2018), the authors proposed a system based on an improved Capsule Neural Network (CapsNet) for the classification of network traffic, replacing the scalar output of a regular CNN with a vector output in reconstructing the images for classification. The dataset from (MOORE, 2005) which was used in training and evaluating their system consists of 12 traffic classes, 12 application types, and 24 traffic characteristics. The dataset was then divided into two sections to evaluate the performance of their system on unbalanced and balanced datasets. They performed different types of experiments on the two sections of the datasets and compared them to the Min-Max Normalization- CNN, the results showed that the proposed algorithm (improved CapsNet) scored an overall accuracy of 99.7% and F1-score of 99.5% in classifying the traffic data into (www, email, FTP-data, FTP-PASV, FTP-Control, services, database, P2P, attack, multimedia, interactive and game) compared to the MMN-CNN.

Strength: They adjusted the size of the neurons during the reconstruction module which improved the efficiency of the system by not affecting its accuracy in classification. Also, their system performed better using small datasets in training and testing of their system.

Weakness: Despite their system performing better using small datasets, this affected the efficiency of the system because either; (1) the system did not learn about more features and classes during the training, thereby, will not be able to classify unknown network traffic and its application (2) Or it performed better because the dataset contained the relevant information of a network flow for classification which was not stated in the paper. Just as in (Shapira et. al., 2019), the authors

did classify the unknown traffic because it was able to learn the samples of other applications with the same traffic.

Overview: Authors in (Z. Chen et. al., 2017) introduced an online classification algorithm system called "Seq2Img" for the classification of network traffic and to detect its application type using the CNN model with an embedded Reproducing Kernel Hilbert Space (RKHS) to convert the early flow sequences into two-dimensional images. To achieve the online classification of network traffic, they considered the first 10 packets of the flow and disregarded the handshake packets. The datasets consist of 5 protocols and 5 applications for the classification. They performed experiments on the dataset and the result showed that their novel system can classify encrypted network traffic into 10 classes using the protocol dataset with an accuracy of 99.84% and detect the application that is associated with the flow with an accuracy of 88.42%.

Strength: The model utilizes the static and dynamic behavioral information of the flow attributes in the classification of the network traffic data. The time-series features of each flow are converted into 2- dimensional images using Reproducing Kernel Hilbert Space (RKHS) which contributed to the high accuracy of the model.

Weakness: They compared their results with other machine learning algorithms which had to do with manual feature extraction and neglected to compare it to other deep learning methods. Also, they did not consider the network bandwidth since the traffic classification was done online (on the go) and it might have influenced or affected their system's classification accuracy. Their method was not sufficiently described to allow comparison with other models.

Overview: Authors in (Li et. al., 2018) proposed a system called "semi-supervised classification scheme" to detect malware in network traffic, classify the traffic in terms of their protocols, classify the application, and the attack types using deep generative (density) models. The datasets

from (UNB, 2016, and USTC-TFC, 2016) were used as encrypted network traffic inputs to grade their proposed system's performance. The datasets consist of 6 types of encrypted protocol network traffic, 10 types of normal traffic, and 10 types of malware traffic for the classification. Variational Auto-Encoder (VAE) was used to extract features from the datasets by automatically learning the representation features of the raw traffic flows in a lower-dimensional space. Series of experiments were performed for the evaluation of the system with the results showing an accuracy of more than 95% on a less flow dataset used for training and testing.

Strength: Their system was able to detect malware network traffic as well as classifying it into their respective attack types regardless of the system's main purpose. Making use of VAE for feature extraction gave room for the system to label and classify unknown network traffic with minimal error unlike the system proposed in (Zhang et. al., 2018).

Weakness: The imbalance nature of the datasets was not put into consideration which might have caused overfitting or underfitting of the system. Using the VAE for feature extraction tends missing out the relevant features that must have influenced the system compared to CNN not minding its ability to classify unknown network traffics. The amount of dataset used to train the system was relatively small which might have influenced the system's accuracy result and not considering the packet range stated by authors in (Lopez et. al., 2017).

Overview: In (Liu et. al., 2018), the authors presented a system called Multi-attribute Markov Probability Fingerprints (MaMPF) to classify encrypted network traffic and the applications involved while considering the multi-attributes (length block sequence and relative occurrence probabilities). The encrypted network traffic used as input was obtained from a campus SSL/TLS network which consists of more than 95000 encrypted traffic flows which involve 18 applications while focusing on the message type and packet length for the classification. The Markov-based

technique was used to capture the fingerprints under Message Type Sequences (MTS). The model converts the packet length sequence into Length Block Sequence (LBS) to reduce the packet length size by replacing the low-frequency packets with the closet high-frequency length using the power-law distribution. The result of the experiments performed on the dataset showed that their proposed model reached a True Positive Rate of 96.4% and False Positive Rate of 0.2% compared to other state-of-the-art methods.

Strength: The system assigns importance to the applications involved in a flow. Using the power-law distribution or division was efficient to the system because it transforms the length sequences into Length Block Sequences in building an effective model.

Weakness: Considering only SSL and TLS network traffic protocols, the system will not be able to classify and identify network traffic and its applications for other protocols like SRTP and RTP to mention just a few which leads to underfitting. Also, when the message type sequence is being repeated between different applications, it increases as the number of applications increases thereby weakening the differentiation capacity of the message type during classification which can affect the model's performance.

Overview: Authors in (H. Yao et al., 2019) presented a system that is based on Capsule Neural Network for end-to-end classification of IoT encrypted network traffic for smart cities. Utilizing the network traffic data from (W. Wang et. al., 2017) as the input in their model, the capsule network classifies the original traffic dataflows disregarding the complex feature extraction process done by Convolutional Neural Network (CNN). The dataset was divided into 80% training set, 10% for validation, and the remaining 10% for testing the efficiency of the system. Experiments in two different scenarios were conducted and the results showed that their proposed system performed better by achieving the highest overall accuracy of 98.4% in the classification

of network traffic data and malware detection because of the capsule network that uses dynamic routing in extracting the features of the traffic dataflows efficiently and classifies the traffic to the appropriate class compared to that of CNN which uses pooling algorithm for feature extraction.

Strength: They considered the traffic data that was imbalanced by using the segmentation and padding operations-based method to balance the dataset to prevent overfitting. Also, they experimented and found out that 20 packets and MTU of 1100 bytes is to be used for network traffic classification. Their model did not only achieve a high accuracy rate but also it is very stable in classifying distinct network traffic.

Weakness: Their model performed very low in terms of malware classification and performed poorly in classifying traffics such as Weibo and SMB because of the flow and packet size.

Overview: In (Y. Chen et. al., 2019), authors proposed a classification system called Multi-Attribute Associated Fingerprints (MAAF) to classify mobile encrypted traffic to its original application using the information in the DNS traces generated during the runtime of the mobile application and the handshake certificates during the encrypted flows. To train and test their system, they collected traces of mobile networks while sixteen mobile applications were running using two methods: active and passive trace set collections. Using monkeyrunner (monkeyrunner, 2019) to automatically collect the traces, attributes were generated for each encrypted flow from the specified attributes extracted by the preprocessor. Experiments were performed to determine the accuracy of the system and the result showed that their system scored an accuracy of 98.69% compared to the three state-of-the-art methods.

Strength: Their system achieved better accuracy using the real-world traces of the mobile applications considered. Also, it can be able to predict the actual or original application(s) of the traffic for partial-attribute lacking flows because of the independent nature of the specified

attributes. The preprocessor in the training and classification phase was used to extract domain name, common name, and the organization from the encrypted flows to construct attribute application correlation dictionaries for the training traces.

Weakness: More attributes should be considered for the system to learn about in view to improve the effectiveness of the system.

Overview: Authors in (P. Wang et. al., 2018) introduced a system called DataNet, a Deep Learning based encrypted network classification installed in SDN Home Gateway for allocating and managing end-to-end of the smart home network resources in real-time. It was developed using three approaches which are Multi-Layer Perceptron (MLP), Stacked Auto-Encoder (SAE) and a Convolutional Neural Network (CNN). Making use of the public dataset available in (Gil et. al., 2016) which consists of more than 206,000 data packets from 15 applications encrypted with various security protocols, they created a balanced subset dataset having a total of 73,392 data packets, using it as input in their system. Both full and balanced datasets were used to train and test the system for evaluation purposes. The results of the experiments performed showed that CNN achieved a better result compared to the other two approaches with an average result of 98% in classifying the encrypted network traffic into 15 classes for all three metrics.

Strength: They used the under-sampling method to balance the dataset by randomly removing the major classes' samples until the classes were balanced to a fair point while some researchers like (Lopez et. al., 2017) did not consider the imbalanced problem of the dataset. They used the SAE model during the pre-training stage for feature extraction and dimensionality reduction.

Weakness: Using the full dataset for training the system, it might have affected the performance of their system (DataNets) by boosting its efficiency.

Overview: Authors in (C. Liu et. al., 2019) proposed a system called "FS-Net" for an end-to-end classification of encrypted network traffic data using a multi-layer Recurrent Neural Network (RNN) by learning the representative features from the raw traffic data flows and then classifies them into specific or the original applications. It uses an encoder-decoder structure to generate the features of the packet length sequence and reconstructs the input sequences to directly classify the feature vectors to recognize the applications. Using the encrypted network traffic data from (Lui et. al., 2018) captured from a real-world campus environment comprising 18 applications to evaluate their system, the results show that their system achieved a magnificent True Positive Rate (TPR) of 99.14% and outperformed the other state-of-the-art methods.

Strength: By applying the reconstruction mechanism, it was able to boost the feature learning process which improved the performance of the classification. Also, they used a two-layer perceptron with a ReLu activation function to prevent overfitting from happening to the system.

Weakness: If the certificate lengths of two different traffic flows are close to each other, their certificate characteristic will be considered the same thereby making the two flows to be likely classified as the same application.

Overview: In (Tong et al., 2018), the authors proposed a CNN-based encrypted network traffic classification technique comprising of two stages traffic classification using the flow-based and packet-based features to classify Five Google services; voice call, video streaming, chat, Google play music and file transfer that uses QUIC (Quick UDP Internet Connection) protocol. QUIC is a new transport layer network protocol on top of UDP developed by Google and it provides security protection that is equivalent to TLS. The network traffic data used as input was collected from the network flow of (QUIC, 2018) and captured via Wireshark and contains the voice call, video streaming, chat, file transfer, and Google play music traffic data. 8 flow-based features were

extracted to detect Google Hangout services. In the first stage of the traffic classification, they used the flow-based traffic data with Random Forest (RF) algorithm to recognize and differentiate chat and voice call from other traffic classes while in the second multiclass classification stage, they combined the packet-based traffic data with CNN to classify other traffic classes into file transfer, video streaming and Google play music. The results of the experiments performed to evaluate the efficiency of their model showed that their system achieved a micro-average F1-score of 99.34% compared to the macro-average F1-score in the two-stage classification.

Strength: Their proposed system can detect QUIC-based services such as YouTube, Google Hangout voice call, Google Hangout chat, file transfer, and google play music. Considering the nature of their input dataset and the harm it will cause to their system, they performed padding and normalization to balance it to have corresponding sizes.

Weakness: In their first stage classification it requires the flow-based traffic to be observed thereby making the system online suitable for offline applications. Also, it is not suitable to predict traffic flows earlier. They did not specify the extracted features. Using the flow-based features led to an increase in the computational and classification time.

Overview: In (Vu et. al., 2018), the authors proposed and developed a novel deep learning-based classification approach (LSTM) using a time series feature extraction technique to classify encrypted network traffic data and its applications. The system consists of two-stages where the feature engineering technique is used in the first stage to extract significant attributes from the encrypted network traffic behaviors by analyzing the time series of the receiving network packets. While in the second stage, the developed deep learning technique (LSTM) is used to automatically extract the time dependence of time series packets. They used the real packet dataset from (UNB, 2016) which has 12 types of applications as their input to assess the performance of their system

and extracted only the packet features because it represents encrypted packets greatly. The results of their experiments showed that a feature size of 55 gave the highest accuracy with good processing time and 5 number of packets achieved the highest F1-score of 98.17% in classifying encrypted traffic data and its applications.

Strength: the feature engineering technique combines with payload flow-based methods to state the behavior of the encrypted traffic data efficiently. Extracting only the packet features and feeding it to the LSTM network to extract the time dependency, efficiently retains the characteristics of the network traffic while improving the system's accuracy and reduces the delay in the classification of the encrypted traffic. Also, they verified the claim by the authors in (Lopez et. al., 2017) on the number of packets per flow to use in classifying traffic data to be 5.

Weakness: They did not consider the imbalance nature of the dataset they used as input which might have influenced the performance of the system and can cause overfitting as well.

Overview: In (Vrána et. al., 2019), the authors introduced and implemented the hardware acceleration technique to analyze and monitor encrypted network traffic data faster while using FPGA logic utilization and propose a new acceleration method to extract features from the encrypted network data. The analysis was based on real-time as the hardware acceleration is required to attain a wire-speed of 10Gbps throughput. They utilized the first 10 packets of the network flow in computing the average packet size as they extracted the interarrival time of the packets, the packet length, and the n-bit length entropy of the packet as features. The results of the experiment showed that the precision classification decreased by 0.1 to 0.2.

Strength: Approximate computing was used to increase the entropy's computation speed for the input network data and as well as reduced FPGA logic utilization. They used ISCSILOG in

(Alachiotis and Stamatakis, 2010) which is an iterative implementation in reducing resource consumption.

Weakness: The implementation of the architecture needs to be optimized to support the processing of large data frames. The proposed implementation used is limited to standard Ethernet frames.

Overview: Authors in (Miller et. al., 2018) proposed a system based on Multi-Layered Perceptron Neural Network to detect and classify incoming TCP layer web traffic data as either created by an OpenVPN or normal encrypted traffic or not. The TCP flows that were captured while using the encrypted VPN sessions are analyzed next to other TCP flows that were captured during a mix of general browsing. A total of 10 features were extracted from the flow-based traffic dataset for experimentation using Pearson's Correlation Coefficient which determines the strongest features for classification. The result of the experiment showed that the proposed model can classify the network traffic data into VPN and non-VPN or normal traffic data with an accuracy of 94%.

Strength: This model can differentiate between VPN and non-VPN traffic data using the flow-based extracted features.

Weakness: This model can not be used to classify other VPN protocol traffic data since it was trained using the OpenVPN traffic data. Also, it cannot classify other forms of network traffic other than traffics generated from the web. Doing this can improve the system's capacity.

Overview: In (Casino et. al., 2019) introduced a novel threshold-based methodology called HEDGE technique that uses randomness evaluation of the payload information in a randomly selected network traffic data to classify the traffic packets into 2 classes, namely; encrypted and compressed traffic packets by exploring each of the intercepted packets individually. A total number of 3 features and different packet sizes were extracted to train, test, and validate their proposed model. The results of the experiment show that the proposed model can correctly classify

random packets with 64KB size with an accuracy of 94.72% while the system achieved lower accuracies below 70.61% for small packet sizes.

Strength: This analysis is based on real-time detection to differentiate and classify traffic packets into encrypted and unencrypted data streams. They balanced the dataset to avoid biases by dividing it into 50% each for encrypted and compressed files. The model can easily detect compressed files in the network traffic flow.

Weakness: The features considered were handcrafted which might not have consisted of relevant features that have high positive classification. The proposed technique can be applied to any file size based on the training threshold levels stability according to their experiments but can achieve low percentage accuracy in classifying small file sizes. More features should be considered which might boost the accuracy of the model.

2.2 Machine Learning Aided Network Traffic Classification

Overview: Authors in (McGaughey et. al., 2018) suggested a statistical feature selection technique that uses the Fast Orthogonal Search (FOS) algorithm to select subset features that have predictive values from the raw network traffic data containing a large set of features to classify encrypted network traffic. By selecting the subset features from the raw traffic data, it reduces the error that might occur during the classification process. The raw traffic data feature set (primary feature set) consisting of 44 features was extracted using NetMate (Schmoll and Zander, 2009). To use the FOS algorithm as a feature selector, it requires a training set of traffic flows with known classes. The FOS algorithm selected a subset of 12 features and was fed into a kNN classifier for classification. Considering the Dropbox traffic as the type of encrypted traffic data to test their system, a series of experiments were conducted, and the results showed that there was a reduction in the computation time for the classification which achieved 81% accuracy compared to the kNN

classifier that used the 44 features for classification and achieved an accuracy of 98.98% for the 12 feature set.

Strength: Their system was able to minimize the error that occurs during classification by using the subset features that were extracted from the primary feature set. The FOS algorithm can achieve better accuracy when used with CNN or Capsule Neural Network to maximize the selection accuracy of the predictive features during feature extraction.

Weakness: Considering on Dropbox network traffic, there is room for overfitting when an unknown traffic data is fed into the system for classification. Also, this method is based on professional knowledge about the features compared to sequential feature selection and deep learning methods.

Overview: In (Aouini et. al., 2018), the authors proposed and designed a system for the classification of residential encrypted network traffic using an improved version of C4.5 called the C5.0 machine learning algorithm. The traffic data used as input in their system was captured at a major French ISP residential aggregation network which involves more than 34000 customers. During the training stage of their system, they extracted a total number of 7 features (payload size, flow direction, interarrival time, inter uplink and downlink packet arrival time, protocol, and the destination port number) while considering the first 4 packets of the traffic flow. They experimented to determine the right number of packets for classification just like in (Lopez et. al., 2017) and the result showed a minimum of 4 packets and a maximum of 10 packet size. The results of the experiments conducted on the test datasets showed that their system was able to classify encrypted network traffic into 7 classes (QUIC, BitTorrent, secure web browsing, Skype, Facebook, Google services, and web browsing) with an average accuracy of 98.8%.

Strength: Their system was able to finely classify the traffic flows to their various classes of application as well as differentiating between the secure web browsing and web browsing of simple pages network traffic.

Weakness: The feature extraction stage was handcrafted with features that have high percentage accuracy in influencing the classification process positively. Thereby not knowing the importance of the features not selected. The boost mode for C5.0 is not suitable for their system because it increases the training time by 9 times compared to the default mode. Despite using the destination port to improve the performance of their approach, this might cause overfitting.

Overview: Authors in (Tong et. al., 2018) proposed a CNN-based encrypted network traffic classification technique comprising of two stages traffic classification using the flow-based and packet-based features to classify Five Google services; voice call, video streaming, chat, Google play music and file transfer that uses QUIC (Quick UDP Internet Connection) protocol. QUIC is a new transport layer network protocol on top of UDP developed by Google and it provides security protection that is equivalent to TLS. The network traffic data used as input was collected from (QUIC, 2018) network flow and captured via Wireshark and contains the voice call, video streaming, chat, file transfer, and Google play music traffic data. 8 flow-based features were extracted to detect Google Hangout services. In the first stage of the traffic classification, they used the flow-based traffic data with Random Forest (RF) algorithm to recognize and differentiate chat and voice call from other traffic classes while in the second multiclass classification stage, they combined the packet-based traffic data with CNN to classify other traffic classes into file transfer, video streaming and Google play music. The results of the experiments performed to evaluate the efficiency of their model showed that their system achieved a micro-average F1-score of 99.34% compared to the macro-average F1-score in the two-stage classification.

Strength: Their proposed system can detect QUIC-based services such as YouTube, Google Hangout voice call, Google Hangout chat, file transfer, and google play music. Considering the nature of their input dataset and the harm it will cause to their system, they performed padding and normalization to balance it to have corresponding sizes.

Weakness: In their first stage classification it requires the flow-based traffic to be observed thereby making the system online suitable for offline applications. Also, it is not suitable to predict traffic flows earlier. They did not specify the extracted features. Using the flow-based features led to an increase in the computational and classification time.

Table 2.1 Comparison Table of The Related Works

Paper	Dataset	Input Type	MTU (Bytes)	Number of Packets per-flow	Classes	Technique	Task	Results	Remarks
(Salman et. al., 2018)	Self generated dataset and ISCX TOR-non-TOR ISCX VPN-non-VPN	Session packets	1500	16	4 Classes	CNN	Traffic Classification	Accuracy = 95%	- Imbalanced dataset. - Did not consider the network bandwidth for the online classification
(Lotfollahi et. al., 2017)	ISCX VPN-non-VPN	IP packets	1500	NA	12 Classes and 17 Applications	1D-CNN SAE	Traffic Classification and application identification	1D-CNN Precision = 94% for traffic class. 98% for the app. ID	- Can not detect traffic from an unknown class
(Zeng et. al., 2019)	VPN-non-VPN and (Shiravi et. al., 2012)	Flow-based Time series	900	NA	7 Classes	1D-CNN LSTM SAE	Traffic Classification and intrusion detection	1D-CNN acc = 99.85% for traffic class. LSTM acc = 99.48% for intrusion detection	- Did not discuss the number of packets
(Cui et. al., 2019)	ISCX VPN-non-VPN	Session packets	784	16	12 Classes	Capsule NN	Traffic Classification	Accuracy = 99.1%	- Deleted the browsing and VPN browsing labels from the datasets
(W. Wang et. al., 2017)	VPN-non-VPN	Session packets	784	NA	12 Classes	1D-CNN	Traffic Classification	Accuracy = 99.9% for VPN traffic and 86.6% accuracy for non-VPN traffic	- A more appropriate size of bytes of each session traffic should be determined - Non-VPN traffic performed badly based on its accuracy
(Shapira and Shavitt, 2019)	ISCX TOR-non-TOR VPN-non-VPN	Flow-based	1500	NA	5 Classes	CNN	Traffic Classification and application identification	Accuracy = 85% for non-VPN traffic, 98.4% for VPN traffic and 99.7% for application identification	- Time series feature was not considered
(Lopez et. al., 2017)	RedIRIS	Time series	NA	20	NA	CNN+RNN	Traffic Classification	CNN+RNN-2a accuracy = 96.32%	- Including the source and destination port numbers as among the features being considered for classification might cause overfitting of the model
(Yao et. al., 2019)	ISCX VPN-non-VPN	Flow-based Time series	1500	4	12 Classes	LSTM+HAN	Traffic Classification	Accuracy = 91.2%	- The first 4 packets used as input might have affected the result negatively
(Zhang et. al., 2018)	Moore dataset (MOORE, 2005)	Flow-based	256	NA	12 Classes	Improved Capsule NN	Traffic Classification	Accuracy = 99.7%	- The use of small datasets as input must have affected the model either positively or negatively.
(Z. Chen et. al., 2017)	Network Traffic flow	Flow-based	NA	10	10 Classes	CNN+RKHS	Online Traffic Classification and application identification	Accuracy = 99.84% for traffic class., 88.42% for application identification	- Did not consider the network bandwidth since it was an online classification
(Li et. al., 2018)	ISCX VPN-non-VPN USTC-TFC	Flow-based	NA	NA	6 Classes	Deep generative model: VAE	Traffic Classification Application Identification and malware detection	Accuracy = more than 95%	- Imbalanced dataset - Using VAE for feature extraction tends missing out on the relevant features

(Liu et. al., 2018)	Campus Network traffic data	Flow-based (message type sequence and length block sequence)	NA	NA	18 Classes	Markov Probability	Traffic Classification and application identification	TPR = 96.4%	- Considered only SSL and TLS
(H. Yao et. al., 2019)	VPN-non-VPN	Flow-based	1100	20	10 Classes	Capsule NN	Traffic Classification and Malware Classification	The overall accuracy of 98.4%	- Low performance in malware classification
(Y. Chen et. al., 2019)	Traces of Mobile Network data	Flow-based	NA	NA	16 Classes	Multi-attribute associated fingerprint	Mobile traffic classification	NA	- More attributes should be considered in view to improve the effectiveness of the model
(P. Wang et. al., 2018)	VPN-non-VPN	Session packets	1500	NA	15 Classes	MLP, CNN, SAE	Traffic Classification	CNN accuracy = 98%	- The full raw dataset used to train the model boosted the efficiency of the model
(McGaughey et. al., 2018)	Network Traffic data	Flow-based	NA	NA	NA	FOS+kNN	Feature selection and extraction / traffic classification	Accuracy = 81% using the 44-feature set and 98.98% accuracy using the 12-feature set	- Feature engineering requires professional knowledge - Considered only Dropbox traffic data in testing the model
(C. Liu et. al., 2019)	Campus Network traffic data	Flow-based (packet length sequence)	NA	NA	18 Classes	RNN	Traffic Classification and application identification	TPR = 99.14%	- Certificate characteristics will be considered the same if the certificate length of two different traffic flows are close to each other
(Tong et. al., 2018)	QUIC Network flow	Flow-based Packet-based	1500	NA	5 Classes	CNN, RF	Traffic Classification of 5 Google Services	Micro-average F1-score of 99.34%	- The flow-based needs to be observed, thereby making the system online suitable for offline applications
(Vu et. al., 2018)	VPN-non-VPN	Time series	1500	5	12 Classes	LSTM	Traffic Classification and Application Identification	F1-score of 98.17%	- They did not consider the imbalance nature of the dataset they used as input which might have influenced the performance of the system and can cause overfitting as well
(Vrána et. al., 2019)	NA	Flow-based	NA	10	NA	Hardware Acceleration	Real-time analysis and monitoring of encrypted network traffic data	NA	- The implementation of the architecture needs to be optimized to support the processing of large data frames.
(Miller et. al., 2018)	VPN non-VPN	Flow-based Time series	NA	NA	2 Classes	MLP NN	Encrypted VPN network traffic classification	Accuracy = 94%	- This model can not be used to classify other VPN protocol traffic data since it was trained using the OpenVPN traffic data
(Aouini et. al., 2018)	French ISP residential network	Flow-based	NA	4	7 classes	C5.0	Encrypted network traffic classification	Accuracy = 98.8%	- The feature extraction stage was handcrafted and might have influenced the classification process positively might have not been considered
(Casino et. al., 2019)	Random Encrypted and compressed files	Flow-based	64KB	NA	2 Classes	HEDGE	Encrypted and compressed network traffic classification	Accuracy = 94.72%	- The proposed technique can be applied to any file size based on the training threshold levels stability according to their experiments but can achieve low percentage accuracy in classifying small file sizes

Table 2.1 above depicts the comparison analysis of the related works on classifying encrypted network traffic data (non-VPN and VPN). From the table, most of the researchers used the University of New Brunswick (UNB) public available network traffic datasets for the design and development of their classification model to classify those encrypted network traffic. Furthermore, some of the authors like in (Salman et. al., 2018, W. Wang et. al., 2017, and Shapira and Shavitt 2019) were able to utilize only Convolutional Neural Networks (CNN) deep learning technique to classify UNB traffic datasets and achieved various accuracy results of 95%, 99.9% for VPN traffic and 86.6% for non-VPN network traffic and 85% respectively. The highest accuracy was achieved by authors in (W. Wang et. al., 2017) in the classification of VPN network traffic because they used less number of packets, an MTU of 784 bytes as input size in their model and did not specify the number of packets considered for each flow that was used for the training of the model. This must-have influenced the accuracy of their system positively if they had used a number that is less than 16 but not less than 5 ($16 > n \geq 5$), making it to outperform (Salman et. al., 2018, and Shapira and Shavitt 2019). Also, those packets might contain all the headers from the 5 layers of the IP stack. According to the UNB dataset they used, the source and destination IP addresses are unique for each application, thus, their model must have used the feature in the classification of the VPN network traffic. In (Salman et. al., 2018), the authors used an imbalanced network traffic dataset in training their system which might have caused bias or overfitting to their model and affected the efficiency of the model negatively by reducing its accuracy compared to that achieved in (W. Wang et. al., 2017). While authors in (Shapira and Shavitt 2019) might have used a smaller number of packets since they did not specify the number of packets they considered for each flow when training their system which made it achieve a less accuracy of 85%. In (Lopez et. al., 2017), the authors performed a series of experiments to determine the number of packets that will be suitable for training the classification system and came up with a range of 5 to 20 numbers of packets per flow. They also stated that training a system with a smaller number of packets will result in less accuracy of the classification system. Other kinds of deep learning techniques were used

by some other researchers on the same UNB network traffic dataset, and (Zeng et. al., 2019, Cui et. al., 2019, P. Wang et. al., 2018, and Vu et. al., 2018) achieved high accuracies compared to that in (Lotfollahi et. al., 2017 and Yao et. al., 2019). Authors in (Cui et. al., 2019) stated that Capsule Neural Network (CapsNN) is an improved Convolutional Neural Network (CNN) but achieved a lower accuracy result compared to that in (W. Wang et. al., 2017) which was built using CNN technique, the same datasets, and input size. This is so because it is observed that in (W. Wang et. al., 2017), the authors did not specify the number of packets per flow they used in training their system and all the authors did not specify the number of statistical features they considered. Also, in (W. Wang et. al., 2017), their system performed poorly on accuracy in the classification of non-VPN traffic compared to that of (Cui et. al., 2019). Series of experiments were performed by authors in (H. Yao et. al., 2019) to determine the number of packets and the MTU that is more suitable in training a system for the classification network traffic and the result showed that 20 number of packets and MTU of 1100 bytes achieves better accuracy for classification. This supports the experiment performed in (Lopez et. al., 2017) in determining the number of packets to be used. Most authors used the number of packets between 5 to 20, except for authors in (Yao et. al., 2019 and Aouini et. al., 2018) that used 4 number of packets for each flow. In (Yao et. al., 2019), the number of packets affected the result negatively while in (Aouini et. al., 2018), it achieved a higher accuracy result because the feature extraction was handcrafted and might have also influenced the system. Authors in (Salman et. al., 2018, Zeng et. al., 2019, Z. Chen et. al., 2017 and Tong et. al., 2018) considered an online encrypted network traffic classification for their models for a better Quality of Service to the customers or users of the network. Some of the drawbacks in their implementation is that they did not consider the network bandwidth, authors in (Salman et. al., 2018) used an imbalanced traffic dataset to train its system and in (Tong et. al., 2018) the flow-based network traffic needs to be observed, thereby making the system online suitable for offline applications identification.

After a careful study of some of the related works on the classification of encrypted network traffic and the identification of the applications involved, it was found that authors in (Lopez et. al., 2017) achieved the highest accuracy of 99.9% using a packet length of 784bytes and 1-Dimensional CNN technique on UNB network traffic dataset to classify VPN network traffic into 12 classes. Authors in (Zhang et. al., 2018 and Z. Chen et. al., 2017) came close to it with an accuracy of 99.7% and 99.84% respectively using Improved Capsule NN, CCN+RKHS on different network traffic datasets. Therefore, it was concluded that 1D-CNN, CNN, and improved Capsule NN deep learning techniques combined with machine learning techniques can achieve high accuracies when used to build a network traffic classification model.

2.3 Gap Analysis

There are so many techniques and datasets that have been used to classify encrypted network traffic data. Most of the researchers utilized the available University of New Brunswick (UNB) public network traffic datasets in the design of their system to classify encrypted network traffic (VPN and non-VPN). The dataset contains a lot of noises and is imbalanced from what we have seen during the data formatting stage. Imbalanced dataset affects a model badly which can course overfitting or underfitting or even making the model to have a high bias. This is one of the classification problems where the class counts are not represented equally or something close to that (Javaid, 2018). For example, in figure 3.1, we can see that applications like SFTP, Skype, and YouTube have more instances compared to that of BitTorrent, Google, Twitter, Netflix, FTPS, and ICQ. Also, in figure 3.2, having categorized the network traffics from those applications into 7 categories, it is evident that the dataset is imbalanced such that the email, streaming, chat and file transfer categories have fewer instances compared to that of the VoIP and browsing categories. The number of instances should not be the same for each class, but a little different does matter. The imbalanced dataset can lead to overfitting or underfitting of the model and can reduce its accuracy or otherwise. Overfitting is when a model trains too well by fitting too closely to the train set but performs poorly on the test set or the untrained dataset. While underfitting is when the model does not fit the train set thereby

missing the trends or hidden patterns in the dataset and performs poorly on the test set. This is usually caused by not enough predictors, features, or imbalanced datasets. In (Cui et. al., 2019), the authors stated that Capsule Neural Network (CapsNN) is an improved Convolutional Neural Network (CNN) but achieved a lower accuracy result compared to that of the work in (W. Wang et. al., 2017) which had a better accuracy result in classifying VPN traffic data. This is so because it is observed that (W. Wang et. al., 2017) did not specify the number of packets they used in training their system and must have used packets that contain all the headers from every 5 layers of the IP stack. Also, their system performed poorly based on the overall accuracy in the classification of non-VPN traffic compared to the result obtained in (S. Cui et. al., 2019). Some of the researchers did specify the number of the statistical features extracted or used for the classification model. Authors in (Salman et. al., 2018, Zeng et. al., 2019, Z. Chen et. al., 2017 and Tong et. al., 2018) considered an online encrypted network traffic classification for their model for a better QoS (Quality of Service) to the customers or users of the network. Some of the drawbacks in their implementation is that they did not consider the network's bandwidth that the customers might want to subscribe to, authors in (Salman et. al., 2018) used an imbalanced network traffic dataset to train their classification system while in (Tong et. al., 2018), the flow-based network traffic needs to be observed, thereby making their system online suitable for offline applications identification. After a careful study of some of the related works on the classification of encrypted network traffic and the identification of the applications involved, it was found that (W. Wang et. al., 2017) achieved the highest accuracy of 99.9% in classifying VPN traffic using a packet length of 784bytes and 1-Dimensional CNN technique on UNB network traffic datasets to classify the network traffic into 12 classes. Authors in (Zhang et. al., 2018 and Z. Chen et. al., 2017) both came close to it with an overall accuracy of 99.7% and 99.84% respectively using Improved Capsule NN and CCN+RKHS respectively on different network traffic datasets. In (Zhang et. al., 2018), the authors considered 256 bytes as the Maximum Transfer Unit (MTU) and small dataset as input which must have affected their system positively in terms of the accuracy and negatively because the system doesn't have

enough data for the identification of hidden patterns thereby underfitting the system while authors in (Z. Chen et. al., 2017) considered a maximum of 10 packets per-flow and did not consider the bandwidth for the online implementation which also affected their system positively because the dataset might not have much information needed for the classification of network traffic. On the other hand, in (Aouini et. al., 2018), the authors achieved an accuracy of 98.8% in classifying network traffic data into 7 classes while considering only 4 packets as the number of packets per flow using a machine learning algorithm C5.0. Therefore, it was concluded that deep learning techniques such as (1D-CNN, CNN, and improved Capsule NN) and machine learning technique such as C5.0 can achieve high accuracies when used to build a network traffic classification model.

Machine Learning

Machine learning is one of the most powerful and of course influential technologies in the world today, it is the study that gives computers the ability to learn the valuable hidden or physical patterns associated with a data (which can be complex) without being exceptionally programmed and makes predictions on those data (Dimaano, 2019). It is well known for its ability to turn information or data into knowledge. There are different types of machine learning, but we will discuss supervised learning, unsupervised learning, and ensemble learning.

Supervised machine learning can be categorized as classification and regression. Supervised machine learning always has a target class or a labeled data which one can try to predict or analyze using the data's independent variables. The target class can be in the form of numbers or categories. Examples of supervised machine learning algorithms are Naïve Bayes, kNN (k-Nearest Neighbors), SVM (Support Vector Machine), Decision Trees, Logistic Regression to mention just a few (Alpaydin, 2020).

Unsupervised machine learning is a process whereby the machine takes unlabeled data as input and tries to find any pattern or rules that are associated with the data on its own which helps to derive meaningful

insights and describing the data in a better form to users. It can be categorized under clustering and pattern search algorithms. Examples of unsupervised machine learning algorithms are k-means, k-mode, k-prototype, Apriori to mention just a few (Alpaydin, 2020).

Deep learning

Deep Learning is a part of the machine learning that is associated with different algorithms motivated by the structure and function of the human brain called artificial neural networks (Bengio, 2012). Deep learning algorithms have been seen to use the unknown structure in the input distribution to find or detect representations or patterns, often at multiple levels, with a higher-level learned feature defined in terms of lower-level features (Bengio, 2012). Neural networks have existed over the past years as a mathematical concept, for example, the perceptron model was invented in 1957 (Rosenblatt, 1957), authors in (Rumelhart et. al., 1986) invented the backpropagation in 1986. Neural networks became popular over the last decade partly since there are more data generated and available, making the neural network models easier to train. Some of these learning algorithms have been written for high processing units like the GPUs rather than our normal CPUs (Lawrence et. al., 2017). Furthermore, it is possible to create deeper neural networks compared to that of the past due to the activation functions such as ReLu that is friendly to backpropagation compared to that of the sigmoidal function. In (Hinton et. al., 2014), the dropout technique or the batch normalization technique (Ioffe and Szegedy, 2015) that is used for regularization has helped to prevent overfitting when training larger models. Also, Adam's stochastic gradient descent proposed in (Kingma and Ba, 2014) has helped to reduce the training time of a deep neural network as the neurons tend to converge faster.

A deep neural network can have more than one hidden layer as the name implies. It also needs more training data than shallow networks consisting of only one hidden layer as the first layers of a very deep neural network can be immensely slow to learn.

Ensemble Learning

The ensemble learning method is part of a machine learning where two or more models (either classification or regression models) which are often called the base models or weak learners are trained to solve the same task and then combined to achieve better results or performance (Goodfellow et. al., 2016). There are different techniques associated with the ensemble learning method that can help combine the base models. These are the bagging, boosting, and stacking techniques.

2.4 Theory of a Network

Here, we present the main aspects that have to do with working with traffics generated from a network or series of networks. This is structured in a way to first introduce the theoretical model of network traffic, then the various network layers and its model that is associated with the problem we are trying to solve.

2.4.1 The OSI Model

The public dataset used in our model is the encrypted network traffic data made available to the public by the University of New Brunswick, Canada. This encrypted network traffic was generated from a two-way network communication that is based on the OSI (Open System Interconnection) model. The OSI model is very handy when it comes to dealing with network traffic (Dordal, 2014). The OSI model communicates through a network between two nodes in an unreal form of layers ranging from 1 to 7 (Dordal, 2014 and Bonaventure, 2011). The layers 1, 2, 3, 4, 5, 6, and 7 represent the physical layer, data link layer, network layer, transport layer, session layer, presentation layer, and the application layer respectively (Bonaventure, 2011). Different types of protocols are associated with the layers that are based on TCP/IP (Transmission Control Protocol / Internet Protocol) model (Dordal, 2014 and Bonaventure, 2011).

a. **Application Layer:** This is a layer that the user sees and interacts with. It is practically the GUI (Graphics User Interface). There are protocols associated with this layer, for example, HTTPS, SMTP, FTP, SNMP, DNS, SSH to mention just a few.

b. **Presentation Layer:** This layer has to do with formatting the data by translating it to a format that will be acceptable by the application layer for further processing for an output. For example, it converts the data to an ASCII-coded format that is readable by only the application layer. Therefore, encryption and decryption are done at this level.

c. **Session Layer:** This layer overseas the connection between end-user applications processes by opening a session, terminating a session, and managing it as well. This has to do with the request and acknowledgment in response.

d. **Transport Layer:** This layer is responsible for providing the functions and the procedures that allow the data to be transported (in the form of transferring it) between the two nodes called source and destination hosts. It is also responsible for the segmentation and desegmentation, flow control, and error control such that it retransmits missed segments should there be a delivery failure. TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are the protocols associated with this layer.

e. **Network Layer:** This layer is subject to providing the functions and procedures that allow the delivery of the datagram or packets from the source host to the destination host. This is done using the IPv4 and IPv6 addressing of the nodes (source and destination). It also routes the datagrams between different Local Area Networks. Also, ICMPv4, ICMPv6 are among the protocols associated with this layer.

f. **Data Link Layer:** This is where the node to node (source to destination) data transfer takes place. For example, communication between two computers, a computer that is communicating with a switch or router or any other network devices. It uses the MAC address (the unique address of the hardware device) to deliver frames between nodes connected on the same LAN. It also performs error checking.

g. **Physical Layer:** This is where the actual bits are being transferred to the nodes through a medium. E.g. 802.11b or any other method of transportation.

2.4.2 Ethernet Frame

The Ethernet frame refers to the payload of a data packet which or that is sent through an ethernet link or connection. It starts with the Ethernet header which comprises the source and destination MAC address (these are always unique) and the type, the data, and the frame check sequence. The Ethernet frame's Maximum Transmission Unit (MTU) is 1500 bytes which gives a total of 1514 bytes if there is a Frame Check Sequence (FCS) but it is always omitted (Dordal, 2014).

2.4.3 Internet Protocol (IP)

Internet Protocol as the acronym states is the protocol in layer 3 of the OSI model that all internet traffic uses. The routers connected to the internet use this layer to determine the link in which the packet will be forwarded. It is the Internet Protocol's responsibility to route packets from a source host to a destination host (Bonaventure, 2011). The IPv4 packet header has a normal size of 20 bytes long and can also be up to 60 bytes long if additional options or padding were to be added to the header while the IPv6 header is twice as large as the IPv4 header but no additional options or padding are added to it (Song et. al., 2020). There are few relevant fields in the first 20 bytes of an IPv4. The first field from the left-hand side is the version of the IP that the packet belongs to. IPv4 utilizes 32bits IP address fields. This allows for up to 4.29 billion different class A addresses to be connected. IPv6 increases the number of possible IP addresses by making use of 128bits for each address which will then provide an approximate of $3.4 * 10^{38}$ different IPv6 addresses available to be assigned to devices connected on the internet. The next field is the length field that discloses the size of the packet being transmitted and has a limit of 64KB which is the maximum length of an IP packet plus the header. The last field is the header checksum which is used to check the purity of the IP header.

2.4.4 Transmission Control Protocol (TCP)

TCP as the acronym states is one of the most utilized protocols in layer 3 of the TCP/IP model since it gives a reliable byte stream absorption even to treacherous networks. It is used by the transport layer for internet traffic such as https, file transfers, email, etc. The TCP protocol enforces features to reduce or prevent the potential issues that might happen over treacherous networks such as delayed packets, lost packets, damaged or corrupt packets (Song et. al., 2020). The TCP header is 20 bytes in size and can include an optional header data of up to 40 bytes.

In the TCP header, some fields are of utmost importance which is the source and destination ports because they are used to uniquely identify the flow that is contained in the network traffic (Bonaventure, 2011). The other fields are the checksum and window size fields. TCP uses the checksum to check for corrupt packets while the window size is used to specify the maximum amount of traffic the sending host can send before having to wait for an acknowledgment (ACK) packet or response from the receiving hosts. For two hosts on a network to be able to communicate, there must be a TCP session set up between the two hosts, then a three-way-handshake is initiated between them. If one of the hosts wishes to connect to another host, for instance, if hostA wants to connect with hostB, hostA will send a message with the SYN (Synchronize) bit set and a sequence number. This is to show that hostA wishes to synchronize sequence numbers with hostB. Then hostB replies to hostA with an SYN and ACK message where both bits are set and the ACK number is set to the received sequence number plus one, indicating that hostB which is the receiver is ready to receive the next message in the sequence. HostB then inserts its sequence number in the message for hostA to receive. When hostA receives the message, it sends a message with the ACK bit set and with ACK number equal to that of hostB sequence number plus one. The TCP session between the two hosts has now been set up.

2.4.5 User Datagram Protocol (UDP)

UDP is also one of the protocols associated with the transport layer of the OSI model that preserves message boundaries such that it does not give error correction, duplicate elimination, or congestion control (Song et. al., 2020). It can be used instead of the TCP and does not have the same features as that of TCP. Due to the minimal functionality of the UDP, applications that make use of this protocol have control over how the packets are being sent and processed. For a network application making use of UDP to be reliable in the delivery or sequencing of the data, the application layer must implement some of the same services as TCP (Bonaventure, 2011).

2.4.6 Encryption

Encryption simply refers to a set of written algorithms that are used to encode information such as plain text, such that only the receiver uses the decryption key to decode the encrypted information to its original form (Jaikaran, 2020). Encryption is used widely almost across all sectors. In the internet connection, encryption is required as it is an unsecured line of connection (i.e. privacy and wholeness are not preserved). There are two types of encryption used over the internet which are the Transport layer encryption and the application layer encryption. It enables the payload to be encrypted in the transport layer in such that only the sending and the receiving hosts are aware of its content (Lakhtaria and Kamaljit, 2011).

a. **Transport Layer Security:** Transport Layer Security (TLS) is the descendant of Secure Sockets Layer (SSL). TSL utilizes the Certificate Authority (CA) during a browsing session. It appends the letter ‘s’ after the HTTP when the browser returns a secured webpage. Utilizing the public key and the Certificate Authority ensures the validity of the certificate and that it is coming from a trusted source (Lakhtaria and Kamaljit, 2011).
b. **Application Layer Encryption:** This is end-to-end encryption that is provided at the user level. In this layer of encryption, the data being transmitted is only visible in the application’s memory space.

When information is sent over a network to the receiving host, the application layer encrypts the information making the actual content to be hidden such that its only the application layer of the receiving host can be able to decrypt it. This is handy when the application layer does not want changes or modifications to be made by any of the OSI model layers nor the information being sent to be read by them. An application-layer encryption example is the S-HTTP (Secure Hypertext Transfer Protocol) which allows a secure exchange of files on the World Wide Web (OpenLearn, 2020).

2.4.7 Flow and Session

It is important to be able to differentiate between a flow, session, and packets when it comes to network traffic. A flow is a unidirectional stream of packets from one host to another. It is defined by a 5-tuple which are unique: (source IP, source port, destination IP, destination port, and protocol type) (Myung-Sup et. al., 2004). A session is a bidirectional stream of packets between two hosts. In a session, it can contain both directions of flows and their port numbers (source and destination) are interchangeable (N. Lu et. al., 2012).

Chapter 3: Dataset Description

In this chapter, we will review the details of the public dataset and the collection of the Solana Networks dataset used for this study and the pre-processing done on the dataset.

3.1 Public dataset

There are many public datasets for the classification of network traffic which are widely available. Some of these datasets rely on extracted statistical features (such as the duration of the flow, flow bytes per second, mean flow inter-arrival time, etc.) from the flows or sessions. Other datasets focus on intrusion detection in (Shiravi et. al., 2012), malware detection in (T. Li et. al., 2018 and Yao et. al., 2019), mobile traffic classification in (Y. Chen et. al., 2019), and VPN and nonVPN classification. The public dataset we used for this study is from the University of New Brunswick (UNB) network flow dataset and some statistical features derived from those flows. The dataset was generated by their lab members by creating accounts for two users to use services from applications like Facebook, Skype, Vimeo, etc. Table 3.1 below shows the lists of different traffic categories and the applications associated with the traffic. They captured regular session traffic and then a session over the VPN making it a total of 14 traffic categories.

Table 3.1 List of captured traffic and applications for the UNB public dataset

Traffic Category	Content
Web browsing	Firefox and Chrome
Email	SMTP, POP3S, and IMAPS
Chat	ICQ, AIM, Facebook, and Hangouts
Streaming	Vimeo and YouTube
File Transfer	Skype, FTPS, and SFTP using FileZilla and an external service
VoIP	Facebook, Skype, and Hangouts voice calls (duration for 1 hour)
P2P	uTorrent and Transmission (BitTorrent)
VPN Web browsing	Firefox and Chrome
VPN Email	SMTP, POP3S, and IMAPS
VPN Chat	ICQ, AIM, Facebook, and Hangouts
VPN Streaming	Vimeo and YouTube
VPN File Transfer	Skype, FTPS, and SFTP using FileZilla and an external service
VPN VoIP	Facebook, Skype, and Hangouts voice calls (duration for 1 hour)
VPN P2P	uTorrent and Transmission (BitTorrent)

The description of the traffic category generated is given below.

a. **Web browsing:** This contains the HTTPS traffic generated from Firefox and Chrome web browsing applications while the users were browsing or surfing the internet. Traffic was captured for several browsing flows even though browsing was not the main activity.
b. **Email:** This traffic was generated using a Thunderbird client and the email accounts created for the two users.
c. **Chat:** The traffic was generated using Facebook and Hangouts through a web browser, Skype, AIM, and ICQ using the pidgin application.
d. **Streaming:** Traffic was captured from Vimeo and YouTube which are multimedia applications with continuous and steady streams of data using Firefox and Chrome web browser.
e. **File Transfer:** Traffic was generated and captured while sending and receiving documents using Skype, FTPS, and SFTP using FileZilla.
f. **VoIP:** This is referred to as Voice over Internet Protocol. Traffic was generated using voice applications such as Facebook voice calls, Skype, and Hangouts.
g. **P2P:** Traffic was generated while using the uTorrent and other transmission applications.

All the network traffic was captured using the Wireshark and tcpdump tools. An external VPN service provider was used for VPN traffic.

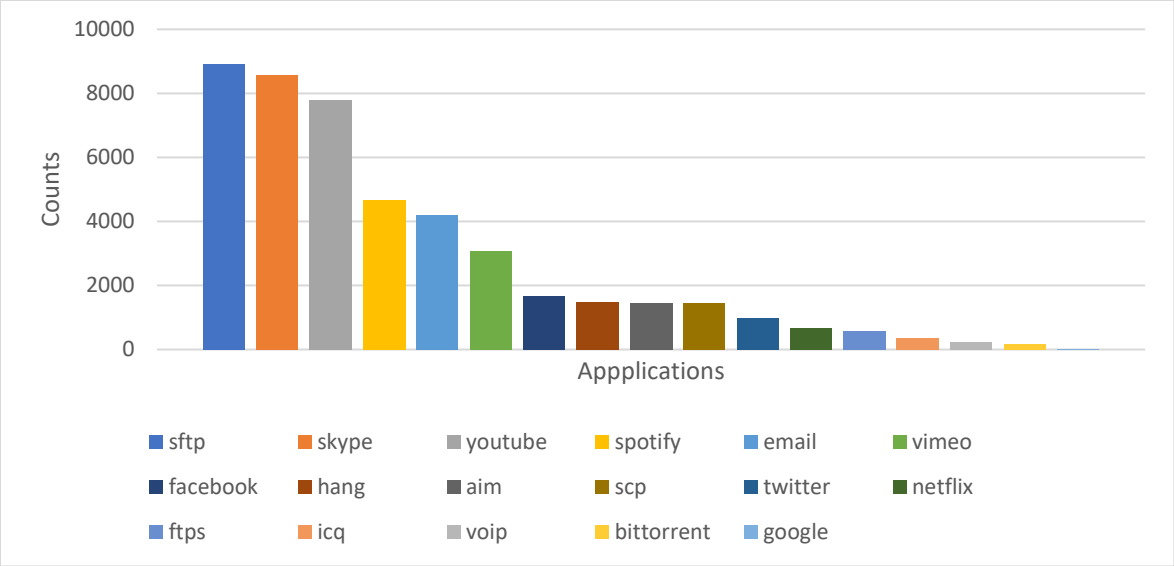


Figure 3.1 *Public Dataset visualization*

Figure 3.1 above depicts the counts of network traffic flows of each application used in the public dataset. It is evident that traffics generated from Bit torrent, Google, Twitter, Netflix, FTPS, and ICQ is below 1000 counts. For this reason, it was then classified into 7 different traffic categories listed in table 3.1 above. It contains a total of 14104 instances (bidirectional flows). The classified network traffic is pictured in figure 3.2 below.

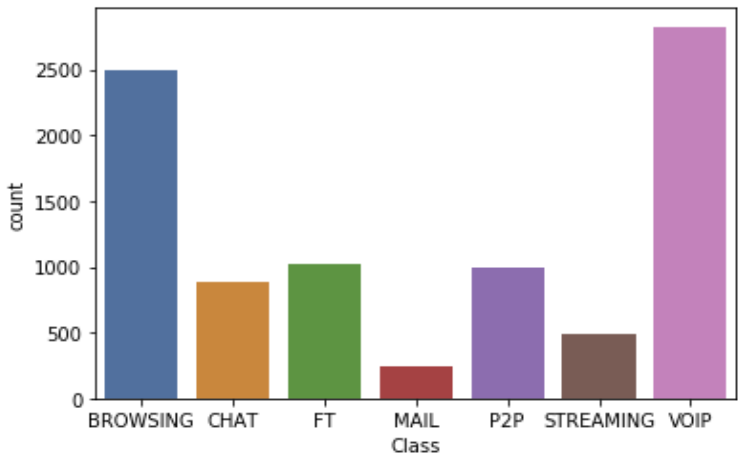


Figure 3.2 *Public Dataset visualization of the categorised traffic*

3.1.1 Generation of Solana Networks network traffic dataset

To have good and well-labeled network traffic data, Solana Networks created its controlled environment. In the controlled environment, they had various frameworks and techniques to create a well-labeled network traffic dataset. This section will describe how the dataset was generated and captured.

a. **Linux Mint OS:** It is an open-source computer operating system that is designed to work on modern x86 and x64 systems. It was also designed in a way to work with other operating systems such as Microsoft's Windows, Apple's Mac OS, and to automatically set up a dual boot or multi-boot environment during installation. This allows users to have flexibility over the type of operating system to use at each boot-up (Linuxmint.com). It was used to enable them to capture network traffic with less noise.

b. **Wireshark:** This is an open-source networking tool that is used to capture and analyze network packets. It is designed to capture traffic over a network as data flows across the network interfaces. To capture network traffic using Wireshark, one must identify the interface it has to listen to. The interface can either be a wireless network card, an Ethernet port, an Adapter port, etc. With the aid of Wireshark's GUI, the captured network traffic is presented in a structured form and provides some navigation buttons that enable us to perform some analysis on the traffic captured. For each packet of the network traffic that is captured, Wireshark GUI displays information such as the packet number, time between packets, source IP address, destination IP address, protocol type, length of the packet and information about the packet (whether it is an ACK or SYN or FIN or the data itself). It also provides the users with the filtering functionality to extract packets with the kind of or required characteristics (e.g. packets from a specified source) they want. The network traffic captured using Wireshark can be exported in different file formats such as pcap file format which was used to save the captured network traffic data.

c. **The Captured Network Traffic:** To generate their network traffic, they decided to capture network traffic flows from specified applications which are Facebook, Skype, Telegram, YouTube, Netflix, SoundCloud, edonkey, torrent, Dropbox, Google drive and Spotify. They connected to the internet using different networks to generate traffics from the listed applications for a certain period. The traffics generated were labeled according to the application it was generated from. The applications' websites were accessed using the web browser application.

3.1.2 Processing of Solana Networks captured network traffic

For us to process the Solana Networks captured network traffic, we employed certain processes to extract the statistical features we need and to facilitate the design of our classification model. The processes are outlined as follows.

a. **CICFlowMeter:** This is a network traffic flow generator that is written in Java and offers more flexibility when it comes to choosing the statistical features you want to extract from the pcap files. It is owned by the University of New Brunswick (UNB). It generates the bidirectional flows (source to destination and destination to source) of the captured network traffic. It can calculate a total number of 83 statistical features from the captured network traffic in a forward and backward direction. The output of the calculated statistical features is in a CSV file format with the 83 features (netflowmeter.ca). We used this tool to extract the statistical features from the captured network traffic.
b. **Labeling:** This is to make sure that the captured network traffic was labeled correctly to the appropriate application that it was generated from.

Table 3.2 below shows the list of Solana Networks captured network traffic categories and the associated applications.

Table 3.2 List of Solana Networks captured traffic and applications

Traffic Category	Content
Web browsing	Firefox and Chrome performed using different networks
VoIP	Messenger audio and Skype audio
Chat	Messenger and Telegram
Audio Stream	SoundCloud and Spotify
File Transfer	Dropbox and Google Drive
P2P	Torrent and edonkey
Video Stream	Netflix and YouTube

3.1.3 Data formatting

The public dataset was converted from a pcap (packet capture) file format to .csv (comma separated values) file format using the CICFlowmeter to extract a total of 83 statistical features from the traffic flow. It was then read into python to concatenate the label column which is the target class to the data frame. The dataset was formatted by assigning a float datatype to some of the independent variables to enhance the analysis process. We then checked and removed the NaN and null records from the dataset. The importance of the independent variables was acquired to determine its relevance during the training of our model and prediction. It was later reduced to 23 independent variables and 14104 instances for our analysis using the public dataset. Initially, the public dataset has a total number of 16405 instances and after removing the missing values, it reduced to 14104 instances. This is so because of the inability of the CICFlow meter to calculate some statistical features of the packets captured using different networks. Table 3.3 below shows the list of the used statistical features from the public dataset.

Table 3.3 List of independent variables for the UNB public dataset

Feature	Description
duration	The duration of the flow.
fiat	Forward Inter Arrival Time, the time between two packets sent forward direction (mean, min, max, std).
biat	Backward Inter Arrival Time, the time between two packets sent backward (mean, min, max, std).
flowiat	Flow Inter Arrival Time, the time between two packets sent in either direction (mean, min, max, std).
active	The amount of time flow was active before going idle (mean, min, max, std).
idle	The amount of time flow was idle before becoming active (mean, min, max, std).
fb psec	Flow Bytes per second.
fp psec	Flow packets per second.

For the Solana Networks captured network traffic dataset, we extracted a total number of 27 statistical features and 95711 instances using the CICFlowMeter tool. Table 3.4 below shows the list of the extracted features.

Table 3.4 List of independent variables for the Solana Networks dataset

Feature	Description
duration	The duration of the flow.
fiat	Forward Inter Arrival Time, the time between two packets sent forward direction (mean, min, max, std).
biat	Backward Inter Arrival Time, the time between two packets sent backward (mean, min, max, std).
flowiat	Flow Inter Arrival Time, the time between two packets sent in either direction (mean, min, max, std).
Total fwd Pkts	Total packets in the forward direction of a flow
Total bwd Pkts	Total packets in the backward direction of a flow
Total length of fwd pkt	The total size of packets in the forward direction of a flow
Total length of bwd pkt	The total size of the packet in the backward direction of a flow
Fwd pkt length	The size of the packet in the forward direction of flow (mean, min, max, std)
Bwd pkt length	The size of the packet in the backward direction of flow (mean, min, max, std)
fb psec	Flow Bytes per second.
fp psec	Flow packets per second.

Chapter 4: Gap Analysis and Problem Description

The gaps we discovered from some of the existing related works we reviewed will be discussed in this chapter. Furthermore, we will discuss the research questions, problem statement, and objectives.

4.1 Research Questions

Here, there are some research questions about encrypted network traffic classification that we need to address.

RQ1. Will increasing the number of statistical features affect the classification model or system negatively or positively? Increasing the number of statistical features can lead to better accuracy if those features are correlated but if not, they can lead to the poor performance of the model. According to our review on some of the previous work done, we found out that the researchers have used different statistical features of the network traffic as well as the total number of the statistical features in the design of their model. This led to different accuracy results while using the same dataset and technique.

RQ2. Which of the deep learning and machine learning classification techniques is best for encrypted network traffic classification using the number of statistical features, MTU, and categories we considered in our work? From table 2.1, it is shown that some of the researchers used different machine learning and deep learning classification techniques in their model. Some techniques are suitable for classification while some are suitable for regression purposes or problems (Rajat, 2018).

4.2 Objectives

Here, we outlined and discussed how we will answer our research questions. This thesis aims at analyzing the possibilities of classifying non-VPN encrypted network traffic using ensemble learning techniques. We aim at classifying the network traffic to its appropriate application or class to distinguish between them using the network traffic types.

a. Proper extraction of required statistical features of the network traffic.

Since we are dealing with network traffic data, we want to avoid calculating and extracting irrelevant statistical features associated with the flow. For the public dataset, we used the feature importance library in sklearn to print the name and the importance of each feature. The importance of the features was determined using the Scikit-learn chi-squared algorithm to include relevant statistical features for the training of the model. According to (Scikit-learn, 2019), the importance of a feature is computed as the normalized total reduction of one or more criteria that is brought by such feature. Then we created a selector object that will use the classifier to identify the features that have importance by setting the threshold at 0.05. This was set by looking at the values of the feature importance and the correlation matrix of the features. It also corresponds to the value mentioned in the literature review to enable us to capture more relevant features that will help the model in detecting the hidden patterns in the dataset. By doing so, we were able to narrow a total number of 83 statistical features extracted to 23 important statistical features for the traffic dataset. This does not give us the features that are best used for the encrypted network traffic classification rather it selects the features that have more importance in all the statistical features extracted from the particular network traffic dataset used as it might differ with other network traffic dataset in terms of correlation and hidden patterns since it is a time-series dataset. We believe that some of the calculated statistical features can affect the model negatively. For Solana Networks dataset, we decided to calculate and extract the statistical features that we need (such as the flow duration, mean packet size, maximum and minimum packet size, standard deviation of the packets, inter-arrival time between packets for each flow, etc.) for the classification of the encrypted network traffic and came to a total of 27 statistical features. Generally, we believe that the statistical features we extracted are the most required since it covers the statistics of a flow.

b. Proposing an ensemble learning technique is suitable for encrypted network traffic classification.

Different deep learning and machine learning techniques have been used for the classification of network traffic over the past few years without knowing the best technique to use since it is a multi-classification problem and a time series data. We will perform a series of experiments using different deep learning and machine learning techniques on the public dataset and Solana Networks dataset. Then compare the results to determine the suitable techniques that will make up the ensemble learning technique for the classification of the encrypted network traffic data. It is worthy to know that there are some deep learning and machine learning techniques that are meant for classification while some are meant for regression purposes. By doing so, we will be able to classify network traffic data to administer a good Quality of Service (QoS) to the consumers, proper network monitoring and management, adequate network planning, proper troubleshooting of network issues for ISPs, appropriately securing the network from malware and intrusion and to identify the customer's usage to mention just a few.

Chapter 5: Experimentation

In this chapter, we performed different series of experiment on the public and Solana Networks traffic dataset using different types of deep learning techniques such as the Artificial Neural Networks (ANN), 1D-Convolution Neural Network (1D-CNN), Capsule Neural Network (CapsNet), and Recurrent Neural Network (RNN) using Long Short-Term Memory (LSTM) and machine learning techniques such as Random Forest (RF), Decision Tree (DT), k-Nearest Neighbors (kNN), Support Vector Machine (SVM), and Ensemble Learning. We also boosted the models with the different types of boosting techniques such as the AdaBoost, Gradient Boost, and XGBoost.

5.1 Experiments on the public dataset

5.1.1 Artificial Neural Network (ANN)

ANN is a family member of deep learning networks that was first proposed by Warren McCulloch and W. Pitts in 1943 (Jain et. al., 1996). It is the first technique we used in designing the classification model for the classification of encrypted network traffic data. As one of the deep learning models, Artificial Neural Networks are applicable to process big data.

a. **Features:** These are the independent variables the deep learning network will use to determine what it is meant to classify or predict during analysis. These features have probabilities assigned to them when fed into the machine for training and testing. In our case, the independent variables are the extracted statistical features.
b. **Weights:** Weights are used to determine or decide the importance of each of the independent variables that are fed into the machine to make the final decision and if their probabilities are decreasing or increasing.
c. **Neurons:** This takes the inputs and multiplies them by their weights, sums them up, and applies the activation function to the sum. The neurons adjust the weights based on the input.

d. **Batch size:** This is the number of training samples that are utilized in one iteration before updating the parameters of the internal model. Choosing the number depends on the number of training samples. The batch size number should be able to divide the number of training samples in an even proportion such that the model will train properly to converge well.

e. **Epoch:** This is the number of times the learning algorithm used in designing the model will have to work through the entire training set.

f. **Data preparation:** The public dataset we used in designing the ANN classification model was from UNB. The dataset includes VPN and nonVPN network traffic consisting of 83 independent variables. We decided to deal with the nonVPN traffic by removing the VPN traffics from the dataset and used feature importance in sklearn to determine the importance of the features to reduce it for proper classification. We were able to come to 23 important features by setting a threshold of 0.05 which became our input independent variables and 7 traffic categories as our target class.

g. **Data Cleaning:** This is where we checked for the null and NaN values in the dataset and removed them so that we can have clean data for our preprocessing stage. We then converted the independent variables that have floating numbers to a float data type.

h. **Data preprocessing:** One of the most important stages of machine learning is the data preprocessing stage. It is a required stage. Data preprocessing is transforming the dataset into a format the machine will understand before feeding it to the algorithm for analysis (Danish, 2016). There are different preprocessing methods such as rescaling, standardization, normalization, and binarizing of data. In python, the scikit-learn library has a prebuilt functionality used for preprocessing such as label encoder, standard scaler, and MinMax scaler functions. The preprocessing is performed on the features, which is called feature scaling. As different machine learning algorithms make different assumptions to your data, this may require different data

preprocessing for different algorithms. For the ANN classification model, we used the label encoder to convert the time series data into a format the machine can read, the standard scaler is used in transforming the records into a scaled form and the One Hot encoder is used to encode the target class which has different applications. This will make the classification flexible during analysis. Then we split the dataset into a train set of 70% and a test set of 30%.

Table 5.1 Conversion of Labels to one-hot encodings for the public dataset

Number	Label	One-hot encoding
0	BROWSING	[1 0 0 0 0 0 0]
1	CHAT	[0 1 0 0 0 0 0]
2	FT	[0 0 1 0 0 0 0]
3	EMAIL	[0 0 0 1 0 0 0]
4	P2P	[0 0 0 0 1 0 0]
5	STREAMING	[0 0 0 0 0 1 0]
6	VOIP	[0 0 0 0 0 0 1]

i. **Building the ANN model using Keras:** In this stage, we use the preprocessed public dataset to build the ANN classification model. First, we initialize the training model as a classifier using Keras sequential module library since we are trying to build a classification system. Next, we introduced the dense function to build fully connected layers we will create in the model. And finally, we called the dropout function to prevent overfitting of our model. Our first input layer is made up of 512 neurons, input dimension of 23 independent variables, ReLu activation function, a dropout rate of 0.35 which is 0.1 more than the standard rate of 0.25, and a uniform kernel initializer. We have 4 hidden layers in total and a SoftMax layer. The output layer contains 7 multi-classification categories and uses the SoftMax activation function because of the multi-classification. After which we compiled the ANN model using Adam stochastic gradient descent, the loss function is generated using categorical_crossentropy and accuracy metrics. Next, we

trained the ANN classification model using the train set with a batch size of 125 and epochs of 500.

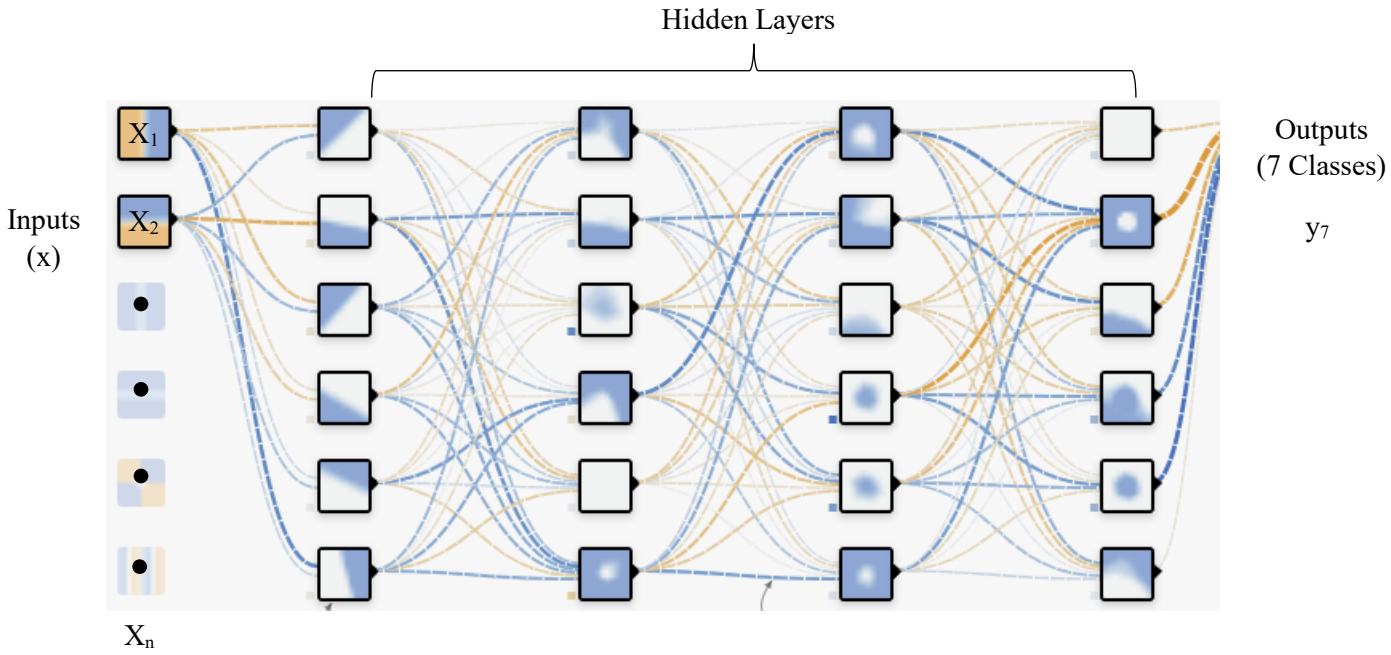


Figure 5. 1 *ANN Architecture (source: TensorFlow, 2020) for the UNB public dataset*

j. Efficiency analysis: Here, we tested the classification accuracy of the ANN classification model on the test set split. Table 5.2 below shows the results obtained in each class while table 5.3 shows the confusion matrix obtained for each class.

Table 5.2 ANN Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	87%	94%	90%
1	CHAT	77%	69%	73%
2	FT	84%	84%	84%
3	EMAIL	65%	64%	64%
4	P2P	90%	86%	88%
5	STREAMING	86%	76%	81%
6	VOIP	99%	99%	99%

Table 5.3 ANN Experiment Confusion Matrix (Public Dataset)

True Label \ Predicted Label	BROWSING	CHAT	FT	EMAIL	P2P	STREAMING	VOIP
BROWSING	713	19	9	7	2	7	3
CHAT	41	179	11	4	18	3	2
FT	24	6	266	9	5	5	2
EMAIL	11	5	4	37	0	0	1
P2P	6	22	10	0	251	2	0
STREAMING	22	0	11	0	4	117	0
VOIP	3	0	5	0	0	2	842

k. **Boosting the ANN classification model using XGBoost:** eXtreme Gradient Boosting (XGBoost) is a library of gradient boosting algorithms that are used to optimize models to solve classification or regression problems (Manish, 2018). It takes the output of the base model (which is the ANN model output) as its input to train the XGB classifier which has already been initialized to improve the accuracy of the predictions on the test sets thereby forcing out the best result of the classification model. We first imported the XGBoost library. Then we created an XGB classifier to solve our classification problem, setting the base estimator as ANN model with 1000 number of estimators, an objective of multi: softmax to handle our multi-class labels, a learning rate of 0.25, maximum depth of 12 and the number of classes of 7 we are trying to predict. Next, we fitted the train set into the XGB classifier which we created for predictions and tested its accuracy using the test set. The results of the experiment are shown in table 5.4 and 5.5 below.

Table 5.4 ANN + XGBoost Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	95%	98%	97%
1	CHAT	95%	88%	92%
2	FT	92%	93%	93%
3	EMAIL	86%	95%	90%
4	P2P	99%	97%	98%
5	STREAMING	92%	87%	89%
6	VOIP	100%	99%	99%

Table 5.5 ANN + XGBoost Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	748	5	0	0	0	7	0
	CHAT	22	228	2	1	2	2	1
True Label	FT	5	3	296	7	2	3	1
	EMAIL	1	0	2	55	0	0	0
	P2P	1	2	6	1	281	0	0
	STREAMING	11	0	9	0	0	134	0
	VOIP	1	1	6	0	0	0	844

5.1.2 Convolutional Neural Networks (CNN)

CNN is also among the deep learning networks' family that was proposed by (LeCun et. al., 1995). It is a deep learning algorithm that takes input in the form of an *n*-dimensional image (where *n* can be 1, 2 or 3), assigns weights and biases to the input to be able to differentiate them from each other, then outputs the result(s) of the analysis. It consists of various convolutional layers which are the building blocks of CNN. Feature learning happens in the convolution layers (Saha, 2018). The second-deep learning technique we designed and implemented in the classification of encrypted network traffic is the one-dimensional

Convolutional Neural Network (1D-CNN). Also, as one of the deep learning models, it applies to process and analyzing big data.

a. **Convolution:** It is the operation that is performed on CNN by multiplying sets of weight with the input. Our input took the 3-dimensional shape, so, the multiplication is executed between an array of input data and a 3-dimensional array of weights which is called the filter. A repetition of the filter applied to the input results in a map of activations will result in a feature mapping. Then the value in the feature is passed to the ReLu activation function.
b. **SoftMax:** SoftMax function is a function used to determine the probability of multiple classes at once, unlike the sigmoid function which is limited to 0 or 1 (binary).

Figure 5.2 *1D-CNN Architecture for the public dataset*

c. **Building the 1D-CNN classification model using Keras:** To build the 1D-CNN classification model, we use the already preprocessed public dataset. First, we initialize the training 1D-CNN model as a classifier using Keras sequential module library because of the classification system we want to build. Next, we introduced the convolution function that will build the convolutional layers. Here, we use the convolution 1D. We then imported the existing MaxPooling function from the Keras API to down-sample the input representation, reducing its dimensions and making assumptions about the features contained in the sub-regions binned. Next was the dense function to build fully connected layers we will create in the model. Next, we called the dropout function to prevent overfitting of the CNN model. And finally, we called the flattening function to convert the output of convolution layers into a 1-dimensional array to create a single feature vector which is then inputted to the final layer called the fully connected layer. Our first convolution layer is made up 512 filters, kernel size of 3 which specifies the length of the 1D convolution window, the same padding which means no padding that the output will have the same length as the original input, input of 23 independent variables with a step size of 1, ReLu activation function, strides of 1 which is the default value, a dropout rate of 0.35 which is close to the standard rate of 0.25 and a Max Pooling with a pool size of 2 to reduce the size of the feature map. We have 6 convolution layers in total, a flatten layer, 1 fully connected layer, and a SoftMax layer. The output layer contains 7 multi-classification categories and uses the SoftMax activation function because of the multi-classification. We then compiled the CNN model using Adam stochastic gradient descent, the loss function is generated using categorical_crossentropy and accuracy metrics. Next, we converted the train and test sets to a 3-dimensional array which will be accepted by the 1D-CNN model. Then, we trained the CNN model using the train set with a batch size of 125 and 500 epochs.

d. Efficiency analysis: Here, we tested the classification accuracy of the 1D-CNN model on the test set split. Table 5.6 and 5.7 below shows the results obtained.

Table 5.6 1D-CNN Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	95%	97%	96%
1	CHAT	95%	88%	91%
2	FT	91%	92%	92%
3	EMAIL	86%	95%	91%
4	P2P	98%	95%	99%
5	STREAMING	94%	84%	89%
6	VOIP	99%	99%	100%

Table 5.7 1D-CNN Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
True Label	BROWSING	750	6	1	0	0	3	0
	CHAT	24	228	3	0	0	1	2
	FT	6	2	298	4	2	3	2
	EMAIL	0	0	3	54	0	0	1
	P2P	1	3	6	0	280	1	0
	STREAMING	13	0	10	0	0	131	0
	VOIP	3	2	5	0	0	0	842

e. Boosting the 1D-CNN model using XGBoost: eXtreme Gradient Boosting (XGBoost) is a library of gradient boosting algorithms that are used to optimize models to solve classification or regression problems (Manish, 2018). It takes the output of the 1D-CNN classification model as its input to train the XGB classifier to improve the accuracy of the predictions on the test set and force out the best accuracy of the model. We first imported the XGBoost library. Then we created an XGB classifier to solve our classification problem, setting the base estimator as 1D-CNN model with 800 number of estimators, an objective of multi: softmax to handle our multi-class labels, a learning rate of 0.1,

maximum depth of 5 and the number of classes of 7 which is the target class. Next, we fitted the train set into the XGB classifier we created for predictions. The results of the 1D-CNN + XGBoost experiment are shown in table 5.8 and 5.9 below.

Table 5.8 1D-CNN + XGBoost Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	94%	98%	96%
1	CHAT	95%	87%	91%
2	FT	91%	93%	92%
3	EMAIL	87%	95%	91%
4	P2P	99%	96%	98%
5	STREAMING	92%	84%	88%
6	VOIP	100%	99%	99%

Table 5.9 1D-CNN + XGBoost Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	748	6	1	0	0	5	0
	CHAT	27	224	2	1	1	2	1
True Label	FT	5	3	296	7	2	4	0
	EMAIL	0	0	3	55	0	0	0
	P2P	1	3	6	0	280	1	0
	STREAMING	13	0	11	0	0	130	0
	VOIP	1	1	8	0	0	0	842

5.1.3 Recurrent Neural Network using Long Short-Term Memory (LSTM) Concept

RNN-LSTM is the third deep neural network we used in designing the classification model for classifying the encrypted network traffic. It was proposed by John Hopfield in 1982. It is used for performing operations on sequences such as text, image, voice sequences to mention just a few and enables the neural networks to learn the patterns (detecting the actions in the sequence of an image or the input) associated with the input (Oshana and Kraeling, 2019). The second hidden layer of an RNN makes use of the first hidden layer's

output as its input to let or permit the neural network to acquire the concept of the temporal memory and sharing the weights. We built the RNN classification model on the Long Short-Term Memory (LSTM) concept.

a. **Building the RNN classification model on the LSTM concept using Keras:** To build the RNN-LSTM classification model, we used the already preprocessed public dataset. First, we imported the necessary libraries, then we initialized the RNN-LSTM training model as a classifier using Keras sequential module library since it is a classification system that we are building. We then introduced the LSTM function to build the LSTM hidden neural network layers. Next was the dropout function that will prevent overfitting of the RNN-LSTM model and the dense function to build fully connected layers we will create later in the model. The first LSTM layer is made up of 512 filters, a ReLu activation function, a 3-dimensional input shape, return sequences set to True, and a dropout of 0.25. We have a total of 4 LSTM layers, 1 fully connected layer, a SoftMax layer, and the output which is the 7 categories of network traffic. We compiled the RNN-LSTM classification model using Adam stochastic gradient descent, the loss function is generated using categorical_crossentropy and accuracy metrics because of the classification problem. Then, we trained the RNN-LSTM model using the train set with a batch size of 200 and 300 epochs.
b. **Efficiency analysis:** After performing a series of experiments on the RNN-LSTM model during training, we tested the classification accuracy of the LSTM based RNN classification model using the test set split. Table 5.10 and 5.11 below show the results obtained.

Table 5.10 RNN-LSTM Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	86%	87%	86%
1	CHAT	69%	75%	72%
2	FT	74%	70%	72%
3	EMAIL	57%	67%	62%
4	P2P	97%	91%	94%
5	STREAMING	62%	58%	60%
6	VOIP	99%	99%	99%

Table 5.11 RNN-LSTM Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	654	52	10	18	0	19	1
	CHAT	45	187	6	1	6	3	0
True Label	FT	33	10	206	13	3	22	6
	EMAIL	7	1	17	50	0	0	0
	P2P	0	19	9	1	283	0	0
	STREAMING	22	0	27	3	1	75	1
	VOIP	1	1	4	1	0	1	871

c. **Boosting the RNN model using XGBoost:** After building, training, and testing the RNN-LSTM classification model using the train and test sets, we implemented the eXtreme Gradient Boosting (XGBoost) on the RNN-LSTM model to boost its overall performance. The XGBoost classifier took the output of the RNN-LSTM classification model as its input (i.e. base estimator) to train the XGB classifier with the number of estimators set to 100, an objective of multi: softmax, a learning rate of 0.1 and 7 number of classes which is the target class. Next, we fitted the train set into the XGB classifier we created for predictions and then tested its accuracy using the test set. The results of the experiment obtained are shown in table 5.12 and 5.13 below.

Table 5.12 RNN-LSTM + XGBoost Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	86%	87%	86%
1	CHAT	69%	75%	72%
2	FT	74%	70%	72%
3	EMAIL	57%	67%	62%
4	P2P	97%	91%	94%
5	STREAMING	62%	58%	60%
6	VOIP	99%	99%	99%

Table 5.13 RNN-LSTM + XGBoost Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	727	14	3	2	1	7	0
	CHAT	40	200	3	1	2	2	0
True Label	FT	11	6	525	9	3	10	2
	EMAIL	4	3	6	62	0	0	0
	P2P	1	1	2	0	307	0	1
	STREAMING	19	3	12	0	0	95	0
	VOIP	6	2	4	0	0	0	867

5.1.4 Capsule Neural Network (CapsNet)

CapsNet is the fourth deep neural network we used in designing the classification model for the classification of the encrypted network traffic. It was proposed by Hinton and his colleagues in the year 2011 (Hinton et. al., 2011 and Hinton et. al., 2018). It is a new architecture in the world of neural networks and is regarded as an advanced approach of the Convolutional Neural Network (CNN). The Capsules which constitute of the CapsNet architecture are made up of neurons (a group of neurons) that takes vectors as inputs and outputs vectors as opposed to CNN that accepts and outputs scalar values. This allows the capsule to learn the features of the dataset given to it as well as its deformations and viewing conditions. The capsule

takes the output of the CNN as its input. The output of the CNN comprises features that the capsule uses in learning the patterns associated with the dataset while the CapsNet output comprises the probability that the features from the CNN output encoded by that capsule are present and the instantiation parameters (i.e. the vector values). The architecture of the CapsNet comprises three (3) main parts namely, (a) Primary capsules (Convolution, Reshape and Squash) (b) Higher layer capsules (Routing by agreement) and (c) Loss calculation (margin loss and Reconstruction Loss).

a. **Primary Capsules:** The primary capsule is the first layer of the CapsNet. This is where the inversion process takes place.
b. **Reshape Function:** This is used to reshape the output features into vectors of any desired dimension related to the input dimension of the dataset.
c. **Squash Function:** This function makes sure that the length of the output which are the vectors from the capsule is not greater than 1 but lies between 0 and 1. This is so because the length of each vector represents the probability of whether or not the feature or object exists in the dataset and its location. It retains the positional information that is in higher dimensions of the vector.
d. **Building the CapsNet classification model using Keras:** At this stage, we use the already preprocessed public dataset to build the CapsNet classification model. First, we imported the necessary libraries from Keras that are required to build the CapsNet model. We defined the squash, margin loss, and the SoftMax functions for the CapsNet model. Then, we created a class called the capsule, defined the primary capsule and higher layer capsules functions of the model in the capsule class, after which we defined a function that will compute the output of the capsules. After building the architecture of the CapsNet, we need to feed it the public dataset for classification. For us to do that, we specified our input shape to be in a 3-dimensional array, we specified the capsule filters to be 256, kernel size of 3, and ReLu activation. We then compiled the CapsNet classification model

using the margin loss function, Adam optimizer, and accuracy for metric measurement. Then, we trained the CapsNet model using the train set with a batch size of 65 and 100 epochs.

e. **Efficiency analysis:** After performing a series of experiments on the CapsNet classification model during training, we tested the classification accuracy of the CapsNet model using the test set split. Table 5.14 and 5.15 below shows the results obtained.

Table 5.14 CapsNet Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	88%	92%	90%
1	CHAT	67%	70%	69%
2	FT	81%	77%	79%
3	EMAIL	54%	67%	60%
4	P2P	86%	85%	86%
5	STREAMING	84%	70%	77%
6	VOIP	100%	99%	99%

Table 5.15 CapsNet Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	697	33	9	8	4	8	1
	CHAT	36	181	8	4	27	2	0
True Label	FT	21	17	244	19	6	9	1
	EMAIL	6	6	7	39	0	0	0
	P2P	4	31	8	0	247	1	0
	STREAMING	22	0	18	2	2	108	2
	VOIP	3	2	6	0	0	0	841

f. **Boosting the CapsNet classification model using XGBoost:** After building, training, and testing the CapsNet model using the train and test sets, we implemented the eXtreme Gradient Boosting (XGBoost) on the model to boost its overall performance. The XGBoost classifier took the output of the CapsNet classification model as its input which is specified as the base estimator to train the

XGB classifier with the number of estimators set to 1000, an objective of multi: softmax, a learning rate of 0.075, maximum depth of 5 and 7 number of classes which is the target class. Next, we fitted the trainsets into the XGB classifier we created for predictions. The results of the experiment obtained are shown in table 5.16 and 5.17 below.

Table 5.16 CapsNet + XGBoost Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	94%	98%	96%
1	CHAT	95%	87%	91%
2	FT	91%	94%	93%
3	EMAIL	87%	95%	91%
4	P2P	99%	96%	97%
5	STREAMING	92%	85%	88%
6	VOIP	100%	99%	99%

Table 5.17 CapsNet + XGBoost Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	748	4	0	0	1	7	0
	CHAT	27	224	2	1	1	1	2
True Label	FT	5	3	298	7	1	3	0
	EMAIL	0	0	3	55	0	0	0
	P2P	1	4	6	0	279	1	0
	STREAMING	13	0	10	0	0	131	0
	VOIP	1	1	7	0	0	0	843

5.1.5 Ensemble Learning Technique

An ensemble learning technique is a process that uses multiple machine learning models or techniques such as classifiers structured together to solve a particular problem and improve the system's accuracy which can as well make the performance better (Paul, 2018). There are different techniques of ensemble learning

methods such as averaging, weighted average, stacking to mention just a few, but the commonly used methods are the boosting, bagging, stacking, and voting based ensemble learning (Tanner, 2019).

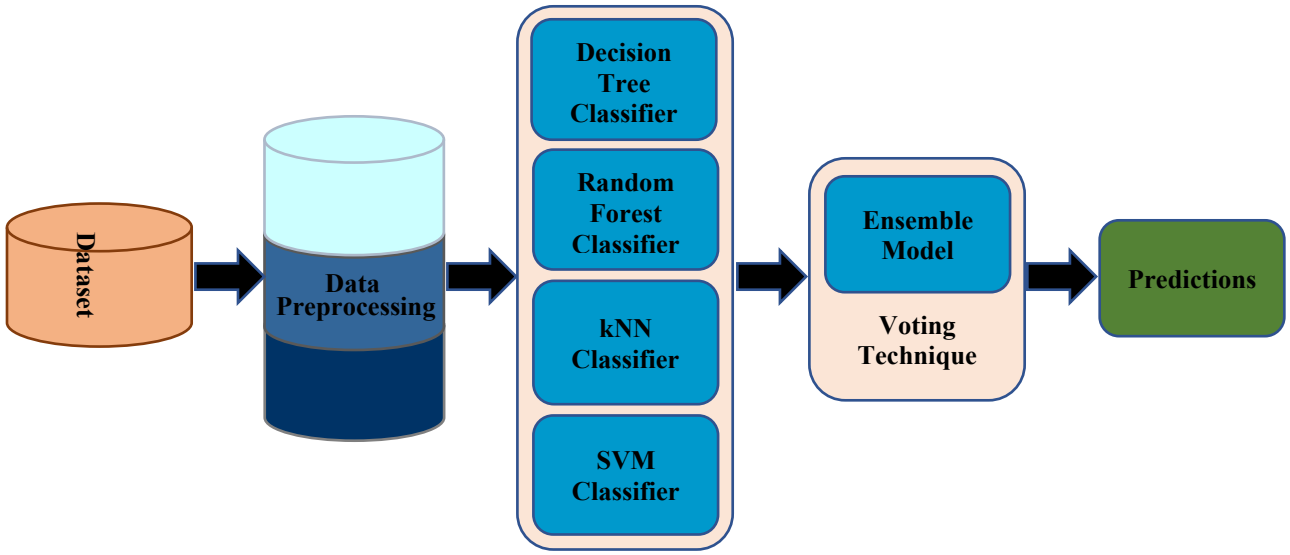


Figure 5.3 *Ensemble Structure for the public dataset*

a. **Boosting:** This is a form of sequential learning technique where the model is trained using the training set, and models are constructed subsequently by fitting the residual error values to the initial model. Therefore, the boosting seeks to allocate higher weight to the observations that were estimated poorly by the previous model. The models we used for our boosting technique are the AdaBoost known as the Adaptive Boosting, the Gradient Boosting Machine (GBM), and the eXtreme Gradient Boosting (XGBoost).

b. **Bagging:** This is also known as Bootstrap Aggregation. The bootstrap forms the foundation in which the bagging technique is built upon. During sampling in the bootstrapping process, we select random '*n*' observations out of the total '*n*' observations. These *n* observations are called bootstrapped samples generated from the train set and will be tested using the test set. The figure below summarises the idea of bagging.

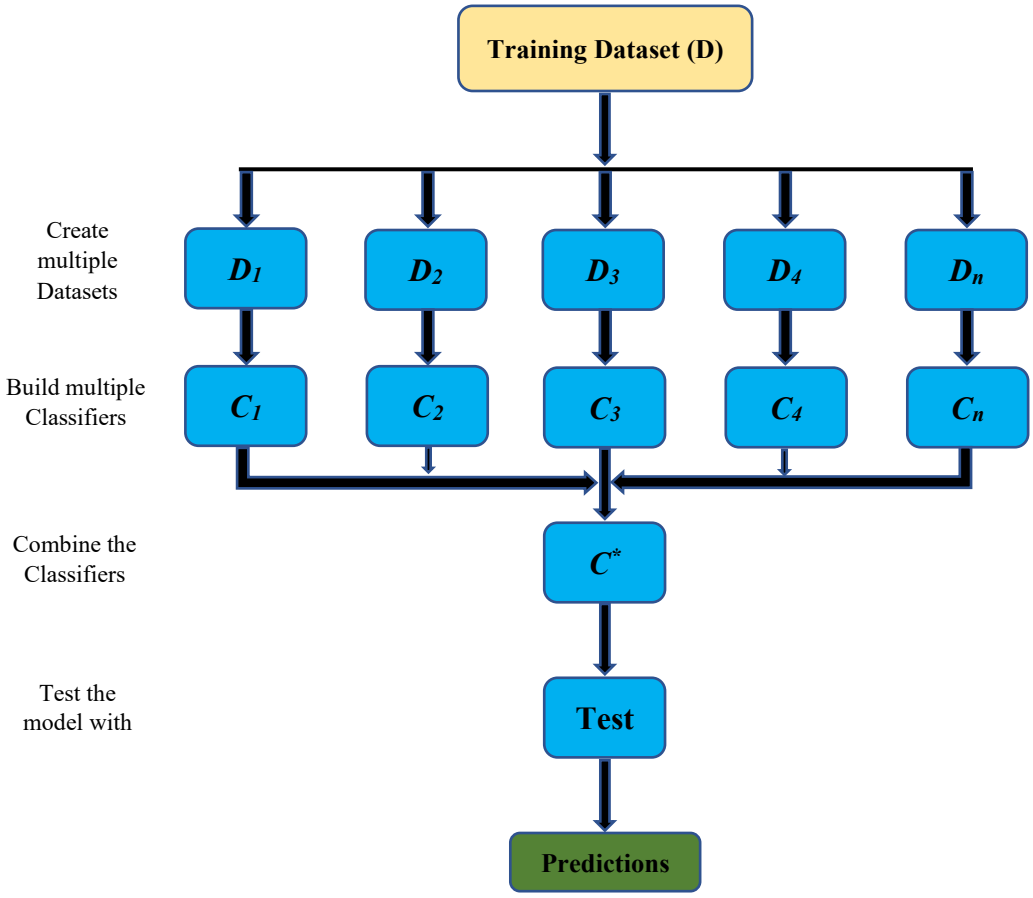


Figure 5.4 *Illustration of Bagging Technique for the public dataset*

c. **Voting:** This is an ensemble learning technique where the predictions from multiple models created while designing the ensemble model are aggregated and averaged. We designed the voting ensemble technique by creating four different classification models (the Decision Tree, Random Forest, k-Nearest Neighbors, and Support Vector classifiers) and then trained the models using the same train set. Next, we imported the voting classifier from the sklearn ensemble and set the voting as hard and then made predictions on the ensemble classification model using the test set.

d. **Building the Ensemble model:** Here, we use the already preprocessed public dataset to build the ensemble model. First, we import the Decision Tree, Random Forest, k-Nearest Neighbors, and Support Vector Machine classifiers from sklearn in python. Next, we initialize the training models as a classifier since we are building a classification model. We then trained the models by fitting the train set into the model. Next, we tested the accuracy of the models using the test set. We imported the voting classifier from the sklearn ensemble and set the voting as hard and then made predictions on the ensemble model using the test set.

e. **Efficiency analysis:** Here, we tested the classification accuracy of the ensemble model using the test set split. Table 5.18 and 5.19 below shows the results obtained.

Table 5.18 Ensemble Model Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	83%	98%	90%
1	CHAT	89%	73%	80%
2	FT	90%	80%	85%
3	EMAIL	84%	83%	83%
4	P2P	96%	94%	95%
5	STREAMING	94%	70%	80%
6	VOIP	100%	99%	99%

Table 5.19 Ensemble Model Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	745	8	3	0	0	3	1
	CHAT	61	188	3	0	6	0	0
True Label	FT	33	10	253	9	5	4	3
	EMAIL	8	0	2	48	0	0	0
	P2P	8	5	4	0	274	0	0
	STREAMING	35	0	11	0	0	108	0
	VOIP	3	1	5	0	0	0	843

f. **Boosting the Ensemble model using XGBoost:** After building, training, and testing the ensemble model using the train and test sets, we applied the eXtreme Gradient Boosting (XGBoost) on the Ensemble model to boost its overall performance. The XGBoost classifier took the output of the ensemble classification model as its input (i.e. base estimator) to train the XGB classifier with the number of estimators set to 500, an objective of multi: softmax, a learning rate of 0.1 and 7 number of classes which is the target class. Next, we fitted the train set into the XGB classifier we created

for predictions and tested its accuracy using the test set. The results of the experiment obtained are shown in table 5.20 and 5.21 below.

Table 5.20 Ensemble Model + XGBoost Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	93%	98%	95%
1	CHAT	92%	84%	88%
2	FT	91%	91%	91%
3	EMAIL	84%	93%	89%
4	P2P	99%	96%	97%
5	STREAMING	89%	86%	87%
6	VOIP	100%	99%	99%

Table 5.21 Ensemble Model + XGBoost Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
True Label	BROWSING	743	5	2	0	0	10	0
	CHAT	34	217	4	1	1	1	0
	FT	7	6	289	9	2	4	0
	EMAIL	1	0	2	54	0	1	0
	P2P	2	5	4	0	279	1	0
	STREAMING	13	0	9	0	0	132	0
	VOIP	2	2	6	0	0	0	842

5.1.6 Machine Learning- Decision Tree (DT) Technique

A decision tree is a supervised machine learning that is used for classification and regression tasks or problems. It is in a tree-like structure where the nodes of the tree represent the independent variables of the dataset being used for analysis, the decision rule is being represented by the branch while the leaf nodes represent the outcome of the analysis (Gupta, 2017).

a. **Building the Decision Tree Model:** We use the already preprocessed public dataset which was done using the preprocessing modules MinMax scaler and label encoder from sklearn to build the decision tree model for the classification of encrypted network traffic. First, we import the Decision Tree

classifier from sklearn in python. Next, we initialize our decision tree model as a classifier since we are building a classification model. We then trained the model by fitting the train set into the DT model. Next, we tested the accuracy of the DT model by making predictions on the model using the test set. The results of the experiment are shown in table 5.22 and 5.23 below.

Table 5.22 Decision Tree Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	89%	89%	89%
1	CHAT	76%	76%	76%
2	FT	77%	88%	82%
3	EMAIL	79%	80%	79%
4	P2P	97%	95%	96%
5	STREAMING	88%	68%	77%
6	VOIP	98%	98%	98%

Table 5.23 Decision Tree Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
True Label	BROWSING	664	38	27	5	0	9	7
	CHAT	34	187	13	3	5	0	3
	FT	15	10	279	5	2	4	2
	EMAIL	5	3	6	55	0	0	0
	P2P	3	4	9	0	298	0	1
	STREAMING	21	2	20	2	0	102	2
	VOIP	1	3	8	0	2	1	830

b. **Boosting the Decision Tree model using XGBoost:** After building, training, and testing the DT model using the train and test sets, we applied the eXtreme Gradient Boosting (XGBoost) on the decision tree model to boost its overall performance. The XGBoost classifier took the output of the decision tree classification model as its input (i.e. base estimator) to train the XGB classifier with the number of estimators set to 500, an objective of multi: softmax, a learning rate of 0.25 and 7

number of classes which is our target class. Next, we fitted the train set into the XGB classifier we created for predictions and tested the model's accuracy using the test set. The results of the experiment obtained are shown in table 5.24 and 5.25 below.

Table 5.24 Decision Tree + XGBoost Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	93%	98%	95%
1	CHAT	91%	86%	88%
2	FT	89%	93%	91%
3	EMAIL	93%	93%	93%
4	P2P	100%	96%	98%
5	STREAMING	91%	77%	83%
6	VOIP	100%	99%	100%

Table 5.25 Decision Tree + XGBoost Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	734	10	1	0	0	5	0
	CHAT	29	210	3	2	0	1	0
True Label	FT	12	6	294	2	0	3	0
	EMAIL	0	0	5	64	0	0	0
	P2P	2	4	4	0	302	3	0
	STREAMING	15	1	17	1	0	115	0
	VOIP	1	0	6	0	0	0	838

c. **Boosting the Decision Tree model using AdaBoost:** Here, we preprocess the public dataset using the MinMax scaler and label encoder preprocessing module to convert the records into a machine-readable format to enable us to build the AdaBoost-based classification model using the decision tree classification model as the base estimator. First, we import the AdaBoost classifier from sklearn in python. Next, we initialize the AdaBoost training model as a classifier since we are building a classification model by setting the base estimator to be the decision tree classification model, 100 number of estimators with a learning rate of 0.1. We then trained the AdaBoost model by fitting the

train set into it. Next, we tested the accuracy of the AdaBoost model by making predictions on the AdaBoost model using the test set. The results of the experiment are shown in table 5.26 and 5.27 below.

Table 5.26 Decision Tree + AdaBoost Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	90%	98%	94%
1	CHAT	89%	82%	85%
2	FT	86%	91%	89%
3	EMAIL	94%	86%	89%
4	P2P	100%	95%	97%
5	STREAMING	96%	72%	83%
6	VOIP	100%	99%	99%

Table 5.27 Decision Tree + AdaBoost Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	735	11	1	0	0	3	0
	CHAT	36	200	7	0	0	0	2
True Label	FT	18	4	289	3	1	1	1
	EMAIL	2	0	7	59	0	0	1
	P2P	2	7	8	0	298	0	0
	STREAMING	20	3	17	1	0	108	0
	VOIP	3	0	7	0	0	0	835

d. **Building the Gradient Boost model:** Here, we preprocess the public dataset using the MinMax scaler and label encoder preprocessing module to convert the records into a machine-readable format to enable us to build the Gradient Boost classification model. First, we imported the Gradient Boosting Classifier and set the number of estimators to 1000, a learning rate of 0.25, maximum features of 23, maximum depth of 20, and a random state value of 0. We then fitted the train set into the Gradient Boost model to train the model. We tested the accuracy of the model by making

predictions on the Gradient Boost-based model using the test set. The results of the experiment are shown in table 5.28 and 5.29 below.

Table 5.28Gradient Boost Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	90%	96%	93%
1	CHAT	86%	81%	83%
2	FT	85%	88%	86%
3	EMAIL	88%	87%	88%
4	P2P	98%	95%	97%
5	STREAMING	91%	73%	81%
6	VOIP	100%	99%	99%

Table 5.29 Gradient Boost Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
True Label	BROWSING	719	18	6	1	0	6	0
	CHAT	38	198	5	0	2	1	1
	FT	20	8	278	6	3	1	1
	EMAIL	1	1	7	60	0	0	0
	P2P	3	4	7	0	299	2	0
	STREAMING	20	1	18	1	0	109	0
	VOIP	2	0	6	0	0	1	836

e. **Building the Decision Tree model using the Bagging Technique:** Here, we use the Decision Tree classification model as the base estimator of the bagging technique model. First, we imported the Bagging Classifier and set the base estimator as the decision tree model, a random state of 8, and 23 number of estimators. We then fitted the train set into the Bagging technique model to train the model. We tested its accuracy by making predictions on the bagging model using the test set. The results of the experiment are shown in table 5.30 and 5.31 below.

Table 5.30 Decision Tree + Bagging Technique Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	90%	95%	93%
1	CHAT	86%	82%	84%
2	FT	82%	89%	86%
3	EMAIL	94%	88%	91%
4	P2P	99%	95%	97%
5	STREAMING	93%	68%	78%
6	VOIP	99%	99%	99%

Table 5.31 Decision Tree + Bagging Technique Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
True Label	BROWSING	714	16	12	1	0	7	0
	CHAT	33	202	7	0	2	0	1
	FT	17	10	283	2	1	1	3
	EMAIL	1	0	6	61	0	0	1
	P2P	1	5	7	0	300	0	2
	STREAMING	23	2	21	1	0	101	1
	VOIP	3	1	8	0	0	0	833

5.1.7 Random Forest (RF) Technique

Random Forest is a supervised machine learning algorithm that can solve any classification or regression tasks. It randomly generates sets of large numbers of decision trees starting its construction at the root node which then creates a forest that is in the form ensemble (Obasi and Shafiq, 2019).

a. **Building The Random Forest Model:** To build the random forest classification model, we used the already preprocessed public dataset which was done using the MinMax scaler and label encoder preprocessing module from sklearn preprocessing in python to build the RF classification model for the classification of encrypted network traffic. The first thing we did is to import the Random Forest classifier from the sklearn ensemble in python. Then we initialize the RF model as a classifier for the classification of the encrypted network traffic. Next, we specified the number of estimators as

400 before training the RF model by fitting the train set into the model. For us to determine the performance of the model based on its accuracy, we tested the RF model by making predictions on the Random Forest model using the test set. The results of the experiment are shown in table 5.32 and 5.33 below.

Table 5.32 Random Forest Technique Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	92%	97%	94%
1	CHAT	90%	84%	87%
2	FT	86%	89%	88%
3	EMAIL	88%	89%	89%
4	P2P	98%	98%	98%
5	STREAMING	85%	71%	78%
6	VOIP	100%	99%	99%

Table 5.33 Random Forest Technique Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	728	11	7	1	0	7	0
	CHAT	29	209	3	1	4	2	0
True Label	FT	10	8	261	7	0	6	1
	EMAIL	3	0	5	67	0	0	0
	P2P	0	3	3	0	305	0	1
	STREAMING	16	1	19	0	1	92	0
	VOIP	2	0	5	0	0	1	871

b. **Boosting Random Forest model using XGBoost:** After building, training, and testing the RF model using the train and test sets, we applied the eXtreme Gradient Boosting (XGBoost) on the RF model to boost its overall performance. The XGBoost classifier took the RF classification model as its input (i.e. base estimator) to train the XGBoost with the number of estimators set to 500, an objective of multi: softmax, a learning rate of 1 and 7 number of classes which is the target class.

Next, we performed predictions on the XGB classifier model we created using the test set. The results of the experiment obtained are shown in table 5.34 and 5.35 below.

Table 5.34 Random Forest + XGBoost Experiment Result for each category (Public Dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	93%	97%	95%
1	CHAT	94%	86%	92%
2	FT	88%	91%	90%
3	EMAIL	93%	91%	92%
4	P2P	99%	99%	99%
5	STREAMING	87%	77%	81%
6	VOIP	100%	99%	99%

Table 5.35 Random Forest + XGBoost Experiment Confusion Matrix (Public Dataset)

		Predicted Label						
	BROWSING	734	10	1	0	0	5	0
	CHAT	29	210	3	2	0	1	0
True Label	FT	12	6	294	2	0	3	0
	EMAIL	0	0	5	64	0	0	0
	P2P	2	4	4	0	302	3	0
	STREAMING	15	1	17	1	0	115	0
	VOIP	1	0	6	0	0	0	838

5.2 Experiments on the Solana Networks dataset

5.2.1 Artificial Neural Network (ANN)

This is the first technique we used in designing the classification model for the classification of encrypted network traffic using the Solana Networks dataset. It is applicable to process big data.

a. **Data preparation:** The dataset we used in designing the ANN classification model for the second batch of the experiment is from Solana Networks. The dataset includes network traffic consisting of 83 independent variables after extracting the statistical features from the pcap files. We used a total of 27 independent variables which became the input independent variables with 95711 flows

and 7 traffic categories as the target class. Figure 5.5 below depicts the traffic categories considered for the Solana Networks dataset and the counts of their instances.

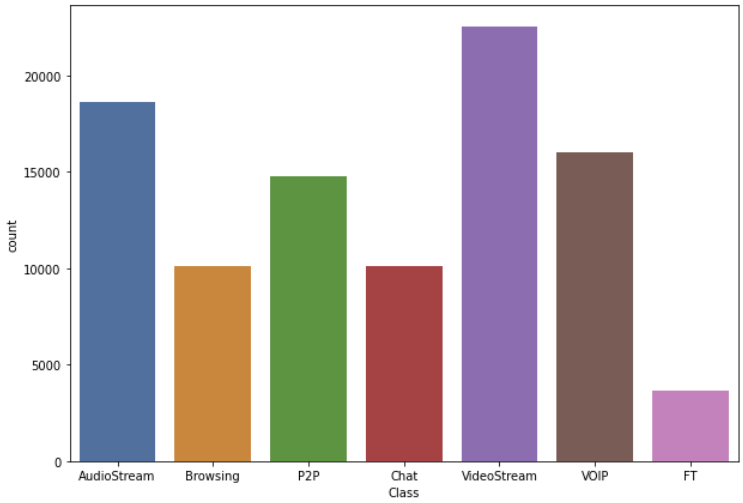


Figure 5.5 *Traffic categories considered for the Solana Networks dataset*

b. **Data preprocessing:** Here, before preprocessing the dataset, we balanced the dataset using the SMOTETomek method from the imblearn.combine library (Harliman and Uchida, 2018). Then, we used the scikit-learn library prebuilt functionality used for preprocessing such as label encoder, standard scaler, and MinMax scaler functions. The preprocessing is performed on the features, which is called feature scaling. For the ANN model, we used the label encoder to convert the time series data into a format the machine can read, the standard scaler is used in transforming the records into a scaled form and the One Hot encoder is used to encode the target classes to make the classification flexible during analysis. Then we split the dataset into a train and test sets of 70% and 30% respectively.

Table 5.36 Conversion of Labels to one-hot encodings for the Solana Networks Dataset

Number	Label	One-hot encoding
0	AudioStream	[1 0 0 0 0 0 0]
1	Browsing	[0 1 0 0 0 0 0]
2	Chat	[0 0 1 0 0 0 0]
3	FT	[0 0 0 1 0 0 0]
4	P2P	[0 0 0 0 1 0 0]
5	VOIP	[0 0 0 0 0 1 0]
6	VideoStream	[0 0 0 0 0 0 1]

c. **Building the ANN model using Keras:** In this stage, we use the preprocessed Solana Networks dataset to build the ANN classification model. To build the model, we first initialized the training model as a classifier using Keras sequential module. Next, we introduced the dense function to build fully connected layers we will create in the model. And finally, we called the dropout function to prevent overfitting of our model. The first input layer is made up of 512 neurons, the input dimension of 27 independent variables, the ReLu activation function, a dropout rate of 0.35, and a uniform kernel initializer. We have 4 hidden layers in total and a SoftMax layer. The output layer contains 7 multi-classification categories and uses the SoftMax activation function because of the multi-classification. After which we compiled the ANN model using Adam stochastic gradient descent as the optimizer, the loss function is generated using categorical_crossentropy and accuracy metrics. Next, we trained the ANN classification model using the train set with a batch size of 400 and epochs of 500.

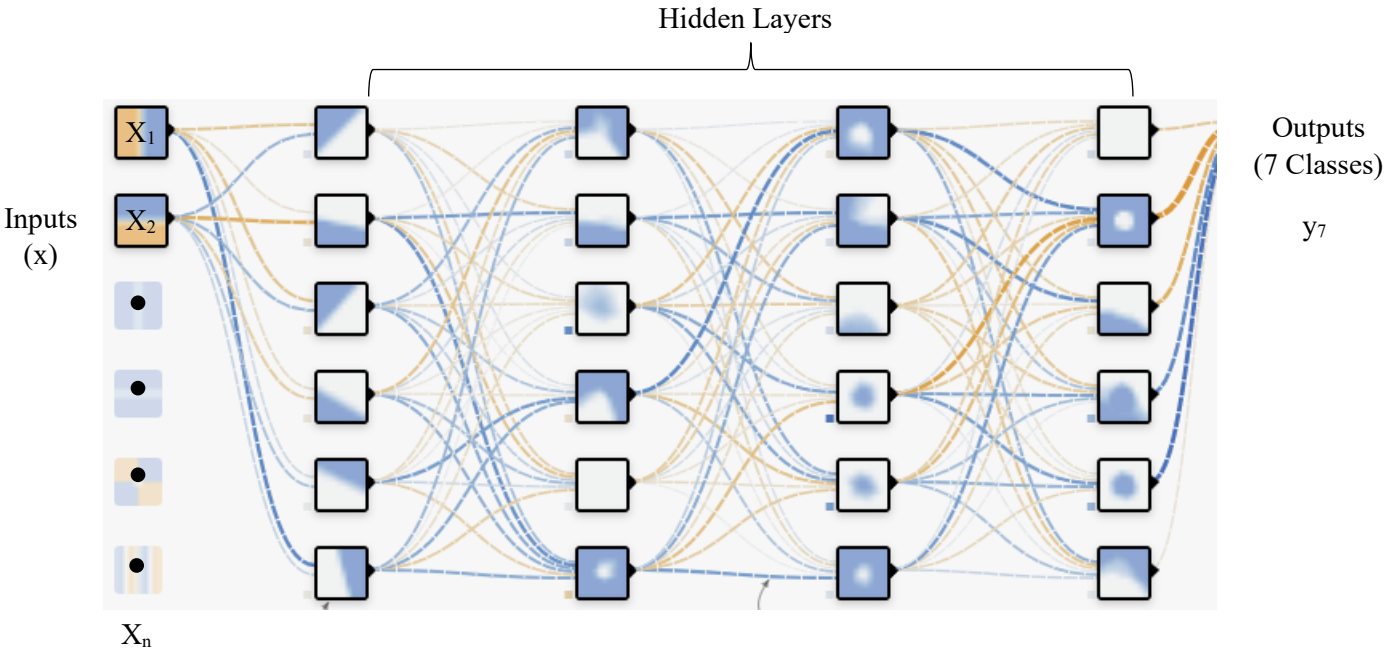


Figure 5.6 *ANN Architecture (source: TensorFlow, 2020) for Solana Networks Dataset*

d. Efficiency analysis: After the design of the model, we tested the classification accuracy of the ANN model on the test set split. Table 5.37 and Table 5.38 below shows the results obtained.

Table 5.37 ANN Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	74%	72%	73%
1	Browsing	85%	69%	76%
2	Chat	65%	95%	77%
3	FT	93%	95%	94%
4	P2P	98%	93%	95%
5	VOIP	89%	91%	90%
6	VideoStream	95%	73%	83%

Table 5.38 ANN Experiment Confusion Matrix (Solana Networks Dataset)

True Label \ Predicted Label	AudioStream	Browsing	Chat	FT	P2P	VOIP	VideoStream
AudioStream	4637	268	1327	51	5	107	47
Browsing	799	4498	681	162	18	294	65
Chat	65	34	6259	7	2	223	6
FT	82	48	82	6367	69	26	5
P2P	82	78	85	127	6112	19	72
VOIP	157	88	208	89	15	6025	41
VideoStream	455	260	934	48	8	60	4748

e. **Boosting the ANN classification model using XGBoost:** The XGBoost takes the output of the base model (which is the ANN model output) as its input to train the XGB classifier which has already been initialized to improve the accuracy of the predictions on the test set thereby forcing out the best accuracy result of the model. We first imported the XGBoost library. Then we created an XGB classifier to solve the classification problem, setting the base estimator as ANN classification model with 450 number of estimators, an objective of multi: softmax to handle the multi-class labels, a learning rate of 0.25, maximum depth of 6 and the number of classes of 7 we are trying to predict. Next, we fitted the train set into the XGB classifier which we created for predictions and tested its accuracy using the test set. The results of the experiment are shown in table 5.39 and 5.40 below.

Table 5.39 ANN + XGBoost Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	77%	83%	80%
1	Browsing	85%	85%	85%
2	Chat	88%	95%	91%
3	FT	98%	98%	98%
4	P2P	98%	95%	97%
5	VOIP	96%	94%	95%
6	VideoStream	90%	80%	85%

Table 5.40 ANN + XGBoost Experiment Confusion Matrix (Solana Networks Dataset)

True Label \ Predicted Label	AudioStream	Browsing	Chat	FT	P2P	VOIP	VideoStream
AudioStream	5372	424	327	22	19	41	237
Browsing	540	5532	139	43	32	72	159
Chat	175	28	6236	3	5	102	47
FT	30	46	21	6525	38	9	10
P2P	67	92	13	57	6279	17	50
VOIP	111	104	121	21	11	6199	56
VideoStream	707	292	247	15	13	44	5195

5.2.2 Convolutional Neural Network (CNN)

This is the second-deep learning technique we designed and implemented for the classification of Solana Networks encrypted network traffic. Also, as one of the deep learning models, it applies to process and analyzing big data.

a. **Building the 1D-CNN classification model using Keras:** To build the 1D-CNN model, we used the already preprocessed Solana Networks dataset. First, we initialized the training 1D-CNN model as a classifier using Keras sequential module. Next, we introduced the convolution function that will build the convolutional layers. Here, we use the convolution 1D. We then imported the MaxPooling function from Keras to down-sample the input representation, reducing its dimensionality and allowing for assumptions to be made about the features contained in the sub-regions binned. Next was the dense function to build fully connected layers we will create in our model. Next, we called the dropout function to prevent overfitting of the 1D-CNN classification model. And finally, we called the flattening function to convert the output of convolution layers into a 1-dimensional array to create a single feature vector which is then inputted to the final layer called the fully connected layer. Our first convolution layer is made up 128 filters, kernel size of 3 which specifies the length of the 1D convolution window, the same padding which means no

padding that the output will have the same length as the original input, input of 27 independent variables with a step size of 1, ReLu activation function, strides of 1 which is the default value, a dropout rate of 0.25 which is the standard rate and a MaxPooling with a pool size of 2 to reduce the size of the feature map. We have 6 convolution layers in total, a flatten layer, 1 fully connected layer, and a SoftMax layer. The output layer contains 7 multi-classification categories and uses the SoftMax activation function because of the multi-classification. We then compiled the 1D-CNN classification model using Adam stochastic gradient descent, the loss function is generated using categorical_crossentropy and accuracy metrics. Next, we converted the train and test sets to a 3-dimensional array which will be accepted by the 1D-CNN model. Then, we trained the 1D-CNN model using the train set with a batch size of 800 and 500 epochs.

Figure 5.7 *1D-CNN Model Architecture for Solana Networks Dataset*

b. Efficiency analysis: After training the model, we tested its classification accuracy on the test set split. Table 5.41 and 5.42 below shows the results obtained.

Table 5.41 1D-CNN Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	71%	83%	76%
1	Browsing	83%	76%	79%
2	Chat	84%	92%	88%
3	FT	94%	95%	94%
4	P2P	98%	94%	96%
5	VOIP	90%	93%	91%
6	VideoStream	94%	75%	83%

Table 5.42 1D-CNN Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
	AudioStream	5371	393	451	39	8	56	124
	Browsing	920	4985	179	145	18	181	89
True Label	Chat	162	54	6096	7	7	266	4
	FT	84	54	39	6333	58	107	4
	P2P	100	110	15	111	6174	17	48
	VOIP	147	98	93	80	8	6150	47
	VideoStream	834	338	349	43	12	44	4893

f. Boosting the 1D-CNN classification model using XGBoost: This takes the output of the 1D-CNN classification model as its input to train the XGB classifier to improve the accuracy of the predictions on the test set and force out the best accuracy of the model. We first imported the XGBoost library. Then we created an XGB classifier to solve our classification problem, setting the base estimator as CNN model with 650 number of estimators, an objective of multi: softmax to handle our multi-class labels, a learning rate of 0.75, maximum depth of 4 and the number of classes of 7 we are trying to predict. Next, we fitted the train set into the XGB classifier we created for

predictions. The results of the 1D-CNN + XGBoost experiment are shown in table 5.43 and 5.44 below.

Table 5.43 1D-CNN + XGBoost Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	77%	82%	79%
1	Browsing	85%	85%	85%
2	Chat	88%	94%	91%
3	FT	98%	98%	98%
4	P2P	97%	96%	97%
5	VOIP	95%	94%	94%
6	VideoStream	89%	80%	84%

Table 5.44 1D-CNN + XGBoost Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
True Label	AudioStream	5280	435	330	17	30	61	289
	Browsing	506	5530	143	28	45	90	175
	Chat	168	31	6232	6	9	104	46
	FT	34	50	15	6513	45	13	9
	P2P	62	88	12	50	6288	19	56
	VOIP	109	104	132	16	19	6193	50
	VideoStream	683	290	248	12	21	50	5209

5.2.3 Recurrent Neural Network using Long Short-Term Memory (LSTM) Concept

RNN-LSTM is the third deep neural network we used in designing the classification model for classifying Solana Networks encrypted network traffic data.

a. **Building the RNN classification model on the LSTM concept using Keras:** To build the RNN-LSTM model, we use the already preprocessed public dataset. First, we imported the necessary libraries, then we initialized the RNN-LSTM training model as a classifier using Keras sequential module library. We then introduced the LSTM function to build the LSTM hidden neural network layers. Next was the dropout function that will prevent overfitting of the RNN-LSTM model and

the dense function to build fully connected layers we will create later in the model. The first LSTM layer is made up of 128 filters, a ReLu activation function, a 3-dimensional input shape, return sequences set to True, and a dropout of 0.25. We have a total of 3 LSTM layers, 1 fully connected layer, a SoftMax layer, and the output which is the 7 categories of network traffic. We compiled the RNN-LSTM classification model using Adam stochastic gradient descent, the loss function is generated using categorical_crossentropy and accuracy metrics due to our classification problem. Then, we trained the RNN-LSTM model using the train set with a batch size of 64 and 200 epochs.

b. **Efficiency analysis:** After performing a series of experiments on the RNN-LSTM model during training, we tested the classification accuracy of the LSTM based RNN model using the test set split. Table 5.45 and 5.46 below shows the results obtained.

Table 5.45 RNN-LSTM Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	70%	84%	76%
1	Browsing	84%	74%	79%
2	Chat	84%	93%	88%
3	FT	93%	96%	94%
4	P2P	98%	94%	96%
5	VOIP	90%	92%	91%
6	VideoStream	93%	76%	83%

Table 5.46 RNN-LSTM Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
	AudioStream	5391	342	404	44	4	75	182
	Browsing	965	4811	194	162	18	220	147
True Label	Chat	168	46	6123	7	10	235	7
	FT	93	48	44	6401	77	14	2
	P2P	104	95	19	114	6164	38	41
	VOIP	143	91	136	110	17	6112	14
	VideoStream	812	280	327	47	12	90	4945

c. **Boosting the RNN classification model using XGBoost:** After building, training, and testing the RNN-LSTM model using the train and test sets, we implemented the eXtreme Gradient Boosting (XGBoost) on the RNN-LSTM model to boost its overall performance. The XGBoost classifier took the output of the LSTM classification model as its input (i.e. base estimator) to train the XGB classifier with the number of estimators set to 650, an objective of multi: softmax, a learning rate of 0.5, maximum depth of 6 and 7 number of classes which is the target class. Next, we fitted the train set into the XGB classifier we created for predictions and then tested its accuracy using the test set. The results of the experiment obtained are shown in table 5.47 and 5.48 below.

Table 5.47 RNN-LSTM + XGBoost Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	78%	82%	80%
1	Browsing	85%	85%	85%
2	Chat	88%	95%	91%
3	FT	98%	98%	98%
4	P2P	97%	96%	97%
5	VOIP	95%	93%	94%
6	VideoStream	89%	81%	84%

Table 5.48 RNN-LSTM + XGBoost Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
	AudioStream	5275	407	330	18	27	58	327
	Browsing	483	5571	141	28	34	86	174
True Label	Chat	155	23	6237	3	10	107	61
	FT	30	44	17	6515	50	13	10
	P2P	63	85	12	47	6297	18	53
	VOIP	107	107	132	16	23	6191	47
	VideoStream	642	290	245	16	22	53	5245

5.2.4 Capsule Neural Networks (CapsNet)

CapsNet is the fourth deep neural network we used in designing the classification model for the classification of the Solana Networks encrypted network traffic data.

a. **Building the CapsNet classification model using Keras:** At this stage, we use the already preprocessed Solana Networks dataset to build the CapsNet classification model. First, we imported the necessary libraries from Keras that are required to build the CapsNet model. We defined the squash, margin loss, and the SoftMax functions for the CapsNet model. Then, we created a class called a capsule, defined the primary capsule and higher layer capsules functions of the model in the capsule class, after which we defined a function that will compute the output of the capsules. After building the architecture of the CapsNet, we need to feed it the Solana Networks dataset for classification of the encrypted network traffic. For us to do that, we specified our input shape to be in a 3-dimensional array, we set the capsule filters to be 128, kernel size of 3, and ReLu activation. We then compiled the CapsNet model using the margin loss function, Adam optimizer, and accuracy for metric measurement. Then, we trained the CapsNet classification model using the train set with a batch size of 64 and 150 epochs.

b. **Efficiency analysis:** After performing a series of experiments on the CapsNet classification model during training, we tested the classification accuracy of the CapsNet classification model using the test set split. Table 5.49 and 5.50 below shows the results obtained.

Table 5.49 CapsNet Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	70%	85%	77%
1	Browsing	84%	77%	80%
2	Chat	85%	93%	89%
3	FT	95%	96%	95%
4	P2P	98%	94%	96%

5	VOIP	93%	91%	92%
6	VideoStream	92%	76%	83%

Table 5.50 CapsNet Experiment Confusion Matrix (Solana Networks Dataset)

True Label \ Predicted Label	AudioStream	Browsing	Chat	FT	P2P	VOIP	VideoStream
AudioStream	5465	375	369	45	12	52	124
Browsing	890	4989	148	151	26	168	145
Chat	200	45	6151	10	7	168	15
FT	60	47	39	6387	74	33	39
P2P	100	121	18	93	6178	17	48
VOIP	139	101	210	37	23	6043	70
VideoStream	903	280	280	29	11	43	4967

c. **Boosting the CapsNet classification model using XGBoost:** After building, training, and testing the CapsNet classification model using the train and test sets, we applied the eXtreme Gradient Boosting (XGBoost) on the model to boost its overall performance by classifying the misclassified data thereby forcing out the best accuracy of the model. The XGBoost classifier took the output of the CapsNet classification model as its input which is specified as the base estimator to train the XGB classifier with the number of estimators set to 450, an objective of multi: softmax, a learning rate of 0.5, maximum depth of 4 and 7 number of classes which is the target class. Next, we fitted the train set into the XGB classifier we created for classification of the encrypted network and tested the model's accuracy using the test set. The results of the experiment obtained are shown in table 5.51 and 5.52 below.

Table 5.51 CapsNet + XGBoost Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	76%	83%	79%
1	Browsing	85%	84%	84%
2	Chat	88%	94%	91%
3	FT	97%	98%	98%
4	P2P	98%	95%	96%
5	VOIP	95%	94%	94%
6	VideoStream	90%	80%	85%

Table 5.52 CapsNet + XGBoost Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
	AudioStream	5333	447	325	23	17	49	248
	Browsing	569	5477	140	37	35	87	172
True Label	Chat	176	31	6217	4	7	114	47
	FT	35	42	18	6520	43	11	10
	P2P	70	97	11	73	6252	15	57
	VOIP	115	107	124	20	12	6195	50
	VideoStream	700	277	247	17	19	47	5206

5.2.5 Ensemble Learning Technique

An ensemble learning technique is a process that uses multiple machine learning or deep learning techniques such as classifiers structured together to solve a particular problem and improve the system's accuracy which can as well make the performance better (Paul, 2018). There are different types of ensemble learning methods such as averaging, weighted average, stacking to mention just a few, but the commonly used methods are the boosting, bagging, stacking, and voting based ensemble learning (Tanner, 2019).

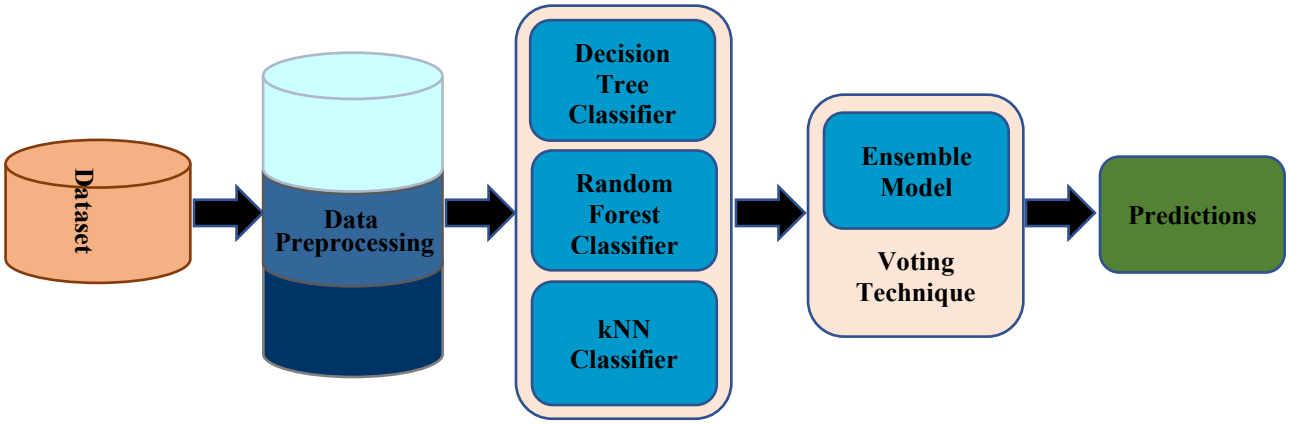


Figure 5.8 *Ensemble Structure for the Solana Networks dataset*

a. **Building the Ensemble model:** Here, we used the already preprocessed Solana Networks encrypted network traffic dataset to build the ensemble classification model. First, we import the Decision Tree, Random Forest, and k-Nearest Neighbors classifiers from sklearn in python. Next, we initialize each of the training models as a classifier since we are trying to classify the network traffic data. We then trained the models by fitting the train set into the model. Next, we tested the accuracy of the models using the test set. We imported the voting classifier from the sklearn ensemble and set the voting as hard to make predictions on the ensemble model using the test set.

b. **Efficiency analysis:** Here, we tested the classification accuracy of the ensemble classification model using the test set split. Table 5.53 and 5.54 below shows the results obtained.

Table 5.53 Ensemble Model Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	77%	83%	80%
1	Browsing	85%	85%	85%
2	Chat	88%	94%	91%
3	FT	98%	98%	98%
4	P2P	98%	96%	97%
5	VOIP	95%	93%	94%
6	VideoStream	90%	80%	85%

Table 5.54 Ensemble Model Experiment Confusion Matrix (Solana Networks Dataset)

True Label \ Predicted Label	AudioStream	Browsing	Chat	FT	P2P	VOIP	VideoStream
AudioStream	5326	392	333	21	24	53	293
Browsing	522	5564	138	31	37	81	144
Chat	176	28	6224	5	9	94	60
FT	34	47	18	6518	39	17	6
P2P	77	92	14	51	6280	16	45
VOIP	122	108	128	15	26	6178	46
VideoStream	678	294	252	14	21	40	5214

c. **Boosting the Ensemble classification model using XGBoost:** After building, training, and testing the ensemble model using the train and test sets, we applied the eXtreme Gradient Boosting (XGBoost) on the Ensemble model to boost its overall performance. The XGBoost classifier took the output of the ensemble model as its input (i.e. base estimator) to train the XGB classifier with the number of estimators set to 500, an objective of multi: softmax, a learning rate of 0.75, maximum depth of 4, and 7 number of classes which is the target class. Next, we fitted the train set into the XGB classifier we created for predictions and tested its accuracy using the test set. The results of the experiment obtained are shown in table 5.55 and 5.56 below.

Table 5.55 Ensemble Model + XGBoost Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	77%	82%	80%
1	Browsing	85%	84%	85%
2	Chat	88%	94%	91%
3	FT	98%	97%	98%
4	P2P	97%	96%	97%
5	VOIP	95%	93%	94%
6	VideoStream	90%	80%	84%

Table 5.56 Ensemble Model + XGBoost Experiment Confusion Matrix (Solana Networks Dataset)

True Label \ Predicted Label	AudioStream	Browsing	Chat	FT	P2P	VOIP	VideoStream
AudioStream	5312	421	330	17	25	56	281
Browsing	532	5506	142	33	42	88	174
Chat	170	34	6233	6	7	99	47
FT	33	48	16	6511	50	14	7
P2P	66	81	12	59	6284	17	56
VOIP	108	105	140	17	19	6189	45
VideoStream	699	280	248	12	20	46	5208

5.2.6 Machine Learning- Decision Tree Technique

A decision tree is a supervised machine learning that is used for classification and regression tasks or problems. It is in a tree-like structure where the nodes of the tree represent the independent variables of the dataset being used for analysis, the decision rule is being represented by the branch while the leaf nodes represent the outcome of the analysis (Gupta, 2017).

a. **Building the Decision Tree Classification Model:** We use the already preprocessed Solana Networks dataset which was done using the MinMax scaler and label encoder to build the decision tree model for the classification of Solana Networks encrypted network traffic. First, we import the Decision Tree classifier from sklearn in python. Next, we initialize the decision tree model as a classifier since we are building a classification model. We then trained the model by fitting the train set into the DT model using a k-fold of 5. Next, we tested the accuracy of the model by making predictions on the decision tree model using the test set. The results of the experiment are shown in table 5.57 and 5.58 below.

Table 5.57 Decision Tree Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	75%	79%	77%
1	Browsing	84%	82%	83%
2	Chat	86%	94%	90%
3	FT	97%	97%	97%
4	P2P	96%	94%	95%
5	VOIP	93%	92%	93%
6	VideoStream	87%	79%	83%

Table 5.58 Decision Tree Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
True Label	AudioStream	4975	451	357	25	26	79	363
	Browsing	540	5140	178	46	62	125	199
	Chat	173	33	6205	11	15	116	51
	FT	30	32	29	6374	67	24	6
	P2P	60	82	41	102	6053	36	66
	VOIP	129	89	177	21	40	5995	62
	VideoStream	686	304	213	18	34	69	4947

b. Boosting the Decision Tree classification model using XGBoost: After building, training, and testing our model using the train and test sets, we applied the eXtreme Gradient Boosting (XGBoost) on the decision tree model to boost its overall performance. The XGBoost classifier took the output of the decision tree model as its input (i.e. base estimator) to train the XGB classifier with the number of estimators set to 600, an objective of multi: softmax, a learning rate of 0.25, maximum depth of 8, and 7 number of classes which is the target class. Next, we fitted the train set into the XGB classifier we created for predictions and tested its accuracy using the test set. The results of the experiment obtained are shown in table 5.59 and 5.60 below.

Table 5.59 Decision Tree + XGBoost Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	77%	82%	80%
1	Browsing	86%	85%	85%
2	Chat	88%	95%	91%
3	FT	98%	98%	98%
4	P2P	98%	96%	97%
5	VOIP	95%	94%	95%
6	VideoStream	90%	80%	85%

Table 5.60 Decision Tree + XGBoost Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
True Label	AudioStream	5152	397	343	25	19	41	299
	Browsing	500	5343	131	29	49	93	145
	Chat	166	33	6244	10	5	103	43
	FT	23	34	15	6430	35	17	8
	P2P	65	76	20	66	6163	13	37
	VOIP	99	92	112	16	23	6130	41
	VideoStream	664	264	208	13	21	59	5042

c. **Boosting the Decision Tree model using AdaBoost:** Here, we preprocess the Solana Networks traffic dataset using the MinMax scaler and label encoder to convert the records to a machine-readable format to enable us to build the Adaboost-based classification model using the decision tree classification model as the base estimator. First, we import the Adaboost classifier from sklearn.ensemble in python. Next, we initialize our training model as a classifier since we are building a classification model by setting the base estimator to be the decision tree model, 800 number of estimators with a learning rate of 0.1. We then trained the AdaBoost model by fitting the train set into it with a k-fold of 5. Next, we tested the accuracy of the AdaBoost model by making predictions on the Adaboost model using the test set. The results of the experiment are shown in the table below.

Table 5.61 Decision Tree + AdaBoost Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	76%	80%	78%
1	Browsing	85%	83%	84%
2	Chat	87%	94%	91%
3	FT	97%	97%	97%
4	P2P	97%	95%	96%
5	VOIP	94%	92%	93%
6	VideoStream	87%	80%	83%

Table 5.62 Decision Tree + AdaBoost Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
	AudioStream	5034	416	339	27	26	54	380
	Browsing	573	5194	133	40	53	103	194
True Label	Chat	171	44	6236	6	10	95	42
	FT	32	52	27	6371	51	19	10
	P2P	67	70	30	62	6148	19	44
	VOIP	118	95	166	21	26	6012	75
	VideoStream	663	274	209	19	33	64	5009

d. Building the Gradient Boost model: Here, we preprocess the Solana Networks traffic dataset using the MinMax scaler and label encoder preprocessing module to convert the records into a machine-readable format to enable us to build the Gradient Boost classification model. First, we imported the Gradient Boosting Classifier and set the number of estimators to 200, a learning rate of 0.5, and a maximum depth of 6. We then fitted the train set into the Gradient Boost model to train the model. We tested the accuracy of the model by making predictions on the Gradient Boost-based model using the test set. The results of the experiment are shown in table 5.63 and 5.64 below.

Table 5.63 Gradient Boost Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	75%	80%	77%
1	Browsing	83%	80%	81%
2	Chat	83%	93%	88%
3	FT	95%	93%	94%
4	P2P	93%	93%	93%
5	VOIP	94%	92%	93%
6	VideoStream	86%	75%	80%

Table 5.64 Gradient Boost Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
True Label	AudioStream	5018	409	368	39	35	57	350
	Browsing	578	5046	191	66	60	116	233
	Chat	177	43	6154	54	17	115	44
	FT	59	50	304	6072	51	15	11
	P2P	70	121	31	124	5988	28	78
	VOIP	113	145	164	31	44	5964	52
	VideoStream	711	282	247	19	260	59	4693

e. **Building the Decision Tree model using the Bagging Technique:** For us to build this model, we used the Decision Tree model as the base estimator of the bagging technique model. First, we imported the Bagging Classifier and set the base estimator as the decision tree classification model, 27 number of estimators. We then fitted the train set into the Bagging technique model to train the model. We tested its accuracy by making predictions on the bagging model using the test set. The results of the experiment are shown in table 5.65 and 5.66 below.

Table 5.65 Decision Tree + Bagging Technique Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	77%	80%	78%
1	Browsing	84%	83%	83%
2	Chat	87%	94%	91%
3	FT	97%	97%	97%
4	P2P	96%	95%	96%
5	VOIP	94%	93%	93%
6	VideoStream	87%	80%	83%

Table 5.66 Decision Tree + Bagging Technique Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
	AudioStream	4993	447	351	33	24	57	371
	Browsing	522	5199	144	41	65	115	204
True Label	Chat	168	32	6212	14	10	119	49
	FT	23	33	22	6395	63	15	11
	P2P	60	84	25	84	6108	25	54
	VOIP	98	87	150	22	39	6061	56
	VideoStream	649	284	209	15	27	69	5018

5.2.7 Random Forest (RF) Technique

Random Forest is the last classification algorithm we built for the classification of encrypted network traffic data for the Solana Networks dataset in the experimentation stage.

a. **Building the Random Forest Model:** To build the random forest classification model, we utilized the already preprocessed dataset which was done using the MinMax scaler and label encoder preprocessing module from sklearn preprocessing in python to build the RF model for the classification of Solana Networks encrypted network traffic. The first thing we did is to import the Random Forest classifier from the sklearn ensemble in python. Then we initialize the RF model as a classifier for the classification of the encrypted network traffic. Next, we specified the number of

estimators as 650 before training the model by fitting the train set into the model. For us to determine the performance of the model based on its accuracy, we tested the model by making predictions on the Random Forest model using the test set. The results of the experiment are shown in table 5.67 and 5.68 below.

Table 5.67 Random Forest Technique Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	77%	82%	79%
1	Browsing	86%	84%	85%
2	Chat	87%	94%	91%
3	FT	98%	97%	98%
4	P2P	97%	95%	96%
5	VOIP	94%	93%	94%
6	VideoStream	88%	81%	84%

Table 5.68 Random Forest Technique Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
	AudioStream	5253	373	335	22	38	66	355
	Browsing	519	5489	143	37	39	101	189
True Label	Chat	158	35	6217	4	12	112	58
	FT	29	50	20	6506	46	20	8
	P2P	67	81	13	53	6276	19	66
	VOIP	111	108	127	18	22	6181	56
	VideoStream	647	272	251	12	20	50	5261

b. **Boosting Random Forest classification model using XGBoost:** After building, training, and testing the RF model using the train and test sets, we applied the eXtreme Gradient Boosting (XGBoost) on the RF model to boost its overall performance and force out the best accuracy of the classification model. The XGBoost classifier took the RF model as its input (i.e. base estimator) to train the XGBoost with the number of estimators set to 600, an objective of multi: softmax, a

learning rate of 0.5, maximum depth of 8 and 7 number of classes which is the target class. Next, we performed predictions on the XGB classifier model we created using the test set. The results of the experiment obtained are shown in table 5.69 and 5.70 below.

Table 5.69 Random Forest + XGBoost Experiment Result for each category (Solana Networks Dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	78%	82%	80%
1	Browsing	86%	86%	86%
2	Chat	88%	95%	91%
3	FT	98%	97%	98%
4	P2P	97%	96%	97%
5	VOIP	95%	93%	94%
6	VideoStream	88%	81%	84%

Table 5.70 Random Forest + XGBoost Experiment Confusion Matrix (Solana Networks Dataset)

		Predicted Label						
	AudioStream	5277	396	335	18	28	56	332
	Browsing	484	5574	138	31	41	75	174
True Label	Chat	160	23	6242	2	7	105	57
	FT	31	48	17	6512	49	13	9
	P2P	61	83	12	45	6293	23	58
	VOIP	99	101	134	19	22	6189	59
	VideoStream	639	285	248	16	22	49	5254

5.3 Comparison Analysis of the different experiments performed on the public dataset

Here, we compare the experimental results of all the models we designed above using the public encrypted network traffic dataset in terms of their overall accuracy, weighted precision, weighted recall, and weighted F1 score. From the above experiments, we were able to design different types of classification models that we used to analyze the public encrypted network traffic dataset to classify them to their different network traffic categories. The models we designed are classified under Machine Learning (ML) and Deep Learning

(DL) techniques. For the ML technique, we created the Decision Tree classification model, Decision Tree merged with Bagging technique classification model, Decision Tree merged with the Boosting techniques which are AdaBoost and eXtreme Gradient Boosting classification models, a Gradient Boosting classification model, Random Forest classification model, Random Forest merged with eXtreme Gradient Boosting classification model and Ensemble model using the hard voting and XGBoost techniques. For Deep Learning, we created the ANN classification model, ANN merged with the XGBoost classification model, 1D-CNN classification model, 1D-CNN merged with XGBoost classification model, RNN-LSTM classification model, RNN-LSTM merged with XGBoost classification model, CapsNet classification model and CapsNet merged with XGBoost classification model. All these models were designed and used for the multi-class experiments. Different results were obtained for different classification models after performing the classification experiment on them. The comparison result is explained in table 5.71 below.

Table 5.71 Comparison of The Results of The Experiments Performed on The Public Dataset

Models / Measures	DT	DT + Bagging	DT + AdaBoost	Gradient Boost	DT + XGBoost	RF	RF + XGBoost	ANN	ANN + XGBoost	CNN	CNN + XGBoost	RNN-LSTM	RNN-LSTM + XGBoost	CapsNet	CapsNet + XGBoost	Ensemble (Voting)	Ensemble (Voting) + XGBoost
Overall Accuracy	90%	93%	94%	93%	95%	94%	95%	89%	96%	97.76%	96%	86%	93%	88%	96%	91%	95%
Weighted Precision	90%	93%	94%	93%	95%	94%	95%	89%	96%	96%	96%	87%	93%	88%	96%	92%	95%
Weighted Recall	90%	93%	94%	93%	95%	94%	95%	89%	96%	97.6%	96%	86%	93%	88%	96%	91%	95%
Weighted F1 Score	90%	93%	94%	93%	95%	94%	95%	89%	96%	96.87%	96%	87%	93%	88%	96%	91%	95%

Table 5.71 above shows the results of the experiments we performed on the public dataset using different techniques of data analysis and its performance was determined using the overall accuracy, weighted precision, weighted recall, and weighted f1 score metric measurements. From the table, it is evident that the 1-dimensional Convolution Neural Network (CNN) technique performed better than other techniques by achieving an overall accuracy of 97.76%, weighted recall of 97.6%, weighted precision of 96%, and weighted f1 score of 96.87%. While Recurrent Neural Network (RNN) using the Long Short-Term Memory (LSTM) technique has the lowest accuracy of 86%, weighted precision, and weighted f1 score of both 87% and a weighted recall of 86%. The deep learning techniques (ANN, 1D-CNN, CapsNet, and RNN-LSTM) were boosted using eXtreme Gradient Boosting (XGBoost) and it boosted the accuracy of RNN-LSTM, CapsNet and ANN techniques to 93%, 96%, and 96% respectively while in 1D-CNN technique it did not perform better achieving an accuracy of 96%. For XGBoost not to perform better in the 1D-CNN classification model when combined but achieved a weighted f1 score of 96% which is close to the weighted f1 score of the 1D-CNN classification model alone, this means that f1 score

should be used to measure the performance of the models if we need to seek a balance between precision and recall where there is an uneven classification (a large number of Actual Negatives) of the datasets during analysis (Koo, 2018). We then introduced the public datasets to the Ensemble model designed using the voting (hard) technique and then boosted it using the XGBoost technique. The results of the Ensemble Learning classification model using the hard-voting technique showed 91% for its overall accuracy, weighted recall, and weighted f1 score respectively and 92% for its weighted precision. When the XGBoost technique was applied to it, the results were increased to 95% for its accuracy, weighted precision, weighted recall, and weighted f1 score, respectively. Whereas for the ML techniques, the Random Forest technique performed better achieving an accuracy of 94% while the Decision Tree technique achieved an accuracy of 90% which is the lowest accuracy achieved for the ML techniques. The Random Forest technique was boosted using the XGBoost technique and it achieved an accuracy of 95% which is the same accuracy obtained by the Decision Tree technique when boosted using the XGBoost technique. The Decision Tree accuracy also increased when the bagging and AdaBoost technique was applied to it to 93% and 94% respectively. The overall accuracy of 93% was achieved when the Gradient Boosting classification technique was applied to the public dataset. Generally, the XGBoost technique increased the accuracy of all the models designed apart from that of the 1D-CNN when applied to it. The best accuracy of the XGBoost, when applied to the models, is 96% accuracy which was obtained by ANN merged with XGBoost, 1D-CNN merged with XGBoost and CapsNet merged with XGBoost. While RF merged with XGBoost, DT merged with XGBoost and Ensemble learning model using hard voting techniques merged with XGBoost obtained the same accuracy results of 95% respectively.

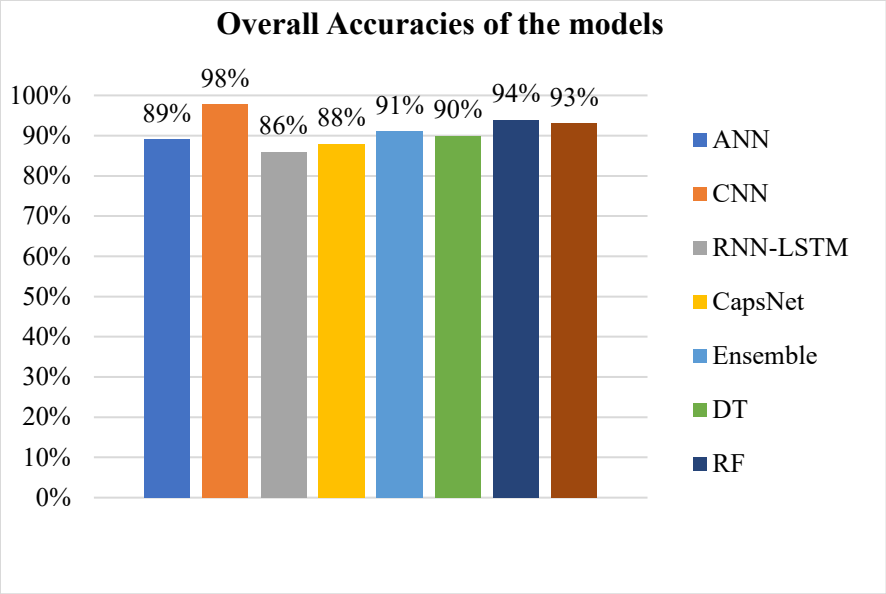


Figure 5.9 *Overall Accuracies of the models designed for the Public Dataset*

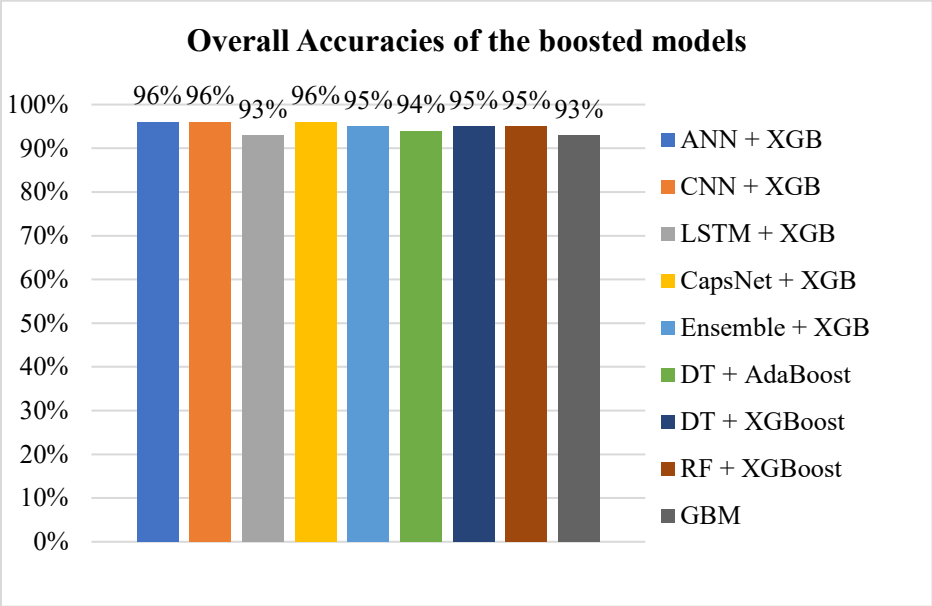


Figure 5.10 *Overall Accuracies of the boosted models designed for the Public Dataset*

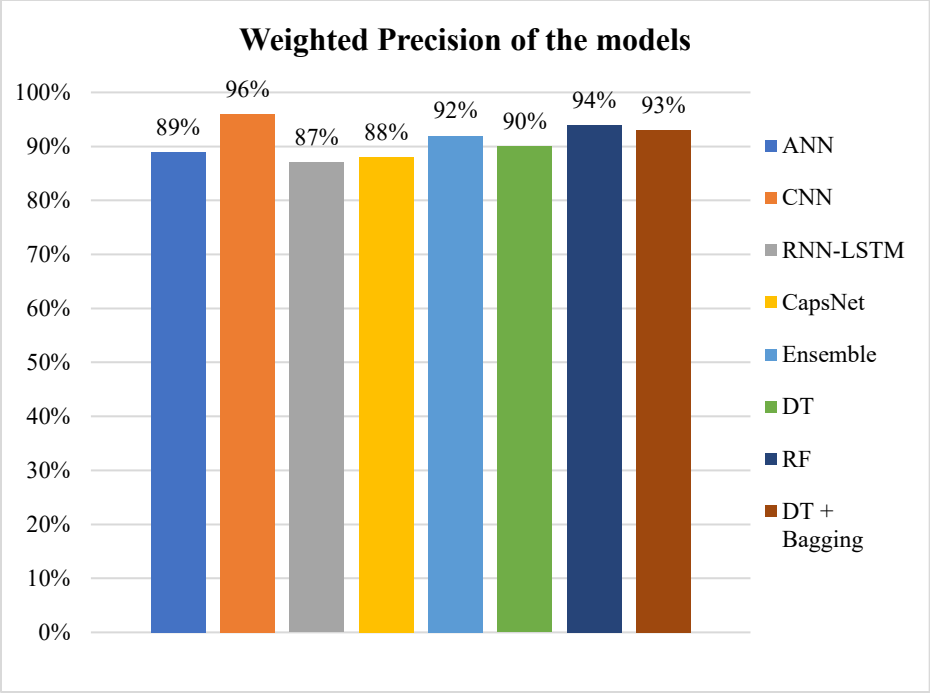


Figure 5.11 *Weighted Precision of the models designed for the Public Dataset*

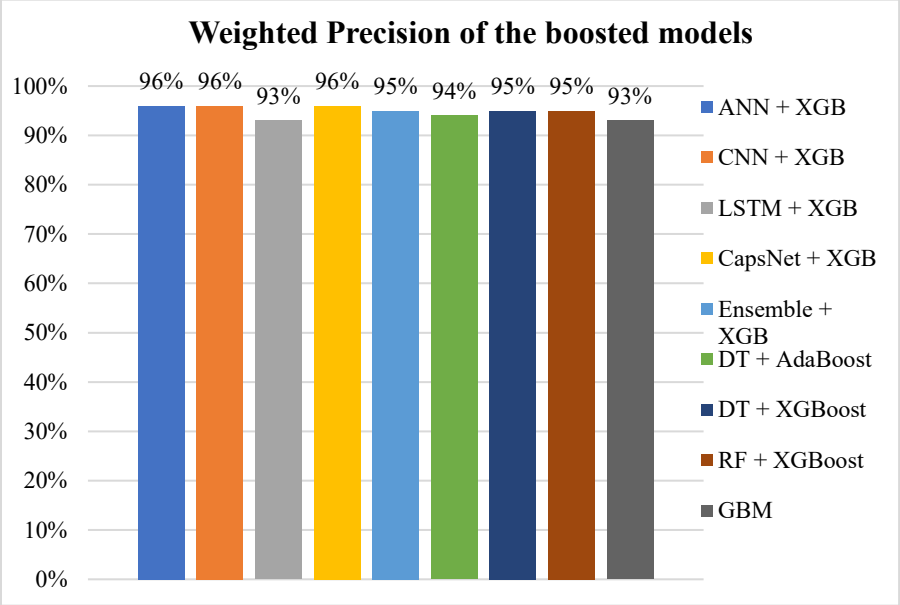


Figure 5.12 *Weighted Precision of the boosted models designed for the Public Dataset*

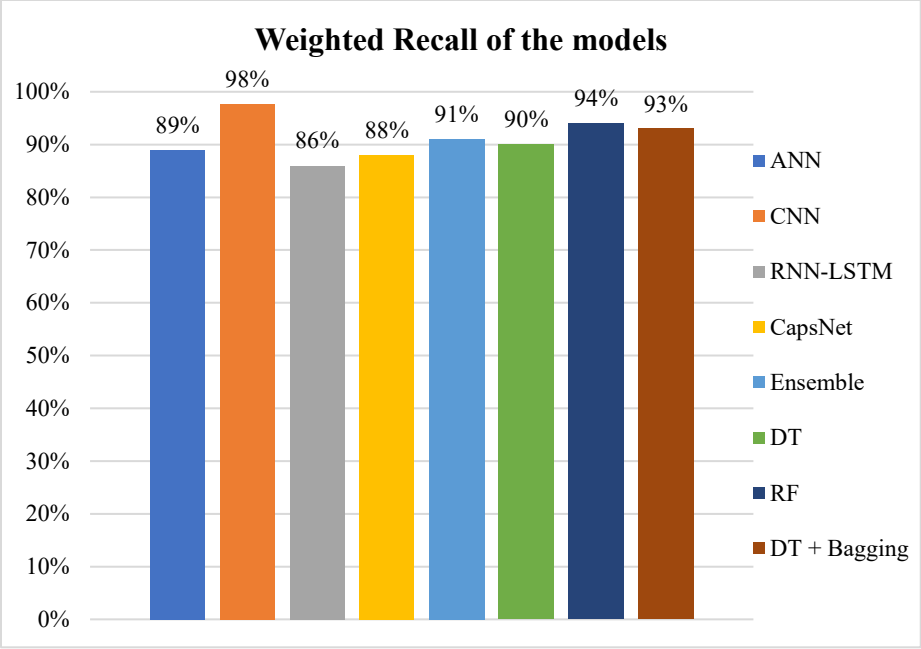


Figure 5.13 *Weighted Recall of the models designed for the Public Dataset*

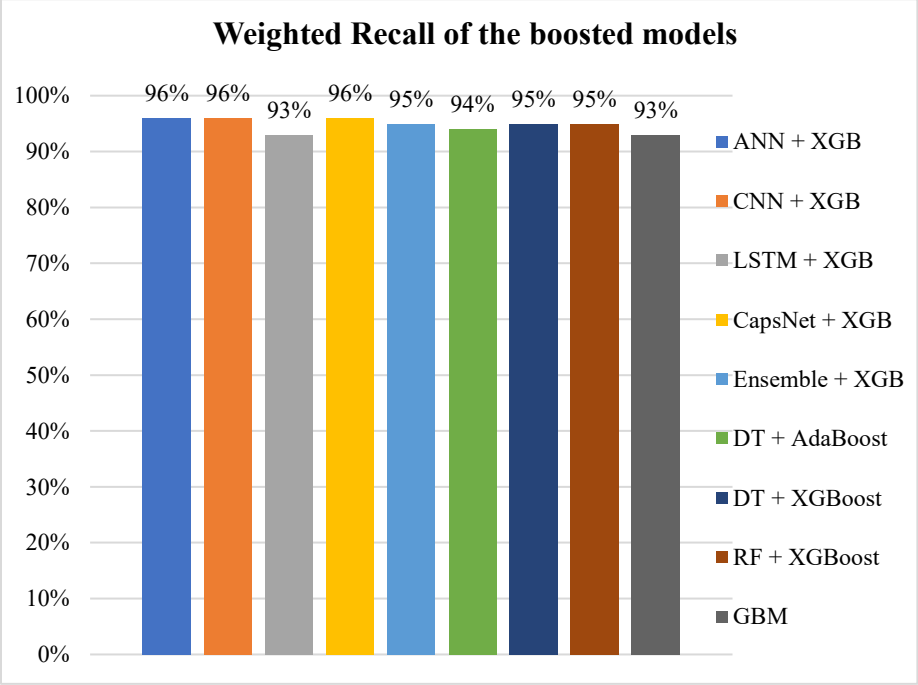


Figure 5.14 *Weighted Recall of the boosted models designed for the Public Dataset*

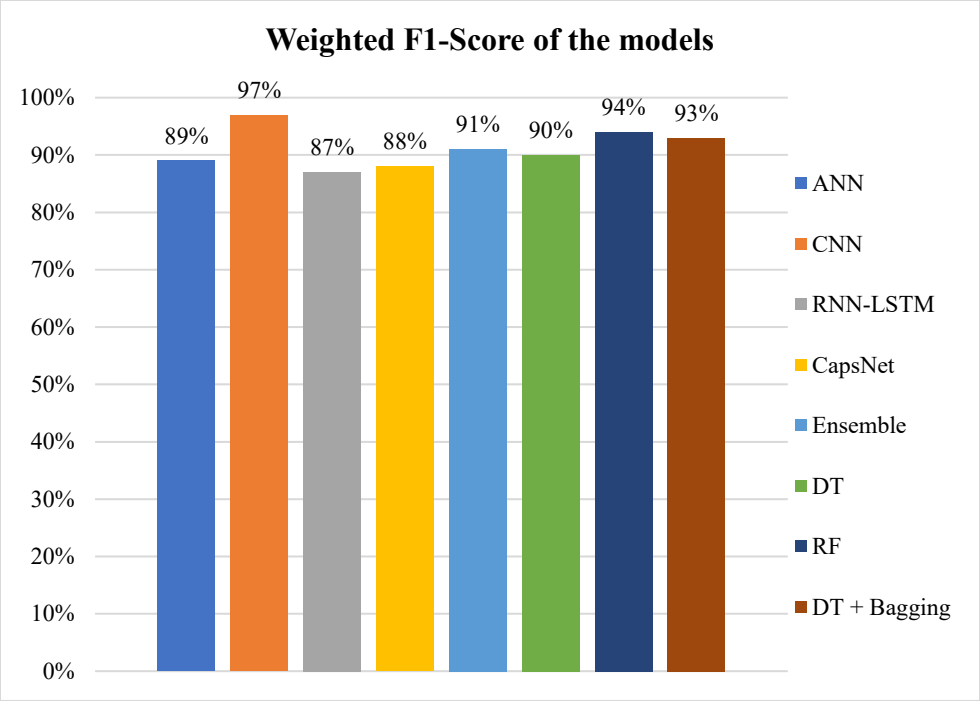


Figure 5.15 *Weighted F1-Score of the models designed for the Public Dataset*

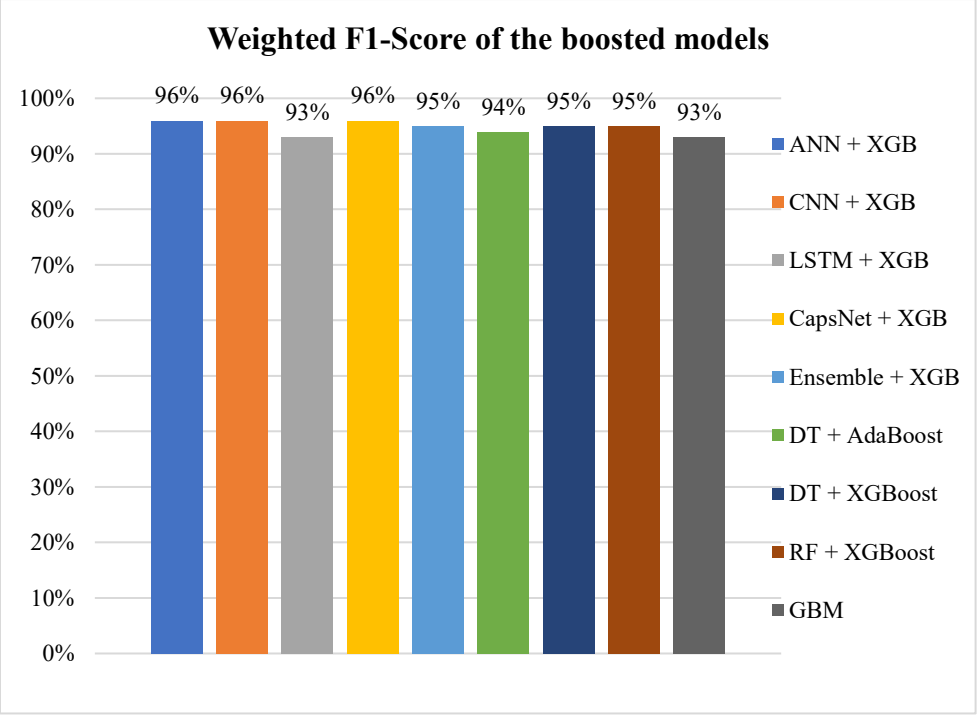


Figure 5.16 *Weighted F1-Score of the boosted models designed for the Public Dataset*

5.4 Comparison Analysis of the different experiments performed on the Solana Networks dataset

Here, we also compare the experimental results of all the models we designed above using the Solana Networks encrypted network traffic dataset in terms of their overall accuracy, weighted precision, weighted recall, and weighted F1 score. From 5.2 of the experiments above, we were able to design different types of classification models using the Solana Networks encrypted network traffic dataset to classify traffic data to their different network traffic categories. The models we designed in 5.2 of the experiments are classified under Machine Learning (ML) and Deep Learning (DL) techniques. For the ML technique, we created the Decision Tree classification model, Decision Tree merged with Bagging technique classification model, Decision Tree merged with the Boosting techniques which are AdaBoost and eXtreme Gradient Boosting classification models, a Gradient Boosting classification model, Random Forest classification model, Random Forest merged with eXtreme Gradient Boosting classification model and Ensemble model using the hard voting and XGBoost techniques. For Deep Learning, we created the ANN classification model, ANN merged with the XGBoost classification model, 1D-CNN classification model, 1D-CNN merged with XGBoost classification model, RNN-LSTM classification model, RNN-LSTM merged with XGBoost classification model, CapsNet classification model and CapsNet merged with XGBoost classification model. All these models were designed and used for the multi-class classification experiments. Different results were obtained for different models after performing the classification experiment on them. The comparison result obtained from the experiments is given in table 5.72 below.

Table 5.72 Comparison of The Results of The Experiments Performed on The Solana Networks Dataset

Models / Measures	DT	DT + Bagging	DT + AdaBoost	Gradient Boost	DT + XGBoost	RF	RF + XGBoost	ANN	ANN + XGBoost	CNN	CNN + XGBoost	RNN-LSTM	RNN-LSTM + XGBoost	CapsNet	CapsNet + XGBoost	Ensemble (Voting)	Ensemble (Voting) + XGBoost
Overall Accuracy	88%	89%	89%	87%	90%	90%	90%	84%	90%	87%	90%	87%	90%	87%	90%	90%	90%
Weighted Precision	88%	89%	89%	87%	90%	90%	90%	86%	90%	88%	90%	88%	90%	88%	90%	90%	90%
Weighted Recall	88%	89%	89%	87%	90%	90%	90%	84%	90%	87%	90%	87%	90%	87%	90%	90%	90%
Weighted F1 Score	88%	89%	89%	87%	90%	90%	90%	84%	90%	87%	90%	87%	90%	88%	90%	90%	90%

Table 5.72 above shows the results of the experiments we performed on the Solana Networks traffic dataset using different techniques of data analysis and its performance was determined using the overall accuracy, weighted precision, weighted recall, and weighted f1 score metric measurements. From the table, CapsNet, 1D-CNN, and RNN-LSTM techniques obtained the best overall accuracy of 87% for the deep learning models while Random Forest and the ensemble learning techniques achieved a better overall accuracy result of 90% as well for the machine learning models. ANN technique is the only model that performed poorly achieving an accuracy of 84%. The deep learning techniques (ANN, 1D-CNN, CapsNet, and RNN-LSTM) were boosted using eXtreme Gradient Boosting (XGBoost) and it boosted the overall accuracy of RNN-LSTM, 1D-CNN, CapsNet and ANN techniques to 90%, 90%, 90%, and 90% respectively, which shows that this is the best overall accuracy XGBoost can force out of the models using the Solana Networks network traffic dataset compared to that of the public dataset which achieved 96%. Random Forest and Decision Tree boosted techniques were both boosted using the XGBoost technique and they achieved an accuracy of 90%. The Random Forest already achieved an overall accuracy

of 90% before boosting which also supports our statement that 90% is the best overall accuracy the dataset can give us using it in different classification models. Decision Tree using the bagging technique and boosted with AdaBoost both achieved the same overall accuracy of 89%. Generally, it is evident the machine learning techniques outperformed the deep learning techniques before boosting them using the AdaBoost, Gradient Boosting, and eXtreme Gradient Boost techniques. The only explanation we can give to this is because of the dataset, which has brought us to the conclusion that the dataset used in designing a model has a major influence on the model's overall performance. The XGBoost technique increased the accuracy of all the models designed apart from that of the Random Forest and Ensemble Learning technique when applied to it because they have attained the maximum overall accuracy the dataset can give. The results obtained from the experiments above using the public and Solana Networks traffic dataset shows that deep learning models are suitable for the classification of encrypted network traffic dataset having achieved the highest accuracy in most cases compared to that of the machine learning techniques.

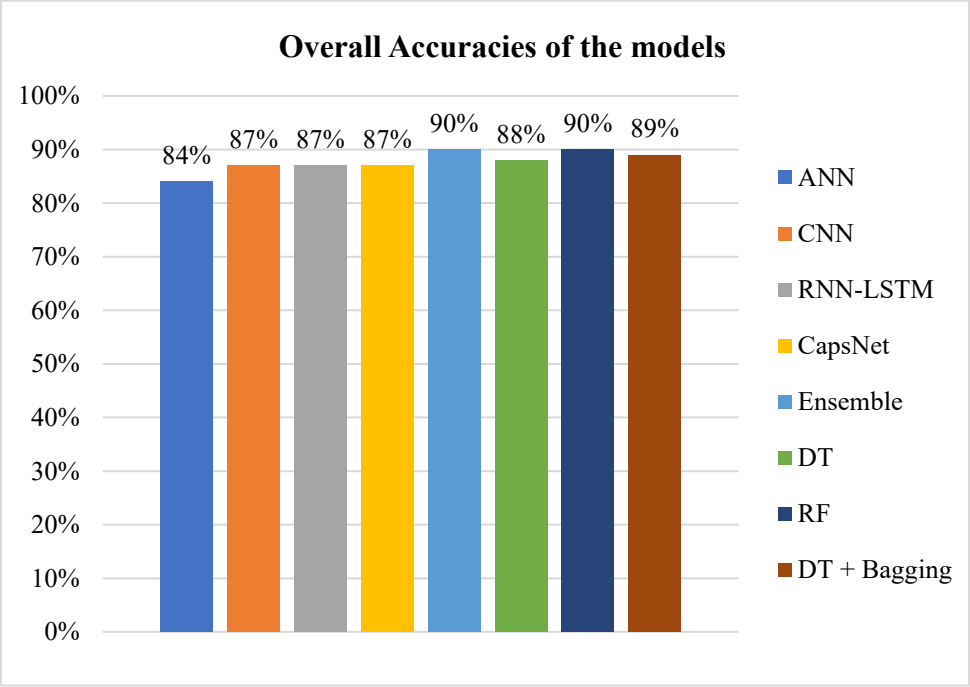


Figure 5.17 *Overall Accuracies of the models designed for the Solana Networks Dataset*

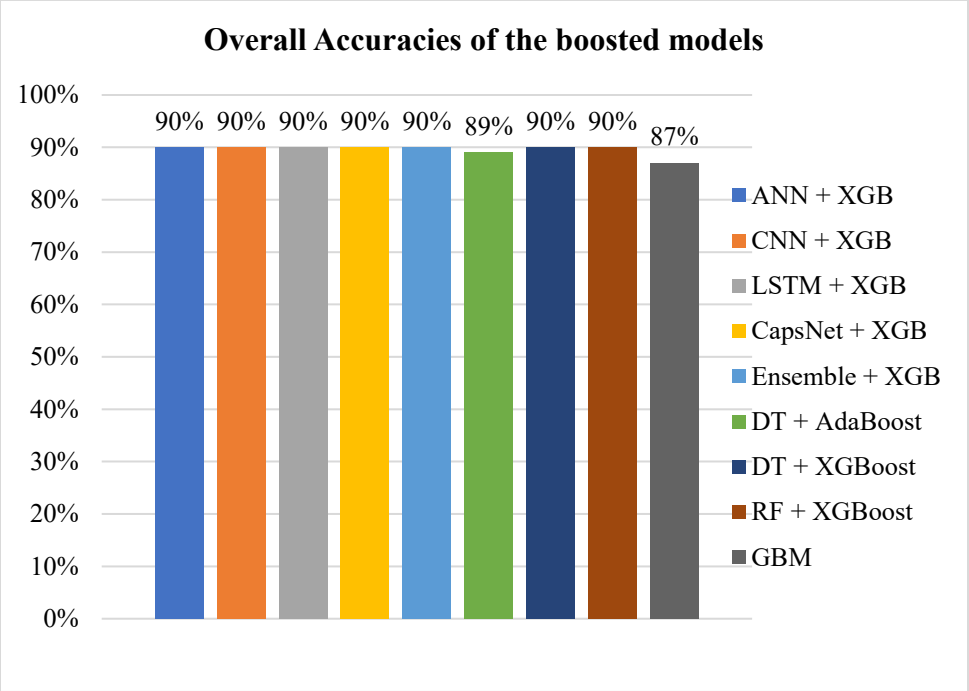


Figure 5.18 *Overall Accuracies of the boosted models designed for the Solana Networks Dataset*

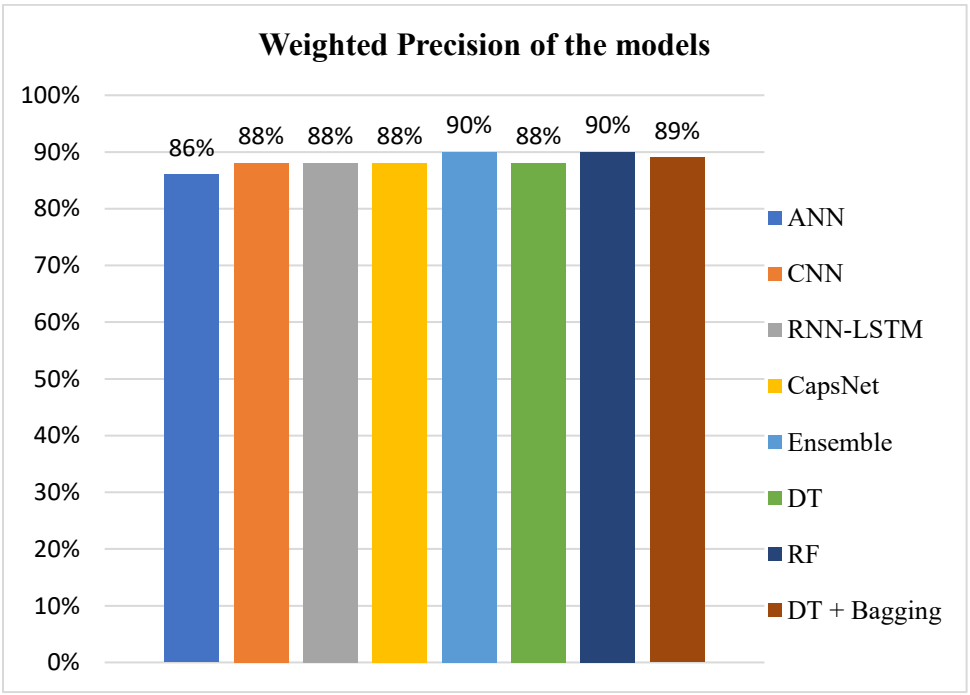


Figure 5.19 *Weighted Precision of the models designed for the Solana Networks Dataset*

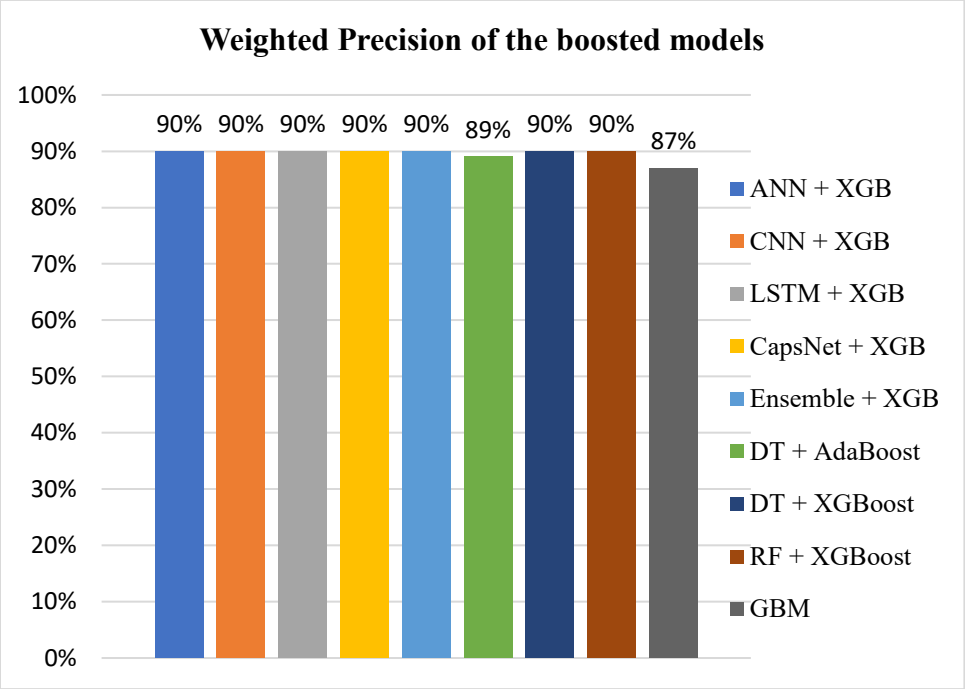


Figure 5.20 *Weighted Precision of the boosted models designed for the Solana Networks Dataset*

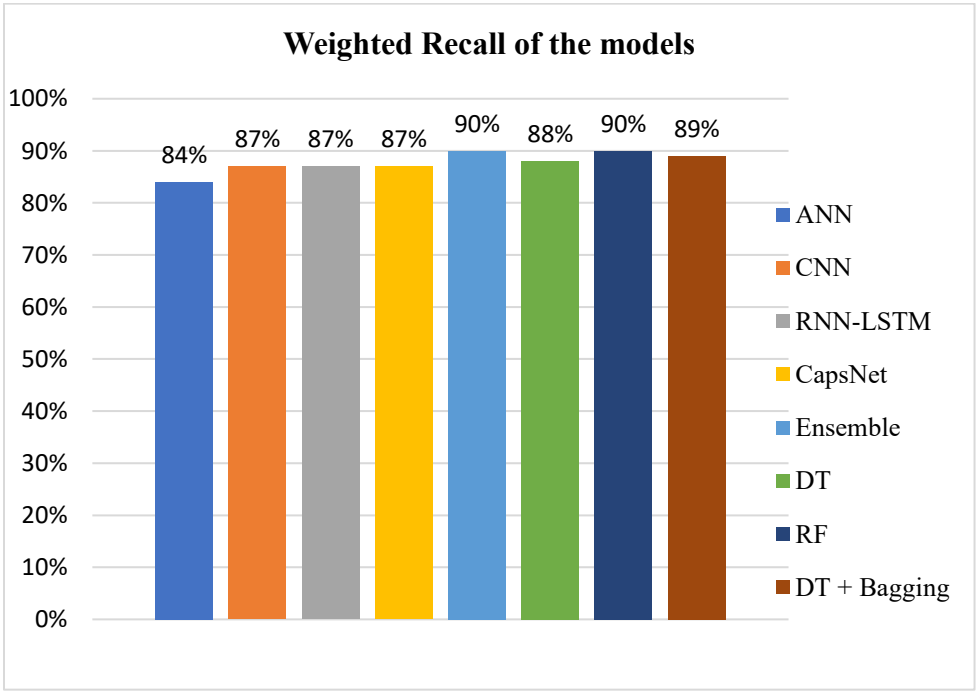


Figure 5.21 *Weighted Recall of the models designed for the Solana Networks Dataset*

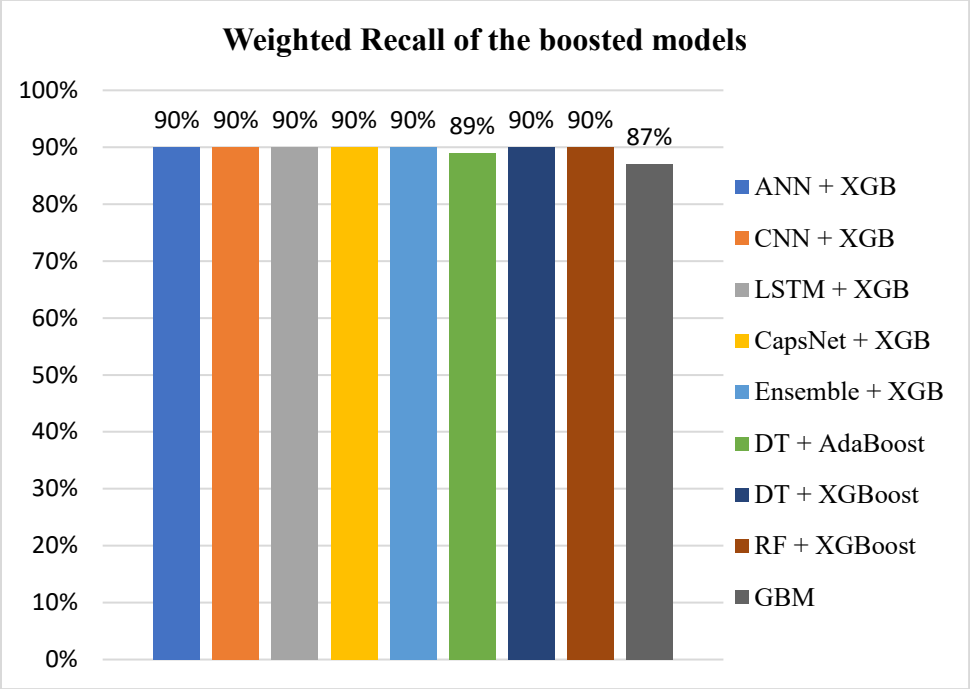


Figure 5.22 *Weighted Recall of the boosted models designed for the Solana Networks Dataset*

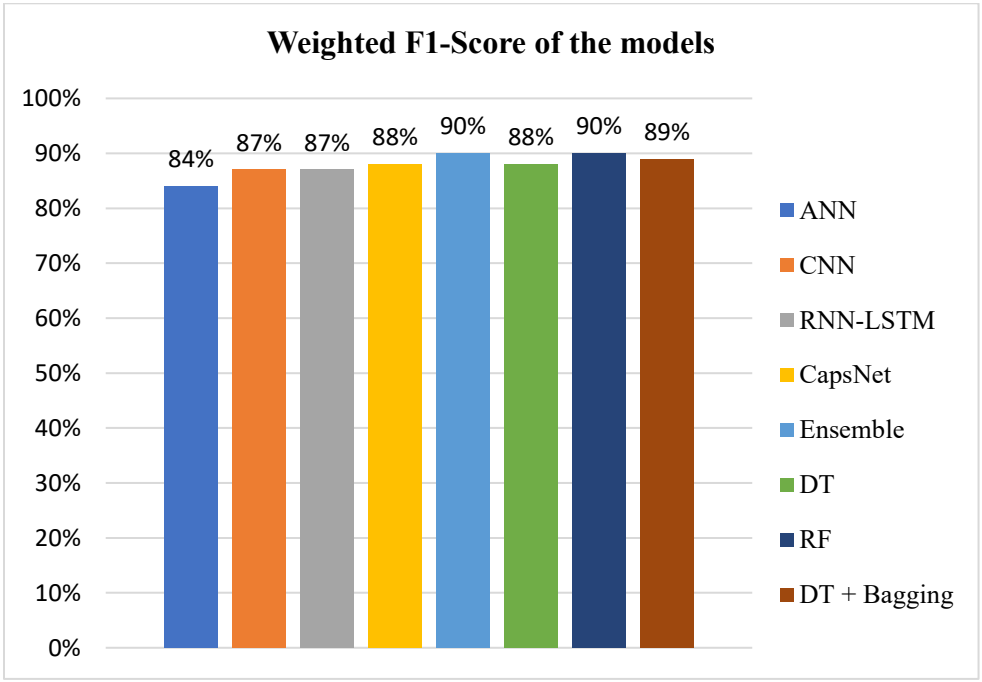


Figure 5.23 *Weighted F1-score of the models designed for the Solana Networks Dataset*

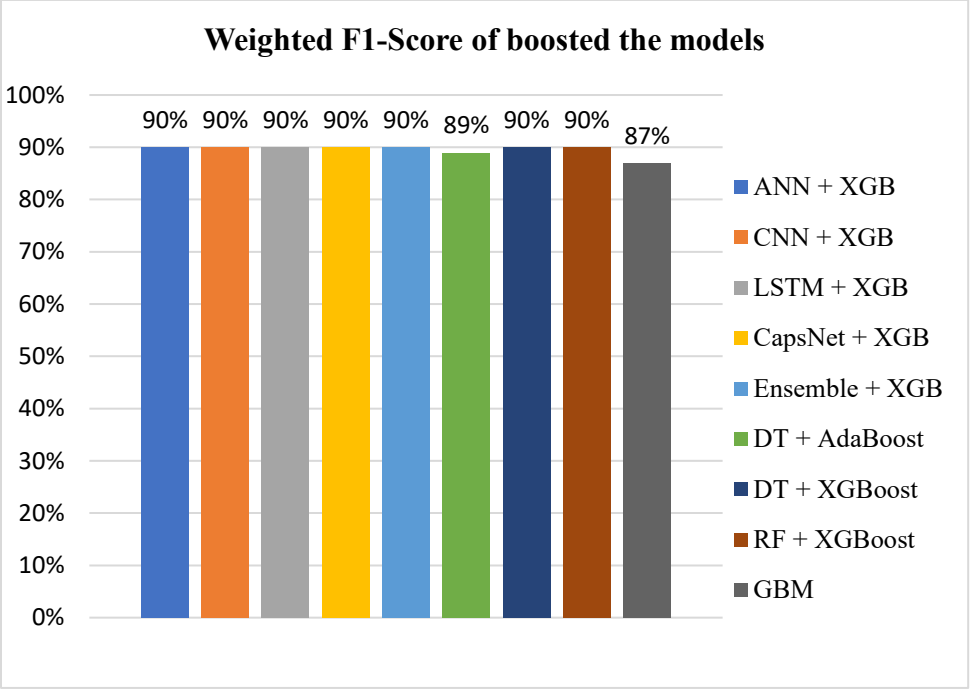


Figure 5.24 *Weighted F1-score of the boosted models designed for the Solana Networks Dataset*

Chapter 6: Proposed Solution

The goal is to classify encrypted network traffic data to its appropriate traffic category. This may help the Internet Service Providers (ISPs) to offer a Quality of Service to the customers as well as monitoring and managing the network, securing the network from malware and intrusions, making proper network planning and troubleshooting of the network to mention just a few. To achieve this, we built an Ensemble Learning Technique Model that is based on the best performance of the Deep Learning and Machine Learning techniques we designed in chapter 5 and applied the stacking technique on it (Ensemble Model) to determine the best performance result of the classification model. The system will have data containing encrypted network traffic (non-VPN) generated by customers waiting to be classified into its specific traffic category. The data will be stored and preprocessed for splitting into train and test dataset to feed it to the Ensemble Learning Technique system to learn the different significant patterns that are associated with the dataset and then classify them into the appropriate traffic category. The classification result of the encrypted network traffic will enable us to determine the accuracy and efficiency of the model and compare it to known classes. The system will also be able to accept new non-VPN network traffic data as an input for classification by relating it to the patterns it has already learned during the training stage.

6.1 System Model

6.1.1 Conceptual Architecture

The conceptual architecture refers to the idea and thought that we considered to form the foundation of our proposed model and to enable us in designing and building the proposed model. Figure 6.1 below depicts the conceptual architecture of the proposed system.

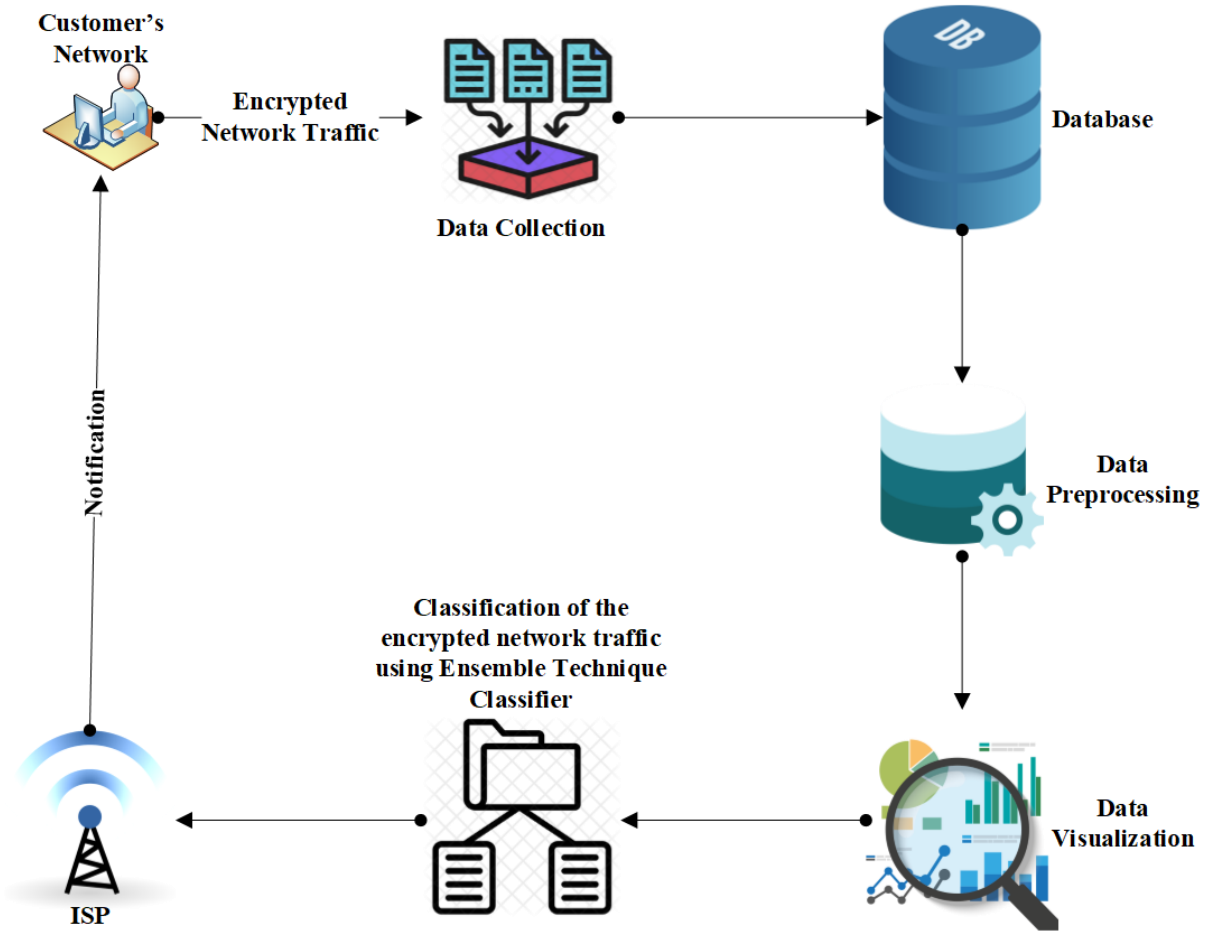


Figure 6.1 *Conceptual Architecture*

From figure 6.1 above, the encrypted network traffic data is being generated by the customer while utilizing various applications on his/her electronic devices which are capable of connecting to the internet, then the Internet Service Provider (ISP) extracts those traffic data that are generated and stores them in a database for that customer as well as labeling them properly. The dataset is then preprocessed, visualized, and fed into the classifier for the classification of the encrypted network traffic. After the classification stage, the results will be displayed for the ISP to view, interpret, and implement the necessary changes if any to enable them to meet the customer's expectations. The proposed ensemble learning model is a machine learning

model or technique where multiple base classification models or weak learners as often called are trained independently and then merged in the right way to feed the output to a new classification model which is the Random Forest Classifier to solve the classification problem as well as obtaining better performances and a robust model.

6.1.2 Technical Architecture

Figure 6.2 below illustrates the proposed system of technical architecture and implementation. The encrypted network traffic dataset is collected and then sets of flow statistical features are extracted and converted into a csv file format. The extracted statistical features data are then cleaned by removing the null and nan values and converting the independent variables with floating numbers to a float data type and the data is preprocessed using the required preprocessing libraries such as the label encoder, standard scaler, MinMax Scaler, and the one-hot encoder to transform it into a machine-readable format. The preprocessed data is then split into train and test sets and fed into the different classification models (CapsNet+XGBoost, ANN+XGBoost, RF+XGBoost, DT+ AdaBoost, DT+XGBoost) that makes up the ensemble model for training and testing using the stacking technique and k-fold cross-validation of 10. The outputs from the different classification models are merged to form a new train set that will be used as input for the second stage classification algorithm (Random Forest Classifier) which is the final model of the ensemble system. The new train set is then fed into the Random Forest Classifier being the new classification model for training and then tested using the test set to classify the encrypted network traffic. The stacking technique studies the heterogeneous base classification models to learn them in parallel and combines their outputs to train the classification model (Random Forest Classifier) for predictions that are based on the different classification models' predictions. The stacking technique produces a strong model to improve the prediction.

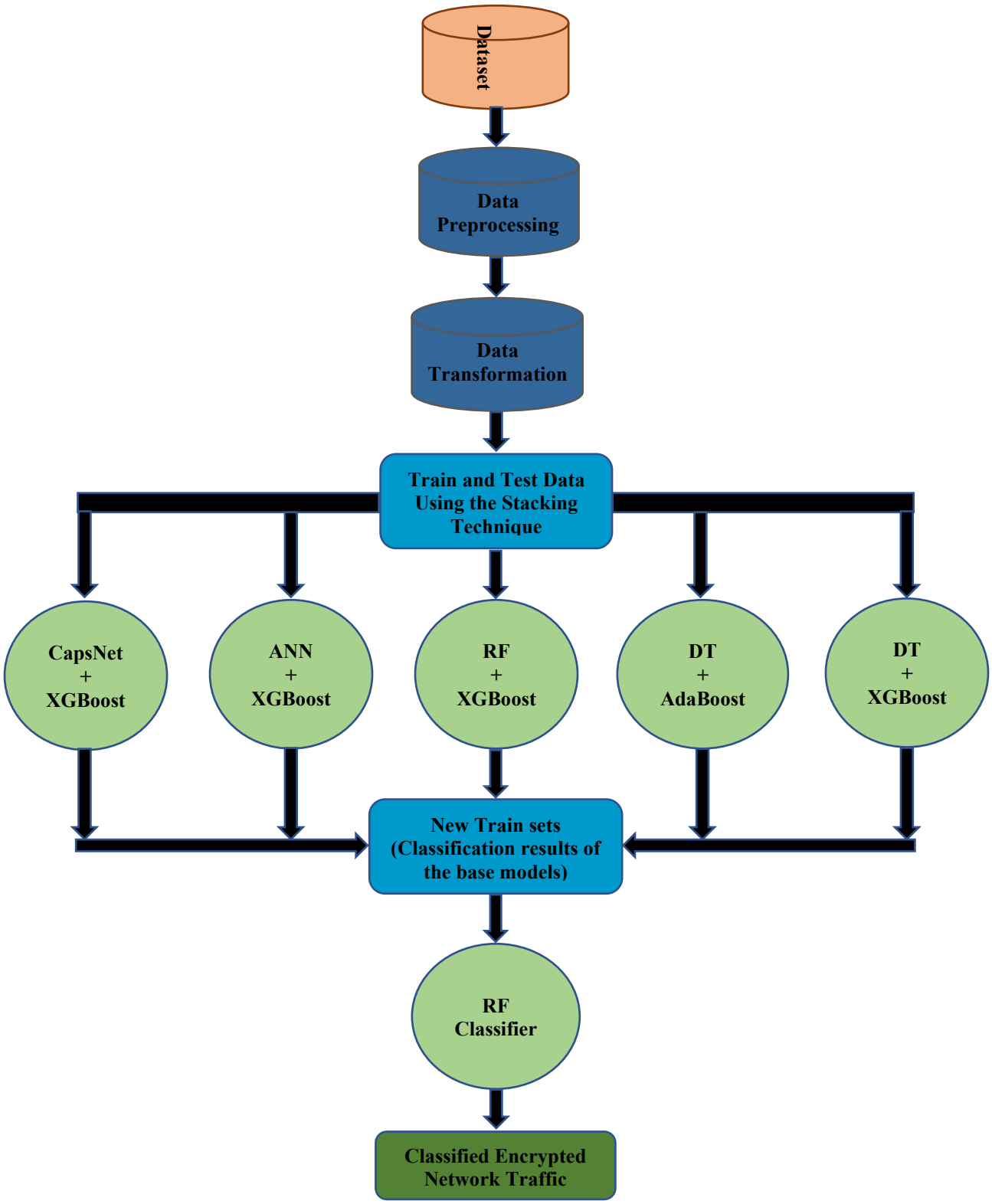


Figure 6.2 *Proposed System Design and Implementation*

Algorithm 1: Encrypted Network Traffic Classification

$\boldsymbol{x = number\ of\ features\ multiplied\ by\ number\ of\ training\ samples}\ (\boldsymbol{n * m})$
$\boldsymbol{y = network\ traffic\ category\ classes}\ (\boldsymbol{y_1, y_2, \ldots, y_n})$

Input: Network traffic dataset $\boldsymbol{S} = \{(\boldsymbol{x_1, y_1}), (\boldsymbol{x_2, y_2}), \ldots, (\boldsymbol{x_n, y_n})\}$;
First stage learning algorithms $\boldsymbol{L_1, \ldots, L_N}$;
Second stage learning algorithm $\boldsymbol{L}$.

Process:
$\boldsymbol{for\ n = 1, \ldots, N}$:
$\boldsymbol{b_n = L_n(I)}$ // Train the first stage individual learner $\boldsymbol{b_n}$ by applying
// the first stage learning algorithm $\boldsymbol{L_n}$ to the dataset $\boldsymbol{S}$
$\boldsymbol{end}$;
$\boldsymbol{S' = 0}$; // Generate new train sets
$\boldsymbol{for\ i = 1, \ldots, n}$:
$\boldsymbol{for\ n = 1, \ldots, N}$:
$\boldsymbol{z_{in} = b_n(x_i y_i)}$ // Use $\boldsymbol{b_n}$ to classify the test set $\boldsymbol{x_i y_i}$
$\boldsymbol{end}$;
$\boldsymbol{S' = S' \cup \{((z_{i1}, z_{i2}, \ldots, z_{iN}), y_i)\}}$
$\boldsymbol{end}$;
$\boldsymbol{b' = L(S')}$. // Train the second stage learner $\boldsymbol{b'}$ by applying the
// second stage learning algorithm $\boldsymbol{L}$ to the new train sets $\boldsymbol{S'}$
Output: $\boldsymbol{B(xy) = b'\ (b_1(xy), \ldots, b_n(xy))}$

Algorithm 1 is the high-level algorithm. For the encrypted network traffic classification, the data set contains the independent variables, the instances, and the target class which are the traffic categories. Let x and y denote the instances and the set of the target classes, respectively.

Therefore, $x = (n * m)$. Where n is the number of independent variables and m is the number of the samples The dataset is then split into train and test sets. The training dataset is given as

$$S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \quad (1)$$

where $x_i \in X\ and\ y_i \in Y (i = 1, \ldots, n)$. After the split, several individual learners or classifiers are generated in the first stage using different classification algorithms $L_1, \ldots, L_N$ and then train the classifiers using the train set. The output from the classifiers is merged to form new train set S' that will be fed into the second stage classification algorithm L**.** The new train set is given as

$$S' = S' \cup \{((z_{i1}, z_{i2}, \ldots, z_{iN}), y_i)\} \quad (2)$$

Where $(z_{i1}, z_{i2}, \ldots, z_{iN}), y_i$ represents the outputs from the first stage individual learners or classifiers. The second classification algorithm L is then used to train the new train set using the second stage learner $b' = L(S')$ and to classify the network traffic data into its appropriate class. The output is given as

$$B(xy) = b'\left(b_1(xy), \ldots, b_N(xy)\right) \quad (3)$$

Where $B(xy)$ is the ensemble classifier, b' is the second stage classification model which contains classification models from the first stage individual learners $\left(b_1(xy), \ldots, b_N(xy)\right)$.

6.1.3 Statistical Features Extracted from the dataset

The dataset contains a series of flows from the network traffic generated by users connected on the internet. The flow is a sequence of packets consisting of the same Source IP, Destination IP, Source Port, Destination Port, and Protocol (TCP or UDP) values. Here, the flows are bidirectional (forward and backward directions). For the flow(s) to be generated, the features that are associated with each flow needs to be calculated. There are tools like the dpkt module, NetMate that can generate flows, and the associated statistical features. To extract the statistical features associated with the network traffic datasets, CICFlowMeter was used. CICFlowMeter is a network traffic flow generator that is written in Java and offers more flexibility when it comes to choosing the statistical features you want to extract from the raw network traffic which are in pcap files. It is owned by the University of New Brunswick. It generates the bidirectional flows (source to destination and destination to source) of the captured network traffic. It can calculate a total number of 83 statistical features from the network traffic in a forward and backward direction. The output of the calculated statistical features is in a CSV file format with the 83 features (netflowmeter.ca). For the Transmission Control Protocol (TCP) flows, it is terminated by the FIN packet while the User Datagram Protocol (UDP) flows are terminated when there is a flow timeout which can be assigned by an individual or a network administrator (Foroushani and Heywood, 2015). The statistical features associated with the dataset that were extracted are flow duration, forward interarrival time (mean, min, max, std), backward interarrival time (mean, min, max, std), flow interarrival time (mean, min, max, std), active time (mean,

min, max, std), idle time (mean, min, max, std), flow bytes per second, flow packets per second, total forward packets, total backward packets, the total length of the forward packets, total length of the backward packet, forward packet length, backward packet length.

i. **Flow Duration:** This is the duration of a flow measured in a microsecond for both the forward and backward directions (netflowmeter.ca). Here, it is denoted as *FD*. This is calculated by subtracting the start time S_t of flow from the end time E_t of the flow.

$$\text{i.e. } FD = E_t - S_t \tag{4}$$

ii. **Forward Interarrival Time:** This is the time between two packets (e.g. packets a_1 and a_2) that are sent by the forwarding host (netflowmeter.ca). The mean time, minimum time, maximum time, and the time standard deviation are calculated for the forwarded packets. We denoted the forward interarrival time as *FIAT*. The time between the two packets is calculated by subtracting the time of the first packet a_{t1} from the time of the second packet a_{t2}. Therefore, $FIAT = a_{t2} - a_{t1}$. The minimum, maximum, mean, and standard deviation of *FIAT* is calculated as follows;

$$FIAT_{min} = min(a_{t1}, a_{t2}, \ldots, a_{tn}) \tag{5}$$

Where $FIAT_{min}$ is the minimum time between two packets sent in the forward direction and *n* is the time of the last packet

$$FIAT_{max} = max(a_{t1}, a_{t2}, \ldots, a_{tn}) \tag{6}$$

Where $FIAT_{max}$ is the maximum time between two packets sent in the forward direction and *n* is the time of the last packet

$$FIAT_{mean} = \frac{\sum_{i=1}^{n} a_{ti}}{n} \tag{7}$$

Where $FIAT_{mean}$ is the mean time between two packets sent in the forward direction, *n* is the number packets and a_{ti} are the packets times

$$FIAT_{std} = \sqrt{\frac{\sum(a_{ti} - FIAT_{mean})^2}{N}} \quad (8)$$

Where $FIAT_{std}$ is the standard deviation time between two packets sent in the forward direction, *N* is the number of packets and a_{ti} is the time of the packet.

iii. Backward Interarrival Time: This is the time between two packets (e.g. packets b_1 and b_2) that are sent backward by the receiving host as a reply to the sending host. The mean time, minimum time, maximum time, and the time standard deviation are calculated for the backward packets (netflowmeter.ca). We denoted the backward interarrival time as *BIAT*. The time between the two packets is calculated by subtracting the time of the first packet b_{t1} from the time of the second packet b_{t2}. Therefore, $BIAT = b_{t2} - b_{t1}$. The minimum, maximum, mean, and standard deviation of *BIAT* is calculated as follows;

$$BIAT_{min} = min(b_{t1}, b_{t2}, \ldots, b_{tn}) \quad (9)$$

Where $BIAT_{min}$ is the minimum time between two packets sent in the backward direction and *n* is the time of the last packet

$$BIAT_{max} = max(b_{t1}, b_{t2}, \ldots, b_{tn}) \quad (10)$$

Where $BIAT_{max}$ is the maximum time between two packets sent in the backward direction and *n* is the time of the last packet

$$BIAT_{mean} = \frac{\sum_{i=1}^{n} b_{ti}}{n} \quad (11)$$

Where $BIAT_{mean}$ is the mean time between two packets sent in the backward direction, *n* is the number packets and b_{ti} are the packets times

$$BIAT_{std} = \sqrt{\frac{\sum(b_{ti} - BIAT_{mean})^2}{N}} \quad (12)$$

Where $BIAT_{std}$ is the standard deviation time between two packets sent in the backward direction, *N* is the number of packets and b_{ti} is the time of the packet.

iv. Flow Interarrival Time: This is the time between two packets sent in a flow. E.g. packets a_1 *and* b_1. The mean time, minimum time, maximum time, and the time standard deviation are calculated for the packets (netflowmeter.ca). We denoted the flow interarrival time as *FlowIAT*. The time between the two packets is calculated by subtracting the time of the first packet a_{t1} from the time of the second packet b_{t1}. Therefore, $FlowIAT = b_{t1} - a_{t1}$. The minimum, maximum, mean, and standard deviation of *FlowIAT* is calculated as follows;

$$FlowIAT_{min} = min(a_{t1}, b_{t1}, \ldots, a_{tn}, b_{tn}) \quad (13)$$

Where $FlowIAT_{min}$ is the minimum time between two packets sent in the flow and *n* is the time of the last packet.

$$FlowIAT_{max} = max(a_{t1}, b_{t1}, \ldots, a_{tn}, b_{tn}) \quad (14)$$

Where $FlowIAT_{max}$ is the maximum time between two packets sent in the flow and *n* is the time of the last packet.

$$FlowIAT_{mean} = \frac{\sum_{i=1}^{n} a_{ti} b_{ti}}{n} \quad (15)$$

Where $FlowIAT_{mean}$ is the mean time between two packets sent in the flow, n is the number packets and $a_{ti}b_{ti}$ is the packets times.

$$FlowIAT_{std} = \sqrt{\frac{\sum(a_{ti}b_{ti} - FlowIAT_{mean})^2}{N}} \quad (16)$$

Where $FlowIAT_{std}$ is the standard deviation time between two packets sent in the flow, N is the number of packets and $a_{ti}b_{ti}$ is the time of the packet.

v. **Active Time:** This is the amount of time a flow (bidirectional) was active before going idle or terminated by the FIN packet or when there is a flow time out. The mean time, minimum time, maximum time, and the time standard deviation are calculated for the flow. We denote it as *AT*.

vi. **Idle Time:** This is the amount of time a flow (bidirectional) was idle before becoming active. The mean time, minimum time, maximum time, and the time standard deviation are calculated for the flow. We denote it as *IT*.

vii. **Flow Bytes/Second:** This is the number of flow bytes per second. We denote it as *FBps*.

viii. **Flow Packets/Second:** This is the number of flow packets per second. We denoted this as *FPps*.

ix. **Total Forward Packets:** This is the total packets in the forward direction sent by the sending Host. We denoted this as *TFP*. This is calculated as

$$TFP = (a_1 + a_2 + \ldots, a_n) \quad (17)$$

Where n is the last packet

x. **Total Backward Packets:** This is the total packets in the backward direction sent by the receiving Host as a reply for the data received from the sending Host. We denoted this as *TBP*.

$$TBP = (b_1 + b_2 + \ldots, b_n) \quad (18)$$

Where n is the last packet

xi. **Total Length of Forward Packet:** This is the total size of the packets in the forward direction of a flow sent by the sending Host. We denoted this as *TLFP*.

xii. **Total Length of Backward Packet:** This is the total size of the packet in the backward direction of a flow sent by the receiving Host. We denoted this as *TLBP*.

xiii. **Forward Packet Length:** This is the size of the packet sent in the forward direction of flow. The mean time, minimum time, maximum time, and the time standard deviation are calculated for the packets. We denoted this as *FPL*.

$$FPL_{min} = min(a_{l1}, a_{l2}, \ldots, a_{ln}) \quad (19)$$

Where FPL_{min} is the minimum size of the packet in the forward direction and n is the size of the last packet.

$$FPL_{max} = max(a_{l1}, a_{l2}, \ldots, a_{ln}) \quad (20)$$

Where FPL_{max} is the maximum size of the packet in the forward direction and n is the size of the last packet.

$$FPL_{mean} = \frac{\sum_{i=1}^{n} a_{li}}{n} \quad (21)$$

Where FPL_{mean} is the mean size of the packet in the forward direction, n is the number packets and a_{li} is the packets sizes.

$$FPL_{std} = \sqrt{\frac{\sum(a_{li} - FPL_{mean})^2}{N}} \quad (22)$$

Where FPL_{std} is the standard deviation size of the packet in the forward direction, N is the number of packets and a_{li} is the size of the packet.

xiv. **Backward Packet Length:** This is the size of the packets sent in the backward direction of flow. The mean time, minimum time, maximum time, and the time standard deviation are calculated for the packets. We denoted this as *BPL*.

$$BPL_{min} = min(b_{l1}, b_{l2}, \ldots, b_{ln}) \quad (23)$$

Where BPL_{min} is the minimum size of the packet in the forward direction and *n* is the size of the last packet

$$BPL_{max} = max(b_{l1}, b_{l2}, \ldots, b_{ln}) \quad (24)$$

Where BPL_{max} is the maximum size of the packet in the forward direction and *n* is the size of the last packets

$$BPL_{mean} = \frac{\sum_{i=1}^{n} b_{li}}{n} \quad (25)$$

Where BPL_{mean} is the mean size of the packet in the forward direction, *n* is the number packets and b_{li} is the packets sizes.

$$BPL_{std} = \sqrt{\frac{\sum(b_{li} - BPL_{mean})^2}{N}} \quad (26)$$

Where BPL_{std} is the standard deviation size of the packet in the forward direction, N is the number of packets and b_{li} is the size of the packet.

6.2 The Building of The Proposed Ensemble Model

The proposed model consists of the outputs from the different base classification models (different Machine Learning and Deep Learning techniques) in the first stage learning algorithms that were merged and fed into a new classification model (Random Forest Classifier) in the second stage learning algorithm for training and classification of the encrypted network traffic using the test sets. To build this proposed system, we utilized the already preprocessed dataset which was done using the MinMax Scaler, label encoder, and one_Hot encoder to build the various machine learning technique (Random Forest and Decision Tree) and deep learning techniques (Capsule Neural Networks and Artificial Neural Networks) and boosted the CapsNet, ANN, RF and DT with XGBoost by setting the base estimators as the weak learners while the DT was also boosted with AdaBoost.

Before preprocessing the public dataset, we balanced it using the SMOTETomek from the imblearn library. It is a combination of undersampling and oversampling methods. It balances the instances of the target class such that there are little differences between them. This prevents the model from overfitting or underfitting. Scaling the dataset to a machine-readable form means to generally change the values without changing the distribution shape. This can help the distributed instances among the independent variables have a similar form of values for the algorithm it will be fed into. For MinMax Scaler, it subtracts the minimum value in the training dataset's independent variable and then divides it by the range. The range is the difference between the actual dataset maximum value of the independent variable and the actual dataset minimum value of the independent variable. The MinMax Scaler returns a default range between 0 and 1 for the independent variables without changing the information originally embedded in the actual dataset. The MinMax Scaler (scikit-learn, 2019) is given as

$$scaler = \frac{(X - X_{min})}{(X_{max} - X_{min})} \tag{27}$$

Where $scaler$ is the new scaled values, X is the actual cell values of the training set, X_{min} is the minimum value of the independent variable and X_{max} is the maximum value of the independent variable. For the label encoding, it is used to encode labels in the categorical features (in this case is the target variables which are the traffic categories) with a value between 0 and (number of classes – 1) i.e. 0 < 1 < 2 < 3.... < n (where n is the last category). It also assigns the same value to repeated labels. The problem with the label encoding is that the machine will see the numbers as hierarchical or in some kind of order, making the machine read a different meaning to the categorical features thereby generating different patterns compared to the original patterns it should have discovered. To overcome this, we employ the well-known One_Hot encoding on the categorical features. As it is widely known, One_Hot encoding takes the independent variable with the categorical data that has been label encoded. It then splits the column into multiple columns. The numbers are then replaced with 0s and 1s (identifying true or false) depending on the column that has what value. This is performed for flexibility during analysis and as well as recognizing the right hidden patterns in the dataset (scikit-learn, 2019).

After the preprocessing stage, it is required that we split the preprocessed data into a 70% train set and a 30% test set. This will enable us to fit our model on the train set to make predictions on the test set (unseen data). This is also instrumental in avoiding overfitting and underfitting our model.

The split dataset needs to be fed into a model for analysis, so we designed CapsNet+XGBoost, ANN+XGBoost, RF+XGBoost, DT+AdaBoost, and DT+XGBoost classifiers to perform analysis on the dataset.

The ANN model contains the input layer $(x_1, x_2, x_3, \ldots x_n)$, the hidden layers $(h_1, h_2, \ldots, h_n)$ and the output layer $(y_1, y_2, \ldots, y_n)$ which are the class variables. The layers are made of nodes that are connected and weights are assigned to those connections. The connections are called linkages. This can be denoted as

$$W(x_1, h_1) \tag{28}$$

where W is the weight of the linkage between the input layer node x_1 and the hidden layer node h_1.

In the framework of the ANN (Neural Networks) classification algorithm, random weights are assigned to all the linkages, using the inputs and the linkage between the input node and hidden layer node to find the activation rate of the hidden layers. This is denoted as

$$h_1 = g(W_1x_1 + W_2x_2, \dots W_nx_n) \quad (29)$$

Where g is the activation function, W_1 is the weight of the first input node, x_1 is the first input node and n is the number of inputs

Next, using the activation rate of the hidden nodes and the linkages to the output to find the activation rate of the output nodes which is the hypothesis function. This is given as

$$h_w(x) = y_1 = g(W_1h_1 + W_2h_2, \dots W_nh_n) \quad (30)$$

Where $h_w(x)$ is the hypothesis function of the output layer, g is the activation function, W_1 is the weight of the first hidden layer node, h_1 is the first hidden layer node and n is the number of inputs

Next is to choose the activation function. The activation function determines the output of the deep learning model and whether a given node should be activated or not, this is based on whether the node's input is relevant during the model's prediction. It also helps in the normalization of the output layer nodes to a range between -1 and 1 or between 0 and 1. Here, we used the ReLu (Rectified Linear Unit) activation function because of its computational efficiency by allowing the neural network to converge faster and also allows for backpropagation. According to (Sagar, 2017), it is denoted as

$$R(y) = max(0, y) \quad (31)$$

It is zero for all the negative values and linear for all positive values.

After applying the ReLu function on the nodes in each layer, we applied the Softmax function to the output scores. It squashes the vector into a range of 0 to 1. This can be interpreted as the class probabilities since we are trying to solve a multiclass problem. According to (Ognjanovski, 2019), it is given as

$$f(s)_i = \frac{e^{s_i}}{\sum_j^y e^{s_j}} \quad (32)$$

Where $f(s)_i$ is the Softmax function computed for each given class s_i, s_j are the scores for each class in y. After applying the SoftMax function, we then compiled the model using the categorical_crossentropy loss function. It is the SoftMax activation function plus the cross-entropy loss function. It is used to train the model to output the probability of each class in the target class. It is given as

$$CE = -log\left(\frac{e^{sp}}{\sum_j^y e^{sj}}\right) \quad (33)$$

Where s_p is the ANN score for the positive class

Then, we applied the XGBoost technique on the ANN model to boost the model. According to (Chen and Guestrin, 2016) the formula for XGBoost is given as

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i{}^{(t-1)} + f_t(X_i)\right) + \Omega(f_t) \quad (34)$$

The capsule neural network contains the input layer, the convolution layer, the reshape layer, the primary capsule layer, the Maxpooling layer, the fully connected layer, and the output. The reshape layer reshapes the output features from the convolution layers into vectors of any desired dimension related to the input dimension of the dataset. The squash function is used to perform the reshaping and according to (Hinton et. al., 2017) it is denoted as

$$\mathbf{v}_j = ||\mathbf{S}_j||^2 1 + ||\mathbf{S}_j||^2 \mathbf{S}_j||\%S_j|| \quad (35)$$

Where $\mathbf{v}_j$ is the output vector of the capsule and $\mathbf{S}_j$ is the input.

Once the CapsNet model takes an input, it transfers it to the convolution layer where the ReLu activation function is applied on the data to maintain its direction while normalizing its length, then the data is then split into groups before it is reshaped into the primary capsule layer using the squash function. The dynamic routing is implemented on the primary capsule layer. The routing by agreement is a more advanced method of connecting the neurons compared to that of max pooling which can lose most outstanding connections (Hinton et. al., 2017).

For max pooling function, it is used to down-sample the input data by reducing its dimensionality and make assumptions on the features contained in the feature bin. It is also instrumental in preventing overfitting of the model. In other words, it calculates the maximum value for each bin of the feature map. While average pooling calculates its average value. After designing the CapsNet model, it was boosted using the XGBoost technique.

After designing the ANN+XGBoost model and the CapsNet+XGBoost, we designed the RF+XGBoost classification model. The Random Forest model comprises many or sets of decision trees. It uses the random sampling concept of training the data points when building trees and splitting the nodes, it uses the random subsets of features considered (Will, 2018). The formula according to (Schott, 2019) is given as

$$Gini = 1 - \sum_{i=1}^{C}(p_i)^2 \qquad (36)$$

The Gini is used to decide how nodes are generated. Where p_i is the relative frequency of the observed class and C is the number of classes. After which we boosted the RF model using the XGBoost technique.

We then designed the Decision Tree classification model and boosted it with both AdaBoost and XGBoost. AdaBoost can only use trees as the base model.

Next, we created a stacking function to perform the stacking technique on the base estimators (CapsNet+XGBoost, ANN+XGBoost, RF+XGBoost, DT+AdaBoost, and DT+XGBoost) while using k-fold cross-validation of 10 by importing the stacking classifier and stratified kfold from sklearn library. The outputs were merged and fed into a new classification model (Random Forest Classifier) for training. To determine the performance of the model based on its accuracy, we tested the model by making predictions on the test set. The results of the experiment are shown in table 6.1 below.

Algorithm 2: The Proposed Encrypted Network Traffic Classification

$\boldsymbol{x = number\ of\ features\ multiplied\ by\ number\ of\ samples}\ (\boldsymbol{m * n})$
$\boldsymbol{y = network\ traffic\ category\ classes}\ (\boldsymbol{y_1, y_2, \ldots, y_n})$

Input: Network traffic dataset $\boldsymbol{D} = \bigcup_{n=1}^{m} \boldsymbol{x_{nm} y_n}$;
First stage learning algorithms $\boldsymbol{L_1, \ldots, L_N}$;
Second stage learning algorithm $\boldsymbol{L}$.

Process:
Split the dataset into train and test set, $\boldsymbol{S} = \{(\boldsymbol{x_1, y_1}), (\boldsymbol{x_2, y_2}), \ldots, (\boldsymbol{x_n, y_n})\}$;
Based on the train set $\boldsymbol{S}$, use the first stage individual learning algorithms $\boldsymbol{L_1, \ldots, L_N}$ to train the classifiers $\{\boldsymbol{b_1, b_2, \ldots, b_n}\}$ and test the classifiers with the test set $\boldsymbol{s}$
$\boldsymbol{for\ first\ stage\ classification, do}$
  $\boldsymbol{for\ i = 1, 2, 3, \ldots, n}$ (where $\boldsymbol{i}$ is the number of iteration)
    $\boldsymbol{b_{1i} = L_1(S)}$ (where $\boldsymbol{S}$ is the 70% train sets used in training the classifiers)
    $\boldsymbol{b_{2i} = L_2(S)}$
    $\boldsymbol{b_{3i} = L_3(S)}$
    $\boldsymbol{b_{4i} = L_4(S)}$
    $\boldsymbol{b_{5i} = L_5(S)}$
  $\boldsymbol{end}$;
  Generate new train set from the output of the first stage classifiers
  $\boldsymbol{S' = 0}$
  Classify the test set $\boldsymbol{s}$ using the individual classifiers
  $\boldsymbol{for\ i = 1, 2, 3, \ldots, n}$
    $\boldsymbol{z_1 = b_{1i}(s)}$ (where $\boldsymbol{s}$ is the 30% test sets)
    $\boldsymbol{z_2 = b_{2i}(s)}$
    $\boldsymbol{z_3 = b_{3i}(s)}$
    $\boldsymbol{z_4 = b_{4i}(s)}$
    $\boldsymbol{z_5 = b_{5i}(s)}$
  $\boldsymbol{end}$;
  $\boldsymbol{S' = S' \cup \{z_1, z_2, z_3, z_4\}}$; (merge the outputs from the base classifiers)
$\boldsymbol{end}$;
Train the second stage classifier $\boldsymbol{b'}$ by applying the learning algorithm $\boldsymbol{L}$ to the new train set $\boldsymbol{S'}$
$\boldsymbol{for\ second\ stage\ classification, do}$
  $\boldsymbol{b' = L(S')}$
  Classify the test set $\boldsymbol{s}$ using the second stage classifier
  $\boldsymbol{z' = b'(s)}$
$\boldsymbol{end}$;

Algorithm 2 is the detailed pseudo-code of our proposed solution. The encrypted network traffic dataset D contains the independent variables, instances (which are network flows), and the target class (which comprises the network categories). Data preprocessing was performed on the dataset and then split into a 70% train set S and a 30% test set s. We then designed and initialized the first stage classifiers $\{b_1, b_2, \ldots, b_n\}$ using the individual learning algorithms $L_1, \ldots, L_N$ (CapsNet+XGBoost, ANN+XGBoost, RF+XGBoost, DT+AdaBoost, and DT+XGBoost learning algorithms). The learning algorithms in the first stage have no

particular order because the goal is to get the output from the different learning algorithms and merge them to form a new train set that will be used to train the second stage learning algorithm. Next, we trained the first stage classifiers using the train set and tested the models' accuracy using the test set. The outputs were merged to form a new train set S' and fed into the second stage classifier b' designed using the second stage learning algorithm L (Random Forest learning algorithm). It was then used to classify the test set to obtain the classification results of the encrypted network traffic data. We used the public dataset and Solana Networks dataset to test the performance of our proposed system. The results obtained for each of the network traffic datasets are shown below.

Table 6.1 Proposed Ensemble Model Experiment Result (public dataset)

Measurement metric	Result
Overall Accuracy	96%
Weighted Precision	96%
Weighted Recall	96%
Weighted F1 score	96%

Table 6.2 Proposed Ensemble Model Experiment Result for each category (public dataset)

Number	Label	Precision	Recall	F1-score
0	BROWSING	95%	96%	95%
1	CHAT	97%	97%	97%
2	FT	88%	96%	92%
3	EMAIL	98%	97%	98%
4	P2P	99%	98%	99%
5	STREAMING	96%	89%	93%
6	VOIP	100%	99%	99%

Table 6.3 Proposed Ensemble Model Experiment Confusion Matrix (public dataset)

True Label \ Predicted Label	BROWSING	CHAT	FT	EMAIL	P2P	STREAMING	VOIP
BROWSING	793	18	5	0	2	12	0
CHAT	15	843	4	2	1	2	0
FT	6	4	799	10	1	12	1
EMAIL	1	0	20	785	0	0	1
P2P	1	6	5	0	831	2	0
STREAMING	13	1	74	5	2	778	1
VOIP	2	1	4	1	1	1	835

Table 6.4 Proposed Ensemble Model Experiment Result (Solana Networks dataset)

Measurement metric	Result
Overall Accuracy	90%
Weighted Precision	90%
Weighted Recall	90%
Weighted F1 score	90%

Table 6.5 Proposed Ensemble Model Experiment Result for each category (Solana Networks dataset)

Number	Label	Precision	Recall	F1-score
0	AudioStream	77%	83%	80%
1	Browsing	85%	86%	85%
2	Chat	88%	94%	91%
3	FT	97%	98%	98%
4	P2P	98%	95%	97%
5	VOIP	95%	94%	95%
6	VideoStream	91%	79%	84%

Table 6.6 Proposed Ensemble Model Experiment Confusion Matrix (Solana Networks dataset)

True Label \ Predicted Label	AudioStream	Browsing	Chat	FT	P2P	VOIP	VideoStream
AudioStream	5355	455	324	28	21	41	218
Browsing	505	5587	140	39	33	70	143
Chat	171	29	6214	5	7	120	50
FT	31	40	18	6520	47	16	7
P2P	65	96	15	50	6275	19	55
VOIP	108	116	99	25	18	6213	44
VideoStream	765	286	246	21	16	47	5132

6.3 Design Discussions

The reason we selected the CapsNet+XGBoost, ANN+XGBoost, RF+XGBoost, DT+AdaBoost, and DT+XGBoost as the base classification models in building the proposed ensemble learning system is that they have the best performance accuracies from the experiments performed in chapter 5. Capsule Neural Network is one of the latest architectures in Deep Learning (Neural Networks) and it is an advanced approach to Convolution Neural Networks to overcome the drawback of pooling layers (Kshitizrimal, 2019). To classify data using the CNN classification model, it uses the convolution layers and the pooling layers. The pooling layer is used to reduce the dimensionality of the data thereby achieving spatial invariance (i.e. no matter where the data is, it identifies the features and patterns and classifies them). During Max pooling, it tends to lose some information (hidden patterns associated with the data) that are useful in the classification process thereby making it difficult to recognize patterns or features that are associated with the data. CapsNet was introduced to try reconstructing the positional information of the data using various advanced techniques that are not 100 percent accurate. Also, the reconstruction process is time-consuming.

Boosting techniques are sets of classifiers that use the low accurate classifier as the base estimator to create extremely accurate classifiers. They are less affected by the overfitting problem. We chose XGBoost

classifier as the boosting technique because it uses a regularised model formalization which helps control the model's complexity and helps the model to avoid overfitting and giving the best performance compared to the other boosting techniques such as gradient boosting, AdaBoost. XGBoost classifier tracks the model's misclassification during the prediction or classification process and classifies them into the appropriate classes thereby forcing out the best performance of the model. Gradient Boosting technique, on the other hand, makes use of the base classification model's loss function to minimize the error of the model while AdaBoost helps the model to achieve a less bias and variance (Elsinghorst and Shirin, 2018). CatBoost, which was introduced in 2017 by Yandex researchers, gives indices of categorical columns contained in dataset flexibility during classification and it is encoded using one_hot encoder. It is capable of handling datasets with categorical features for classifications. So, since the datasets used in our classification experiments conducted above have no categorical independent variable(s), we decided not to make use of the CatBoost technique.

The stacking technique of the proposed ensemble learning model is an ensemble model technique where the new classification (Random Forest) model is trained using the combined predictions of the five base classification models (CapsNet+XGBoost, ANN+XGBoost, RF+XGBoost, DT+AdaBoost, and DT+XGBoost) and then predictions are made on the test set. While the voting classifier can combine different machine learning classifiers and then a vote is performed on the predicted class labels of each classification model. Utilizing the hard voting in the voting technique of an ensemble learning system makes the model to use most of the classification models to determine what the classification result could be. Soft voting computes the percentage weight of each classifier and weights are difficult to find if it is based on guessing technique (Singh, 2018). Since there is no stacking classifier in sklearn ensemble methods that can enable us to combine machine learning algorithms with that of deep learning algorithms for our analysis, we decided to create a stacking function to do so. For example, if you have three classification models

namely model1, model2, and model3. After performing a prediction analysis using these models, then on outputting the results of the analysis, model1 says 1, model2 says 0 and model3 says 1, the voting classifier will be used to combine the results of the models and hard voting will be performed on the models' results whereby it chooses the majority voting which is 1 and outputs it as the final result of the prediction analysis. While in the stacking technique, the results of the three models will be merged to form a new train set for a new classification model. The new classification model will take the new train set (which is the predicted results of the three models) as input to train the new classification model and predictions will be done using the test set thereby producing a strong model that improves the prediction. We chose Random Forest Classifier as the new classification model which is the last model of the proposed ensemble learning system because they are a modification of decision tree model that splits on a subset of features during each split (i.e. at each split of the tree during classification, it considers a small subset of the features of the dataset than all the features of the dataset). It is one of the best algorithms for multi-classification due to its impressive nature to handle datasets with any kind of features and target class. It as well handles outliers and unbalanced datasets by trying to minimize the error rate which results in the model having a low bias and a moderate variance. Irrespective of the aforementioned reasons, Random Forest Classifier has a good training speed meaning that it is faster to train compared to that of the Decision Tree Classifier (Kho, 2018).

Generally, an ensemble learning technique improves the model’s results by combining different classification models. It also produces better prediction accuracy compared to that of a single model.

Chapter 7: Evaluation

In this chapter, we evaluate our proposed system model designed to classify encrypted network traffic data. Here, we analyzed the functionality and time complexity of our proposed model algorithms to review and acquire knowledge on its feasibility. We also analyzed our model to justify its effectiveness by performing some experiments with the test dataset. Also, the performance of the algorithm used was analyzed and compared with other studies. To experiment on the dataset, it was split into a 70% training set and 30% test (to avoid overfitting and underfitting our model), and a number of k for k-fold cross-validation of 10 was chosen for the experiment. K-fold cross-validation is a resampling method that is used to evaluate a model's performance on an unseen data sample. The value 10 for k-fold cross-validation was chosen because it has been found that the value 10 results to a good performance model with low bias and modest variance during experimentation (James et. al., 2013 and Kuhn and Johnson, 2013). Lastly, the experimental evaluation of our model was evaluated in terms of overall accuracy, precision, recall, F1-score, specificity, weighted precision, weighted recall, weighted f1-score, ROC (Receiver Operating Characteristics) curve and confusion matrix.

7.1 Functional Evaluation Using the Public Dataset

Here, we evaluate the performance of our proposed system built using the UNB public network traffic dataset in terms of accuracy, precision, recall, F1-score, specificity, ROC (Receiver Operating Characteristics) curve, and confusion matrix.

a. **Accuracy:** This is a measure of our classification model that shows the proportionate number of times that the model is correct in its classification of encrypted network traffic data when it is applied to any similar network traffic dataset. It is calculated using the formula

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Where TP is the True Positive, TN is the True Negative, FP is the False Positive and FN is the False Negative. $TP + FP + TN + FN$ is also the total number or count of the test set.

b. Precision: This is the average probability of relevant retrieval. It can be calculated using the formula; $precision = \frac{TP}{TP+FP}$.

c. Recall: This is the percentage of the total data that are correctly classified. It is classified using the formula; $recall = \frac{TP}{TP+FN}$.

d. F1-score: This is used to seek a balance between precision and recall (Shung, 2018). It is classified using the formula; $F1 - score = 2 * \frac{Precision*Recall}{Precision + Recall}$.

e. Specificity: This is used to evaluate the model's ability to predict true negatives of each available class (i.e. if network traffic does not belong to a specific class). It is classified using the formula; $specificity = \frac{TN}{TN+FP}$.

f. Weighted Precision: This is the weighted mean of precision with weights equal to the class probability or the number of samples from that class. It is used in the multiclass evaluation. It can be calculated using the formula

$$weighted\ precision = \left(\frac{(number\ of\ class_1 sample * precision\ of\ class_1) + \ldots (number\ of\ class_n sample * precision\ of\ class_n)}{Total\ number\ of\ test\ samples}\right)$$

Where n is the class number. The weighted precision is one of the metrics used to measure the performance of a multiclassification model.

g. Weighted Recall: This is the weighted mean of recall with weights equal to the class probability or the number of samples from that class. It is calculated as

$$weighted\ recall = \left(\frac{(number\ of\ class_1\ sample * recall\ of\ class_1) + \ldots (number\ of\ class_n\ sample * recall\ of\ class_n)}{Total\ number\ of\ test\ samples}\right)$$

Where n is the class number. The weighted recall is one of the metrics used to measure the performance of a multiclassification model.

h. Weighted F1-score: This is the weighted mean of F1-score with weights equal to the class probability or the number of samples from that class. It is calculated as

$$weighted\ f1-score = \left(\frac{(number\ of\ class_1\ sample * f1-score\ of\ class_1) + \ldots (number\ of\ class_n\ sample * f1-score\ of\ class_n)}{Total\ number\ of\ test\ samples}\right)$$

Where n is the class number. The weighted f1-score is among the metrics used to measure the performance of a multiclassification model.

i. ROC curve: This is the plot of the False Positives (FP) rate versus the True Positive (TP) rate. Where $TP\ rate = sensitivity = Recall$ and $FP\ rate = specificity$. The ROC curve for the proposed model is shown in fig. 7.1 below.

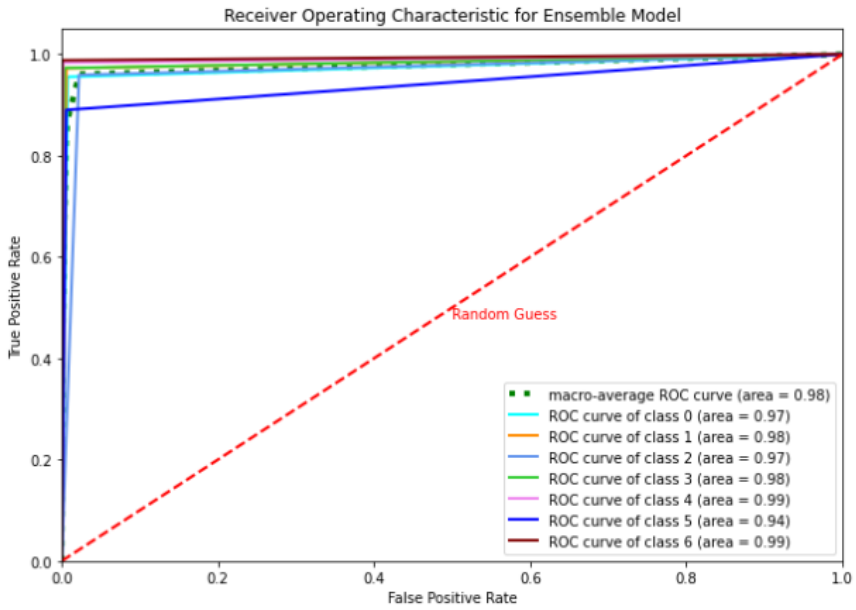


Figure 7.1 *Proposed System ROC curves for the network traffic categories (public dataset)*

j. Confusion Matrix: This is the performance measure for the proposed model's class classification. It is shown in figure 7.2 below.

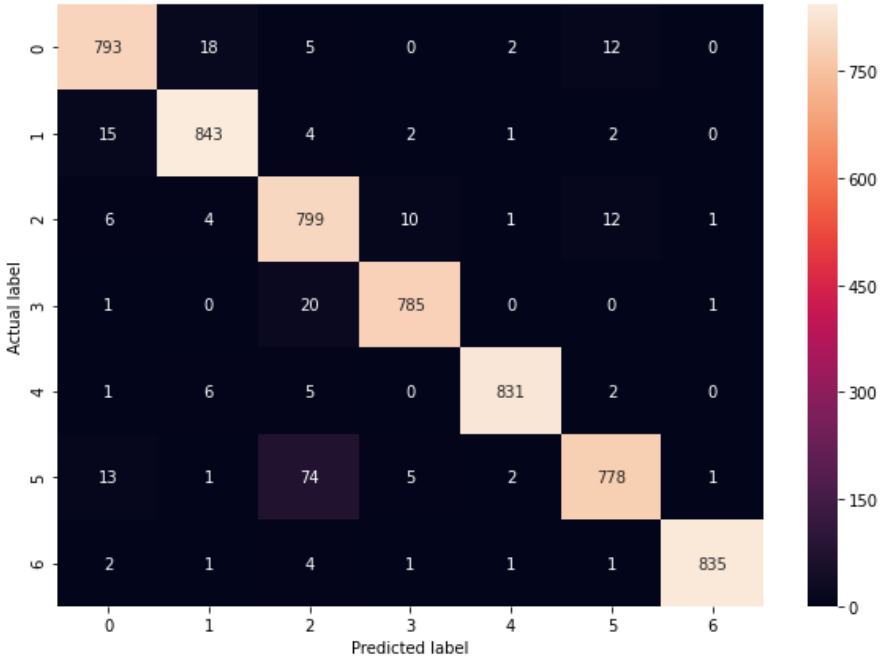


Figure 7.2 *Proposed System confusion matrix for the network traffic categories (public dataset)*

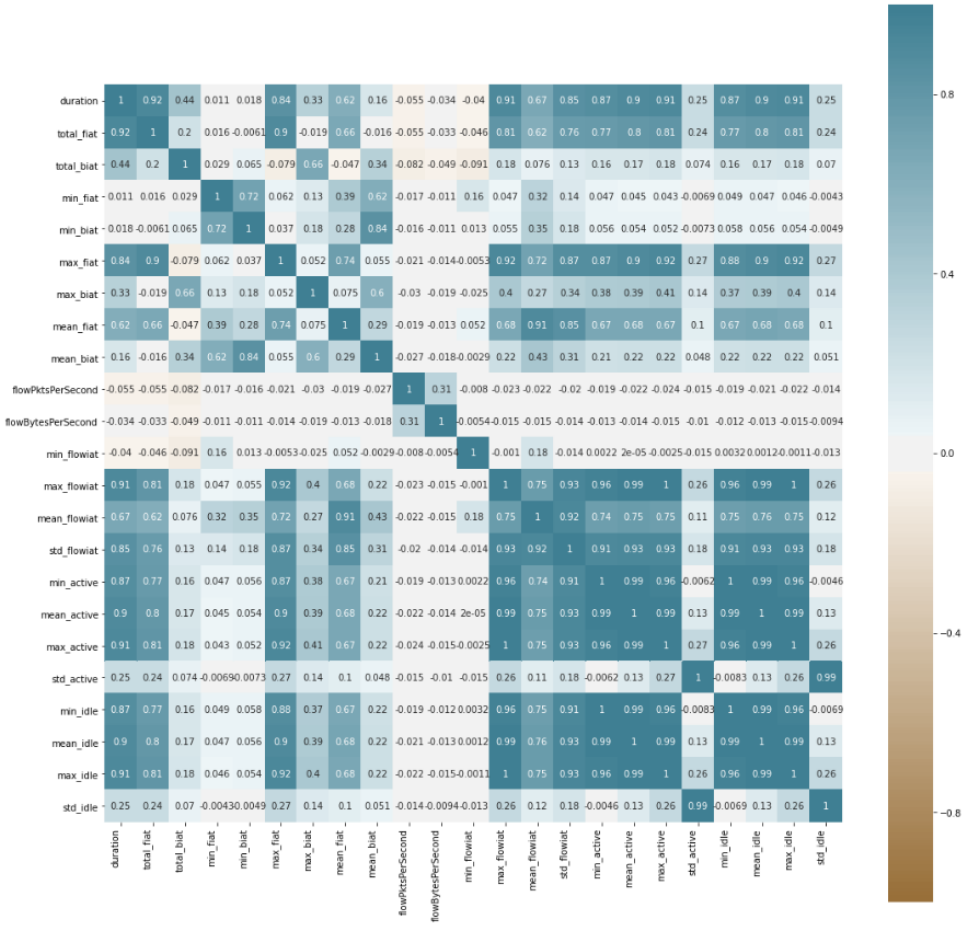
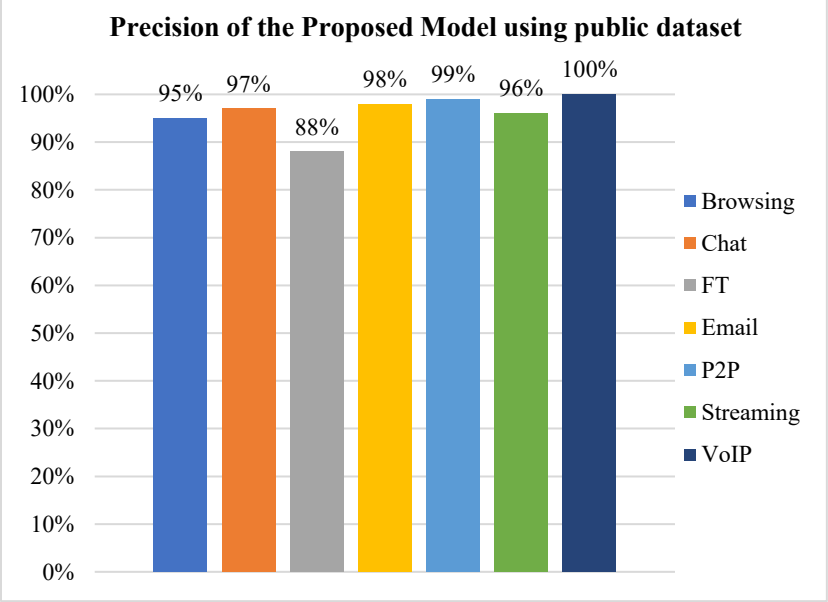
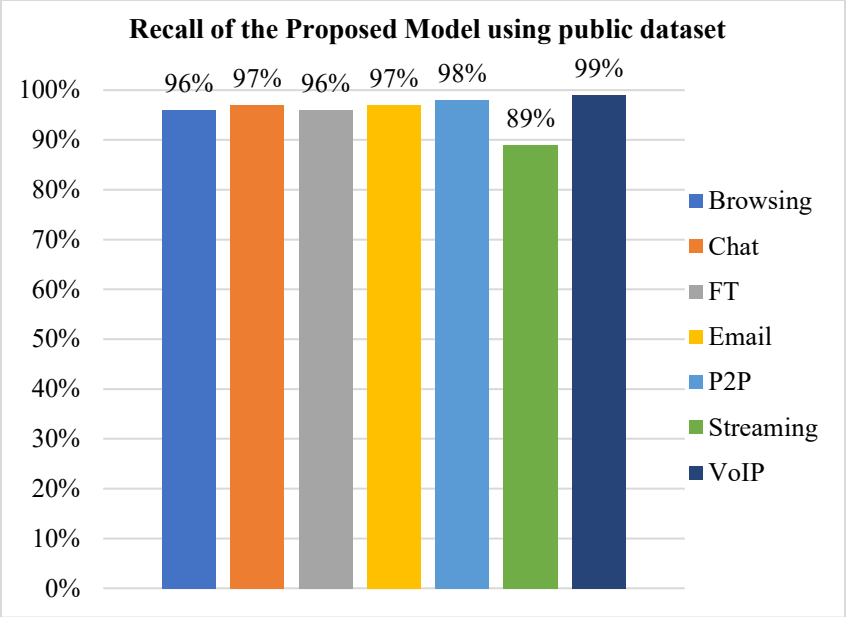


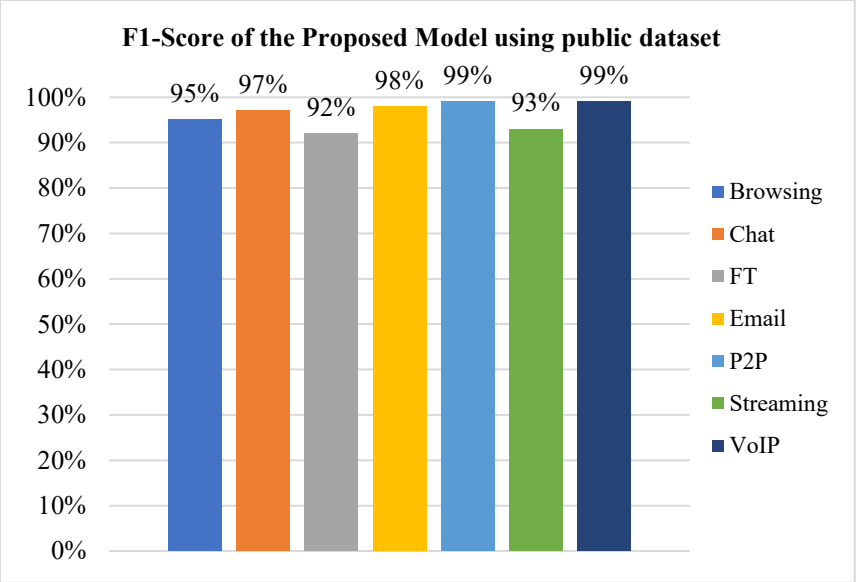
Figure 7.3 *Proposed System correlation matrix (public dataset)*

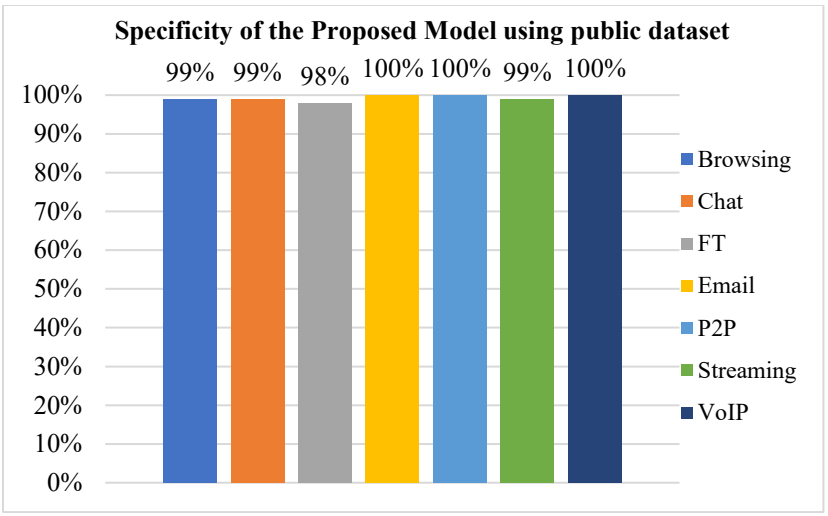
Table 7.1 Proposed Ensemble Model Functional Result for each category (Public dataset)

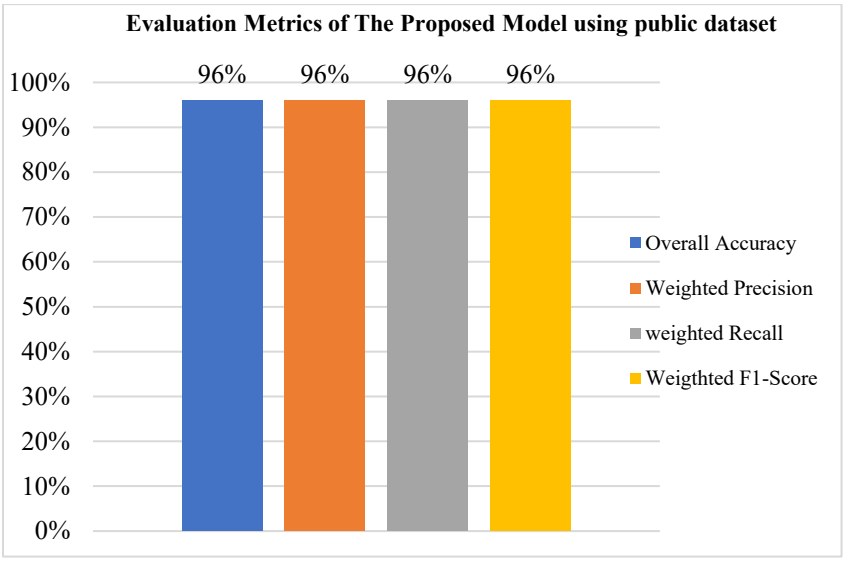
Number	Label	Precision	Recall	F1-score	Specificity
0	BROWSING	95%	96%	95%	99%
1	CHAT	97%	97%	97%	99%
2	FT	88%	96%	92%	98%
3	EMAIL	98%	97%	98%	100%
4	P2P	99%	98%	99%	100%
5	STREAMING	96%	89%	93%	99%
6	VOIP	100%	99%	99%	100%
Overall Accuracy					**96%**
Weighted Precision					**96%**
Weighted Recall					**96%**
Weighted F1-Score					**96%**




Recall of the Proposed Model using public dataset
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
96%
97%
96%
97%
98%
89%
99%
Browsing
Chat
FT
Email
P2P
Streaming
VoIP


F1-Score of the Proposed Model using public dataset
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
95%
97%
92%
98%
99%
93%
99%
Browsing
Chat
FT
Email
P2P
Streaming
VoIP





7.2 Functional Evaluation Using the Solana Networks Dataset

Here, we evaluate the performance of our proposed system tested on the Solana Networks network traffic dataset in terms of accuracy, precision, recall, F1-score, weighted precision, weighted recall, weighted f1-score, specificity, ROC (Receiver Operating Characteristics) curve, and confusion matrix.

a. ROC curve: This is the plot of the False Positives (FP) rate versus the True Positive (TP) rate. Where $TP\ rate = sensitivity = Recall$ and $FP\ rate = specificity$. The ROC curve for the proposed model is shown in fig. 7.4 below.

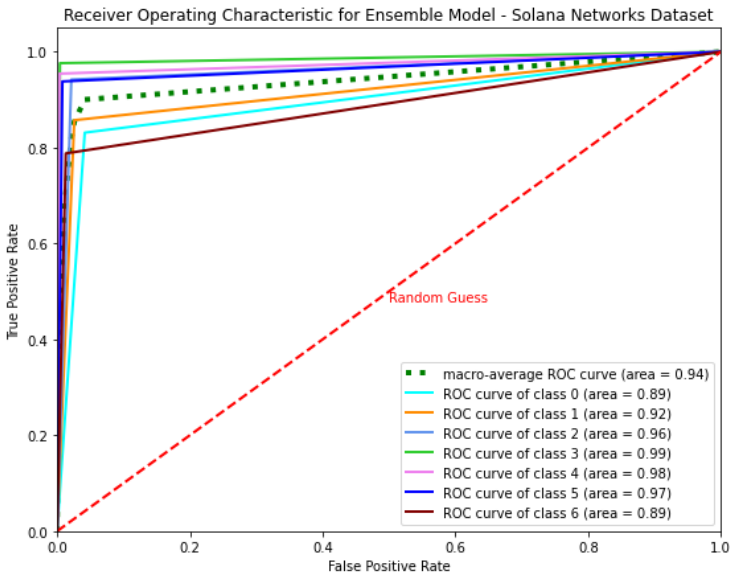


Figure 7.4 *Proposed System ROC curves for the network traffic categories (Solana Networks dataset)*

b. Confusion Matrix: This is the performance measure for the proposed model's class classification. It is shown in figure 7.5 below.

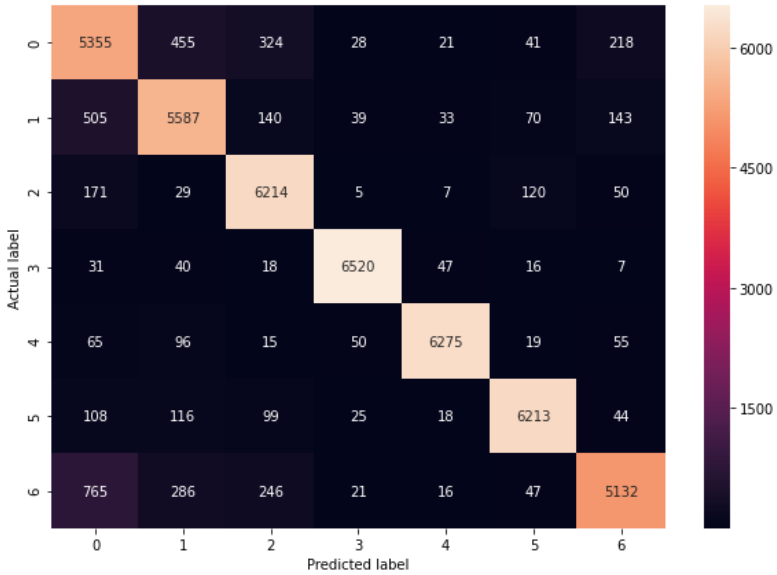


Figure 7.5 *Proposed System confusion matrix for the network traffic categories (Solana Networks Dataset)*

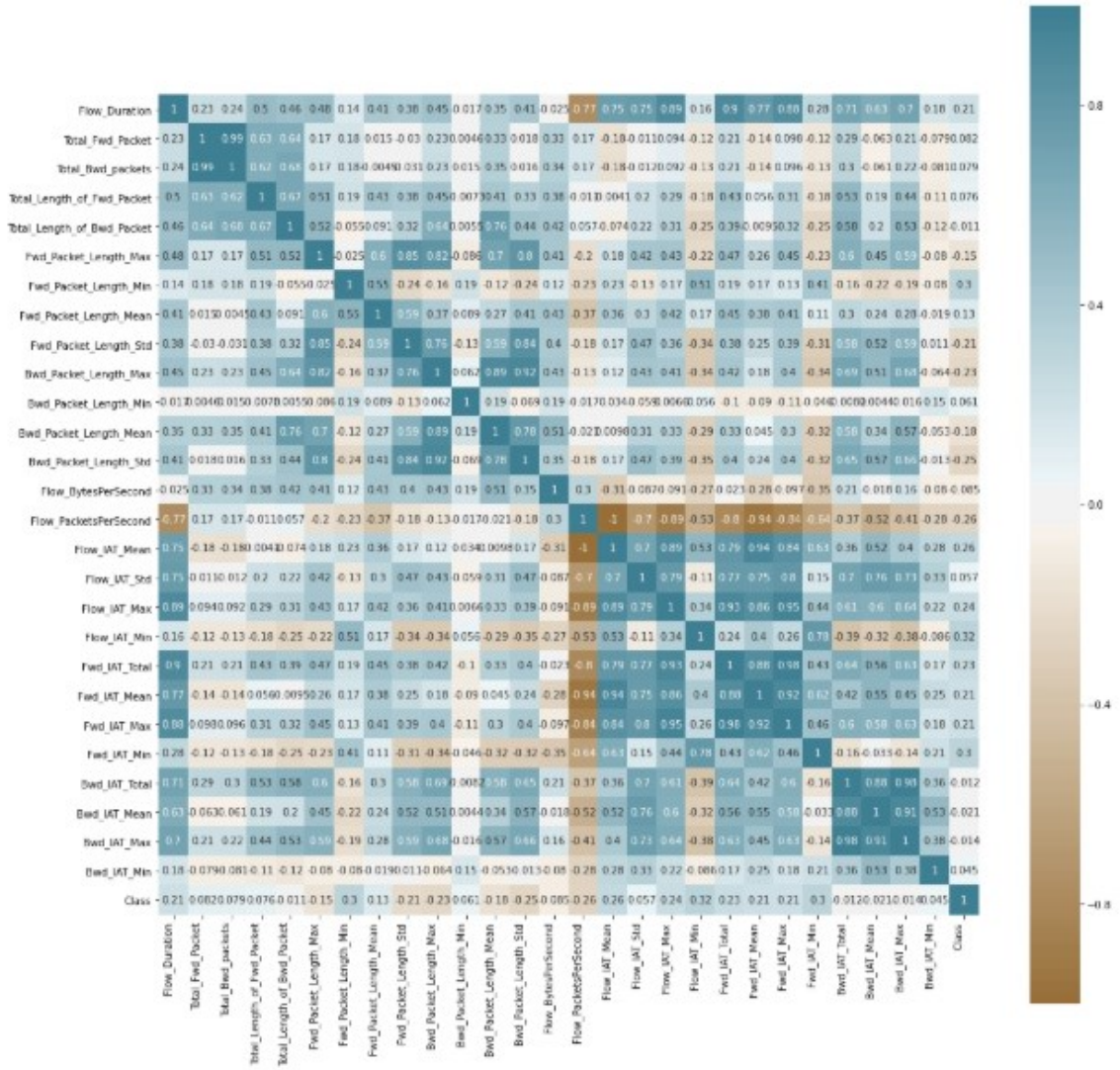
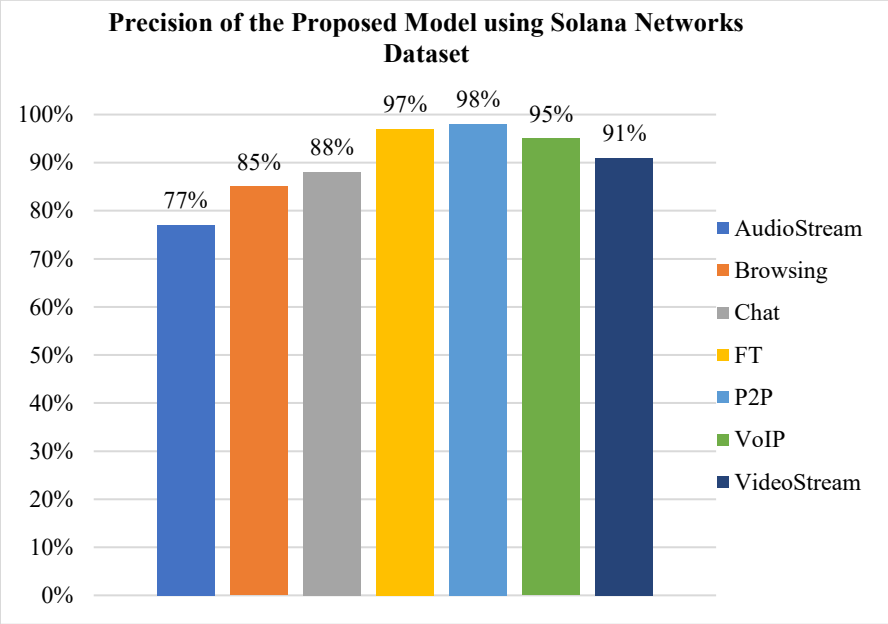
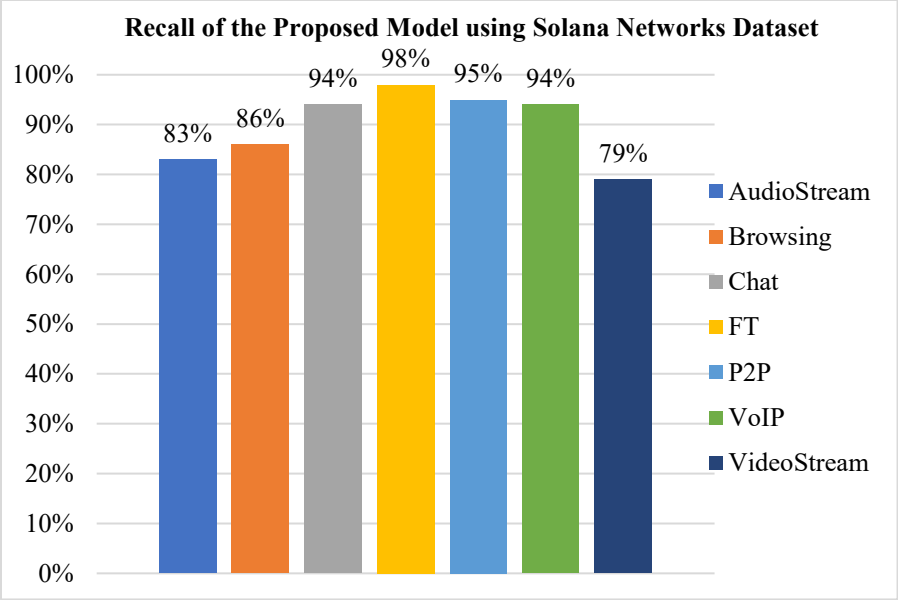


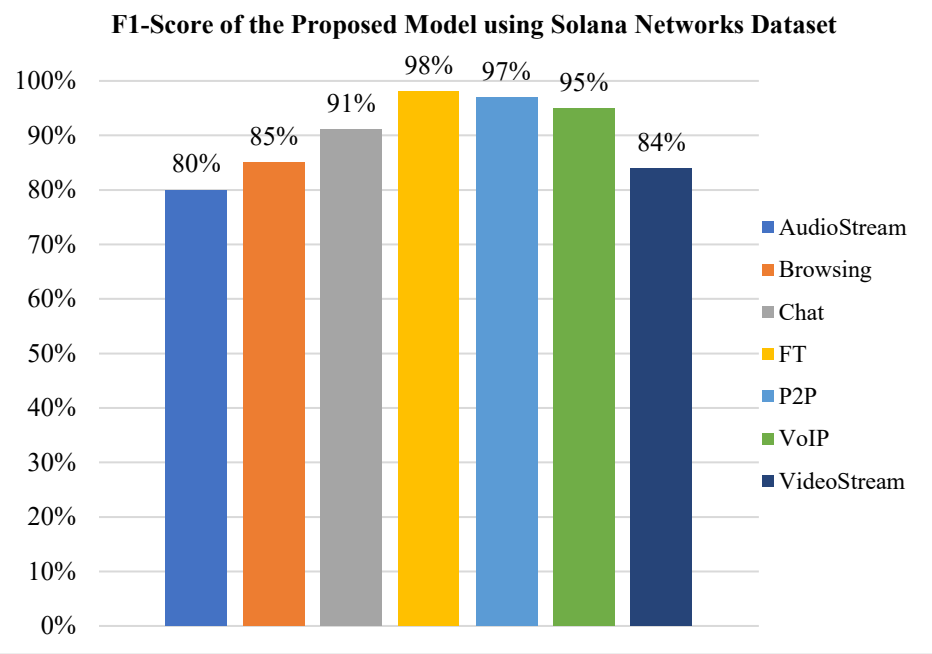
Figure 7.6 *Proposed System correlation matrix (Solana Networks Dataset)*

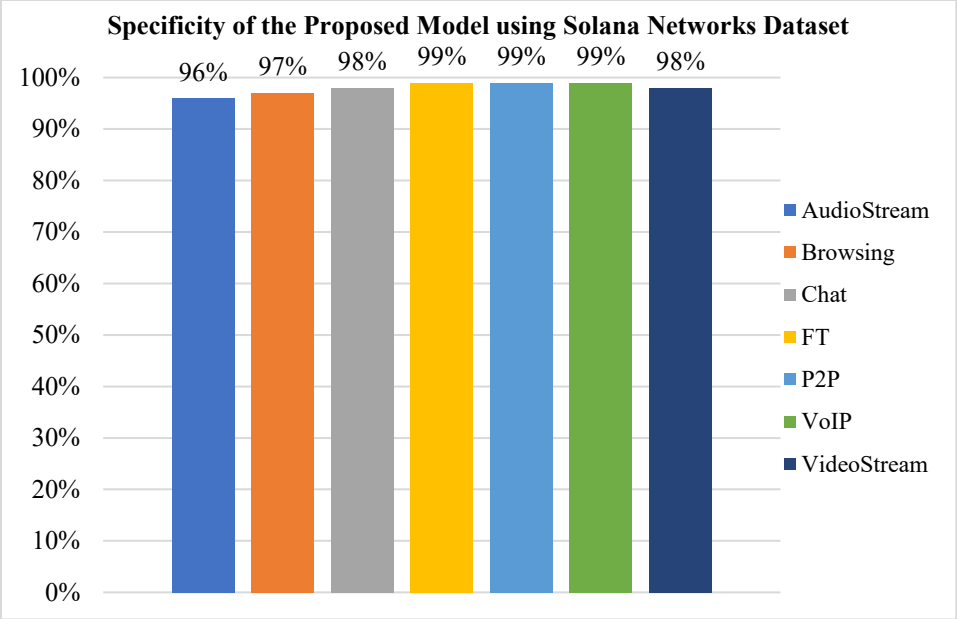
Table 7.2 Proposed Ensemble Model Functional Result for each category (Solana Networks Dataset)

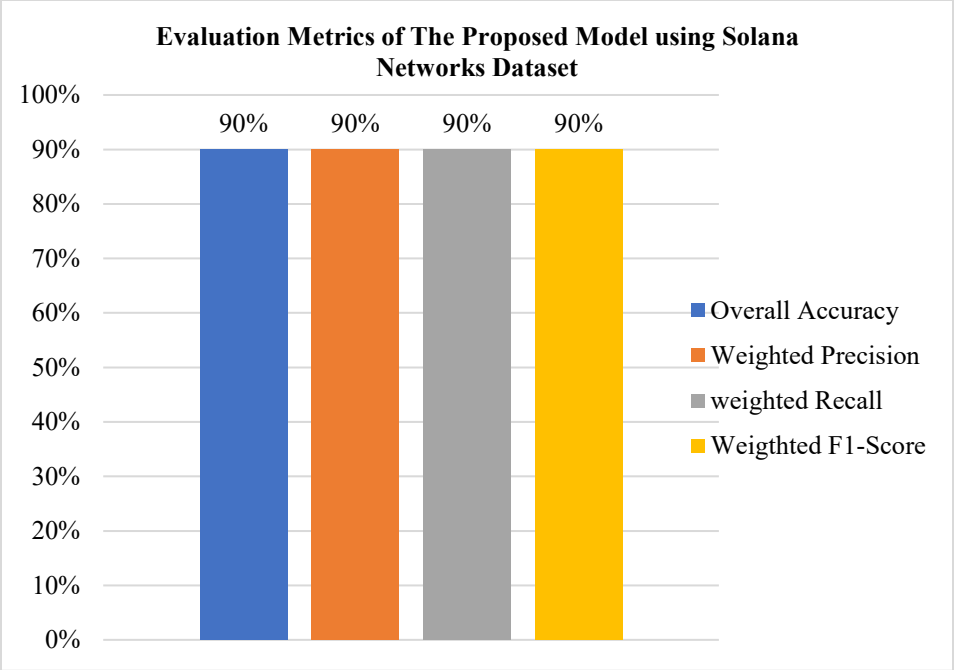
Number	Label	Precision	Recall	F1-score	Specificity
0	AudioStream	77%	83%	80%	96%
1	Browsing	85%	86%	85%	97%
2	Chat	88%	94%	91%	98%
3	FT	97%	98%	98%	99%
4	P2P	98%	95%	97%	99%
5	VOIP	95%	94%	95%	99%
6	VideoStream	91%	79%	84%	98%
Overall Accuracy					**90%**
Weighted Precision					**90%**
Weighted Recall					**90%**
Weighted F1-Score					**90%**




Recall of the Proposed Model using Solana Networks Dataset
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
83%
86%
94%
98%
95%
94%
79%
AudioStream
Browsing
Chat
FT
P2P
VoIP
VideoStream


F1-Score of the Proposed Model using Solana Networks Dataset
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
80%
85%
91%
98%
97%
95%
84%
AudioStream
Browsing
Chat
FT
P2P
VoIP
VideoStream





7.3 Efficiency Evaluation

Here, we evaluate the efficiency of our proposed model in terms of time taken, resource consumption, memory consumption, CPU, Input/output, and its scalability. For us to design and implement our model, we set up a system with specifications 8GB of Random-Access Memory (RAM) size, and operating system of Windows 10 (64-bits), Intel Core i7-8550U (4 cores) at 1.8GHz and disk space of 100GB. Next was to

set up the python environment. We installed python with Anaconda on the system. Anaconda is a distribution of python which includes many libraries including the ones we used in building our model. We used the Jupyter notebook development environment since it is the most popular IDE in data science for analyzing and exploring data. Our proposed model which was designed with the Ensemble Learning method algorithm using the stacking technique takes the encrypted network traffic data as input and outputs the classified encrypted network traffic, was trained using 70% train set of the data set for the model to learn about the instances, the independent variables and the class variables including the hidden patterns between them. During the training of our model, the time taken was recorded as well as the time taken when testing the model's performance on an unseen data set which is the 30% test set of the data set in the classification of encrypted network traffic data. Our proposed model is scalable even when there was an increase in the dataset. When designing our proposed model, we paid close attention to the number of neurons we assigned to the hidden layers of the deep learning model to accommodate a huge amount of dataset when such is fed into the model, also the number of estimators was carefully chosen too. It is also designed to classify encrypted network traffic data with more classes or network categories. Table 7.3 below shows the efficiency evaluation of our proposed model using the public dataset while table 7.4 shows the efficiency evaluation of our model using the Solana Networks dataset.

Table 7.3 Proposed Ensemble Model Efficiency Evaluation (Public dataset)

Resources	Windows 10 (64-bits), python with Anaconda, Jupyter notebook development environment
Memory Size	8GB of RAM and disk space of 100GB
CPU Size	Intel Core i7-8550U at 1.8GHz
Input	Encrypted network traffic dataset
Output	Classified encrypted network traffic data
Time Taken	1Hour 5minutes
RAM used	1.5GB

Table 7.4 Proposed Ensemble Model Efficiency Evaluation (Solana Networks dataset)

Resources	Windows 10 (64-bits), python with Anaconda, Jupyter notebook development environment
Memory Size	8GB of RAM and disk space of 100GB
CPU Size	Intel Core i7-8550U at 1.8GHz
Input	Encrypted network traffic dataset
Output	Classified encrypted network traffic data
Time Taken	4Hour 40minutes
RAM used	4.7GB

7.4 Complexity Analysis

We used the Ensemble Learning algorithm with the stacking technique to design our proposed solution for the classification of encrypted network traffic data. We split the encrypted network traffic dataset into train and test sets and executed our proposed model's algorithms on the train set to improve the model's computing efficiency during the classification of the network traffic data. After training the models in the first stage of the ensemble learning, the test set was fed into the models to test the model's algorithm learning efficiency (i.e. learning about the independent variables and the hidden patterns and then classifying the test sets to the class categories). Then, the classification results were merged to form a new train set for the second stage classification algorithm of the ensemble model. The new train sets were used in training the second stage algorithm and the test set was fed into the model for the classification of the encrypted network traffic data. Here, we determined the time complexity of our model (i.e. how the runtime of our proposed model algorithm grows as the size of the input grows) by using the Big O notation. When executing our proposed model algorithm, it first balances the imbalanced dataset using the SMOTETomek method. The time complexity of this is $O(n)$ because the runtime increases as the input size increases. Next, it preprocesses the data using the MinMax scaler, LabelEncoder, and OneHotEncoder. The time complexity of each preprocessing is $O(1)$ which is a constant time irrespective of the size of the input. We then split the data into a train and test set randomly where each partition is assigned to its respective set. Therefore, the time

complexity of this is $O(n)$. We then designed our Ensemble model using the learning algorithms (CapsNet+XGBoost, ANN+XGBoost, RF+XGBoost, DT+AdaBoost, and DT+XGBoost) in the first stage of the ensemble model for a kfold of 10. The time complexity for each of the first stage algorithms is $O(n) = 5 * O(n)$, where n is the size of the input data. Lastly, we designed the learning algorithm of the last stage of our Ensemble model using the Random Forest algorithm to train the outputs from the first stage algorithms to classify the encrypted network traffic data. Assuming that the depth of the tree which makes up the random forest is $O(\log n)$, this would be $O(ntree * n * \log n)$ where $ntree$ is the number of trees and n is the size of the input. Since the maximum depth of the tree is set and considered to be "d", then its complexity would be $O(ntree * n * d)$. If $n \gg (ntree, d)$, then its time complexity would be $O(n)$. Therefore, the time complexity of the model is

$$T = O(n) + O(1) + O(n) + 5 * O(n) + O(n) = c_1 + 8 * O(n) = O(n).$$

This means that as the input size increases, the runtime increases linearly.

7.5 Comparison of results

Here, we compare the result of our proposed solution with some of the existing studies and research. In the previous studies on the classification of encrypted network traffic data, authors in (Salman et. al., 2018) designed a system called ConvNet that was built using the CNN algorithm to classify the encrypted network traffic data into 4 classes which are interactive, streaming, bulk data transfer and transaction using self-generated dataset and the public dataset from UNB. Setting the Maximum Transfer Unit (MTU) for the packet size at 1500 bytes and considering four features (packet size, interarrival time, flow direction and transport protocol) they achieved an overall accuracy of 95.84%, a precision of 91.65%, a recall of 91.82% and f1-score of 91.38%. Their experiment was performed on a Linux machine with an operating system of Ubuntu 14.04 LTS-64-bit, CPU of Intel corei7-3630QM at 2.40GHz (8cores) and 16GB of RAM. In

(Lotfollahi et. al., 2017), the authors designed a novel system called Deep packet using the CNN and SAE (Stacked Auto Encoder) algorithms. They also utilized the UNB dataset in building their model. They classified the encrypted network traffic into 12 classes and obtained an overall accuracy of 94%, weighted recall of 94%, weighted precision of 93%, and weighted f1-score of 93%. Their system was unable to detect traffic data from an unknown class. In (Cui et. al., 2019), the authors deleted the network traffic data generated from browsing during the classification of their encrypted network traffic data into 12 classes. Their system was built using the Capsule Neural Network algorithm and the UNB dataset. They set the MTU to 784bytes which means that not all packets in a flow were considered. Their experiment was performed on an Ubuntu 16.04 64-bit operating system, a DELL R720 server with 16CPU cores and 128GB of memory, and an Nvidia Corporation GM204GL GPU was used as the accelerator. They achieved an overall accuracy of 99.1%, weighted precision, weighted recall, and weighted f1-score of 99.3%, 99.3%, and 99.3% respectively. The total number of statistical features and their names was not provided. In building our proposed solution, we included network traffic data generated from browsing for the classification of the encrypted network traffic. Authors in (Shapira and Shavitt, 2019) designed their model using the LeNet5 style of CNN architecture (was proposed by Yann LeCun et al. in 1995) in classifying the encrypted network traffic data into 5 classes (VoIP, video, file transfer, chat, and browsing). They achieved an overall accuracy of 85%. In (W. Wang et. al., 2017), the authors achieved the highest accuracy of 99.9% in classifying the encrypted network traffic data (VPN traffic) into 12 classes. Their system was designed using the 1-Dimensional Convolution Neural Network (1D-CNN) algorithm and the UNB dataset. They set the MTU for the packet size to 784bytes thereby not considering enough packets for each flow in the traffic dataset. For their system to achieve an accuracy of 99.9% for VPN network traffic classification, they might have used packets that contain all the headers from the 5 layers of the IP stack. According to the UNB dataset they used, the source and destination IP addresses are unique for each application, thus, their model must have used the feature in the classification of the VPN network traffic. To avoid this problem, we

decided to mask the IP address fields during the preprocessing stage before training and testing our proposed model. Their system performed poorly in classifying non-VPN network traffic data which achieved an accuracy of 86.6%, precision of 85.8%, and a recall of 85.9%. According to (Lopez et. al., 2017), the authors designed a system using CNN+RNN algorithms to classify the encrypted network traffic into their appropriate categories using a total number of 6 features (source port, destination port, direction, payload, window size and interarrival time). Their system achieved an overall accuracy of 96.32%, precision, recall, and f1-score of 95.43%, 96.32%, and 95.74% respectively. Including the source and destination port numbers as among the features being considered for classification might cause overfitting of their model which must have affected the performance of their model. In (McGaughey et. al., 2018), the authors designed their classification system using kNN+FOS (Fast Orthogonal Search) which has to do with manual extraction of the features. They achieved an accuracy of 81% using 44 features and an AUC of 98.98% using 12 features after implementing the FOS technique. After comparing some of the results of the previous work done, it was deduced that our proposed model designed using the Ensemble Learning method algorithm performs better in the classification of the encrypted network traffic data (non-VPN) into 7 classes using a total number of 23 extracted statistical features for the public dataset and 27 extracted statistical features for Solana Networks dataset. Table 7.5 illustrates the comparison of our model and other studies that used the public dataset from UNB.

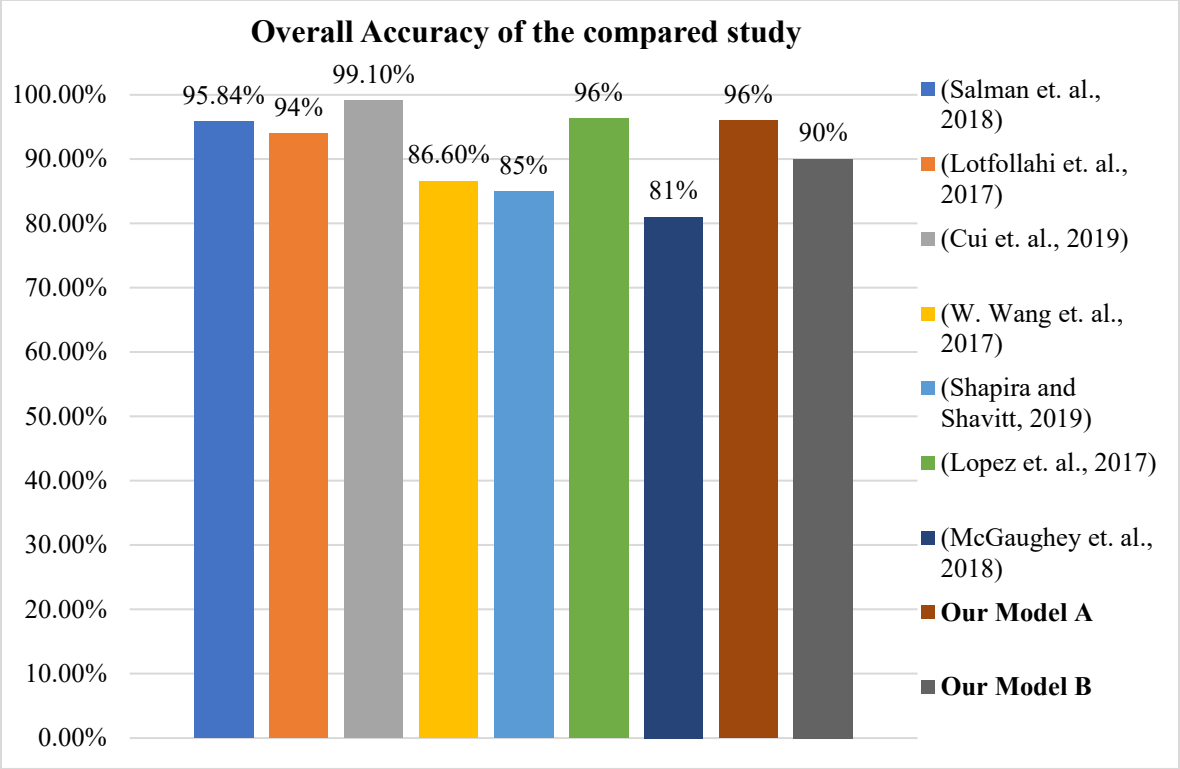
Table 7.5 Comparison of our model and other studies

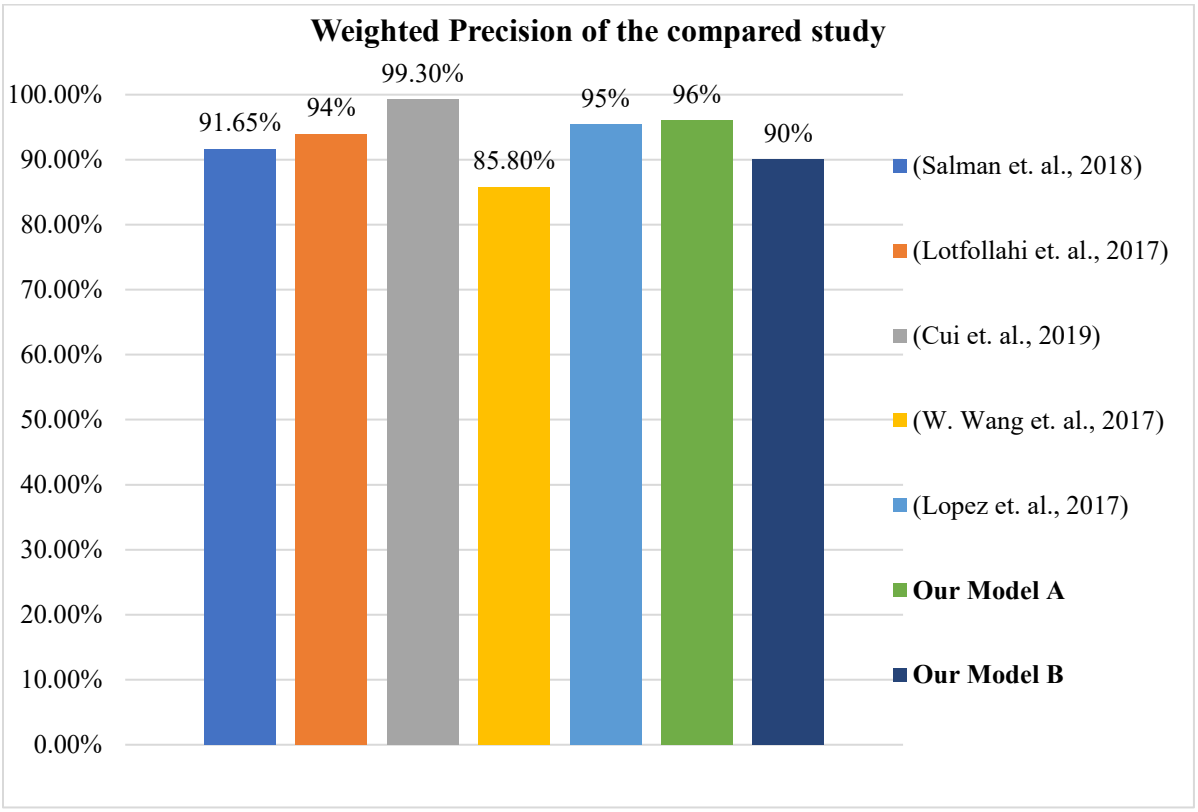
Research / Study	Method	Device	Dataset	Input Size (Bytes)	Number of Features	Classes
(Salman et. al., 2018)	CNN	Linux machine with an operating system of Ubuntu 14.04 LTS-64-bit, CPU of Intel corei7-3630QM at 2.40GHz (8cores) and 16GB of RAM	Self-generated dataset and UNB public dataset	1500	4	4
(Lotfollahi et. al., 2017)	CNN and SAE	-	UNB public dataset	1500	-	12
(Cui et. al., 2019)	CapsNet	Ubuntu 16.04 64-bit operating system, a DELL R720 server with 16CPU cores and 128GB of memory, and an Nvidia Corporation GM204GL GPU was used as the accelerator	UNB public dataset	784	-	12
(W. Wang et. al., 2017)	1D-CNN	-	UNB public dataset	784	-	12
(Shapira and Shavitt, 2019)	CNN	-	UNB public dataset	-	-	5
(Lopez et. al., 2017)	CNN+RNN	-	RedIRIS dataset	-	6	-
(McGaughey et. al., 2018)	kNN+FOS	-	Self-generated dataset	-	44	-
Our model A	**Ensemble Technique**	**Windows 10 (64-bits), 8GB of RAM and disk space of 100GB, Intel Core i7-8550U (4 cores) at 1.8GHz**	**UNB public dataset**	**1500**	**23**	**7**
Our model B	**Ensemble Technique**	**Windows 10 (64-bits), 8GB of RAM and disk space of 100GB, Intel Core i7-8550U (4 cores) at 1.8GHz**	**Solana Networks dataset**	**1500**	**27**	**7**

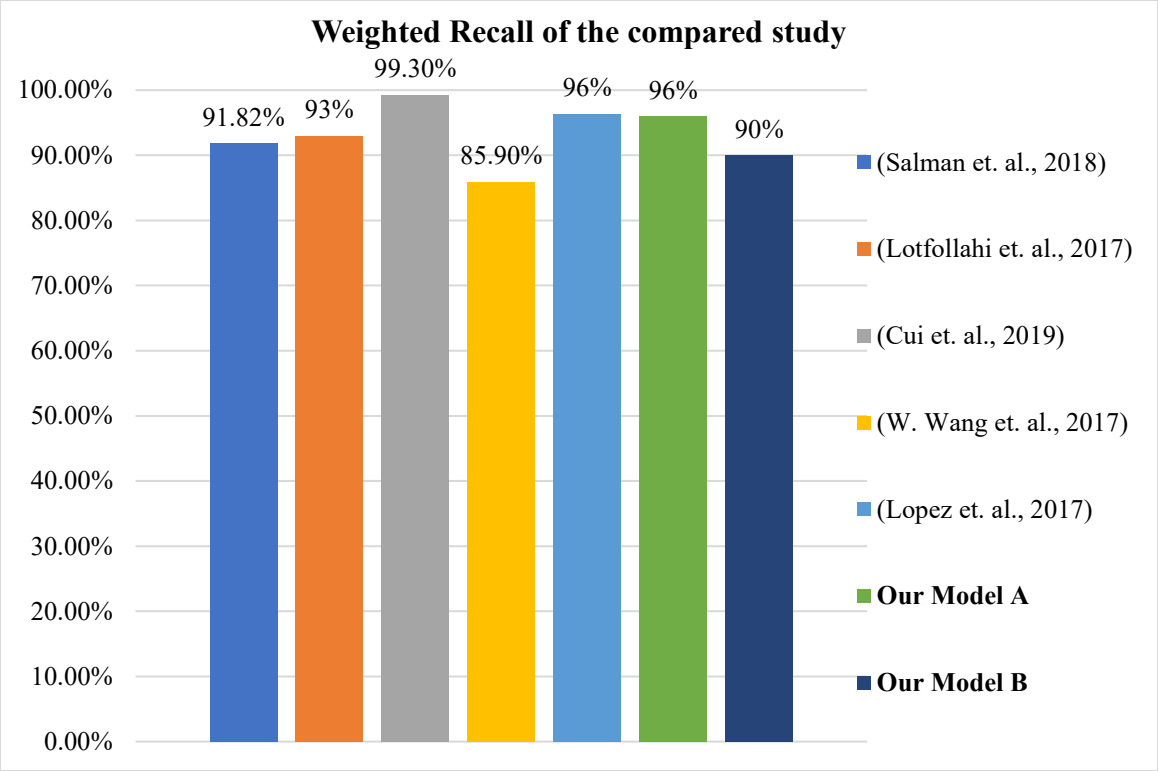
In table 7.6 below, we compare our model with other studies in terms of the number of extracted statistical features, the Maximum Transmission Unit, the labels of the network categories, and non-VPN encrypted network traffic data.

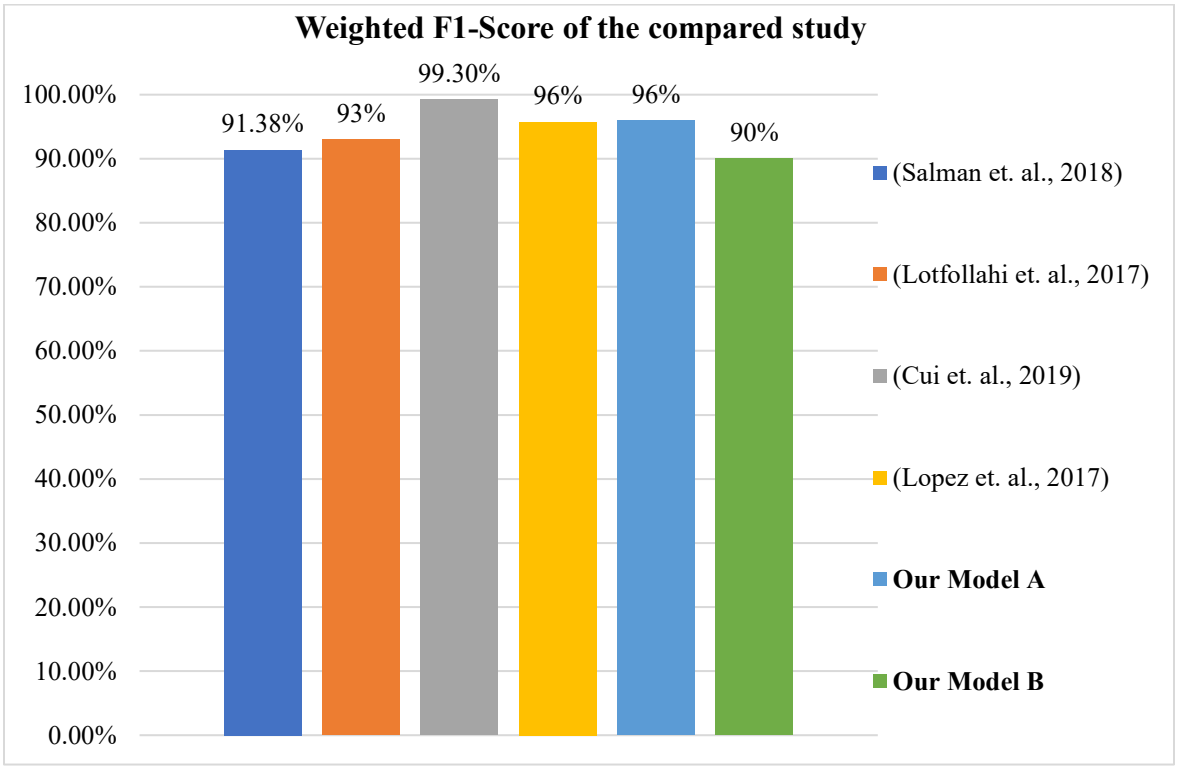
Table 7. 6 Comparison of our model and other studies in terms of performance

Research / Study	Overall Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score
(Salman et. al., 2018)	95.84%	91.65%	91.82%	91.38%
(Lotfollahi et. al., 2017)	94%	94%	93%	93%
(Cui et. al., 2019)	99.1%	99.3%	99.3%	99.3%
(W. Wang et. al., 2017)	86.6%	85.8%	85.9%	-
(Shapira and Shavitt, 2019)	85%	-	-	-
(Lopez et. al., 2017)	96.32%	95.43%	96.32%	95.74%
(McGaughey et. al., 2018)	81%	-	-	-
Our model A	**96%**	**96%**	**96%**	**96%**
Our model B	**90%**	**90%**	**90%**	**90%**


Overall Accuracy of the compared study
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
95.84%
94%
99.10%
86.60%
85%
96%
81%
96%
90%
(Salman et. al., 2018)
(Lotfollahi et. al., 2017)
(Cui et. al., 2019)
(W. Wang et. al., 2017)
(Shapira and Shavitt, 2019)
(Lopez et. al., 2017)
(McGaughey et. al., 2018)
Our Model A
Our Model B


Weighted Precision of the compared study
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
91.65%
94%
99.30%
85.80%
95%
96%
90%
(Salman et. al., 2018)
(Lotfollahi et. al., 2017)
(Cui et. al., 2019)
(W. Wang et. al., 2017)
(Lopez et. al., 2017)
Our Model A
Our Model B


Weighted Recall of the compared study
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
91.82%
93%
99.30%
85.90%
96%
96%
90%
(Salman et. al., 2018)
(Lotfollahi et. al., 2017)
(Cui et. al., 2019)
(W. Wang et. al., 2017)
(Lopez et. al., 2017)
Our Model A
Our Model B


Weighted F1-Score of the compared study
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
91.38%
93%
99.30%
96%
96%
90%
(Salman et. al., 2018)
(Lotfollahi et. al., 2017)
(Cui et. al., 2019)
(Lopez et. al., 2017)
Our Model A
Our Model B

Chapter 8: Conclusion and Future work

8.1 Conclusion

This thesis work presents experiments conducted on the existing public network traffic dataset from the University of New Brunswick (UNB) using the Ensemble Learning technique to classify the non-VPN encrypted network traffic. The model was also tested on a real-world network traffic dataset from Solana Networks to determine its performance accuracy. The traditional method or approach to the classification of network traffic data is being challenged due to the recent advancement in technology that focuses on customers' privacy and security of the communications in the network using encryption enabled protocols thereby making the DPI (Deep Packet Inspection) approach not to be considered. Making use of the CICFlow meter to extract the statistical features of a network traffic flow and setting a threshold to determine the relevant features required for the classification of the encrypted network traffic, we were able to determine a total number of 23 features for the public dataset and 27 features for the Solana Networks dataset. Having extracted the statistical features required for the classification and preparing the structured datasets, we examined the proposed solution (Ensemble Learning approach) which was designed using the deep and machine learning classification algorithms with the stacking technique to solve the network traffic classification problem. As shown in chapter 6 of our study, ensemble learning method is suitable for the classification of the encrypted network traffic data as it improves the model's results by combining different classification models which are the base models and merging their outputs to form a new input for the second stage classification algorithm to produce better classification accuracy compared to that of a single classification model. The proposed system makes use of masked IP addresses and port numbers of flow-based extracted statistical features of a traffic dataset for the classification of encrypted network traffic data. The flow-based dataset is a bidirectional flow that contains a couple of packets per flow with a Maximum Transfer Unit (MTU) of 1500 bytes for the classification of network traffic data into 7 categories. The

proposed system which is the ensemble learning with the stacking technique achieved an overall accuracy of 96% with the public dataset and 90% overall accuracy using the real-world (Solana Networks) dataset in the classification of network traffic data distinguishing between all classes recorded in the dataset. This corresponded to an AUC of 98% and 94% for the public and real-world datasets, respectively. Having achieved a higher accuracy using the public dataset, this led us to investigate the classification results of both datasets when predicting each of the network categories or classes. It was evident the public dataset used 23 independent variables while the real-world dataset used 27 independent variables during their respective classification process. The important findings were that the model was able to classify traffics generated from the P2P and VoIP categories of both datasets properly achieving a precision of not less than 95% respectively, irrespective of the category's contents in both datasets. For the browsing category, the proposed model achieved a precision of 95% using the public dataset but achieved 85% precision using the real-world dataset. The result of the real-world dataset was affected by the different networks used during the browsing session with Google Chrome and Mozilla Firefox web browsers. Also, the proposed system performed poorly in the classification of traffics generated from SoundCloud and Spotify yielding an f1-score of 80%. Generally, our system performed excellently well in the classification of the traffic data using the two different network traffic datasets.

Furthermore, the extracted statistical features played an important role in the performance of the proposed model. The system can be used to help the Internet Service Providers (ISPs) to offer a Quality of Service to the customers as well as monitoring and managing the network, securing the network from malware and intrusions, making proper network planning and troubleshooting of the network to mention just a few. This thesis shows that our proposed model can classify encrypted network traffic properly irrespective of the number of statistical features and the contents of the categories (or classes) considered while maintaining the users' privacy.

8.2 Future Work

Due to the timeframe of this thesis project, which was limited, there are some experiments and implementation that have been left for future work. These are, to extend our work or model to capture other applications and types of encryption tools such as TrueCrypt and Tor traffic data to learn how our model can perform in the classification of those type of traffic data. The network traffic data from applications like WhatsApp, Snapchat, Titok, and SHAREit will be of interest because of their high usage and end-to-end encryption. Also, training the model with non-VPN and VPN network traffic data will be great to experiment and examine how the system will classify both (non-VPN and VPN) traffic data.

Including extracted statistical features like the packet length where the minimum, maximum, mean, standard deviation, and variance is calculated, Finish flag count, Synchronization flag count, Reset flag count, Push flag count and Acknowledgement flag in the classification of the encrypted network traffic data will be of interest to observe its influence in the classification model.

Furthermore, we would like to implement it in an online platform to determine its (the model's) performance and robustness while considering the network bandwidth, demand, and traffic data that runs into billions of traffic flows. By doing so, this will help the ISPs manage the network better as well as implementing the necessary changes faster.

References

(Alachiotis and Stamatakis, 2010) Alachiotis, N., & Stamatakis, A. (2010, April). Efficient floating-point logarithm unit for FPGAs. In 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops, and Ph.D. Forum (IPDPSW) (pp. 1-8). IEEE.

(Ali, 2019) Arish Ali. Ensemble Machine Learning Techniques. 2019. Packt Publishing.

(Alpaydin, 2020) Alpaydin, E. (2020). Introduction to machine learning. MIT press.

(Aouini et. al., 2018) Z. Aouini, A. Kortebi, Y. Ghamri-Doudane and I. L. Cherif, "Early classification of residential networks traffic using C5.0 machine learning algorithm," 2018 Wireless Days (WD), Dubai, 2018, pp. 46-53.

(Bagui et. al., 2017) Bagui, S., Fang, X., Kalaimannan, E., Bagui, S. C., & Sheehan, J. (2017). Comparison of machine-learning algorithms for classification of VPN network traffic flow using time-related features. Journal of Cyber Security Technology, 1(2), 108-126.

(Bengio, 2012) Bengio, Y. (2012, June). Deep learning of representations for unsupervised and transfer learning. In Proceedings of ICML Workshop on unsupervised and transfer learning (pp. 17-36).

(Biersack et. al., 2013) E. Biersack, C. Callegari, and M. Matijasevic, Data traffic monitoring, and analysis. Berlin: Springer, 2013.

(Bonaventure, 2011) Bonaventure, O. Computer Networking: Principles, Protocols, and Practice; The Saylor Foundation: Washington, DC, USA, 2011.

(C. Liu et. al., 2019) C. Liu, L. He, G. Xiong, Z. Cao and Z. Li, "FS-Net: A Flow Sequence Network for Encrypted Traffic Classification," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Paris, France, 2019, pp. 1171-1179.

(Cao et. al., 2014) Cao, Z., Xiong, G., Zhao, Y., Li, Z., & Guo, L. (2014, November). A survey on encrypted traffic classification. In International Conference on Applications and Techniques in Information Security (pp. 73-81). Springer, Berlin, Heidelberg.

(Casino et. al., 2019) F. Casino, K. R. Choo, and C. Patsakis, "HEDGE: Efficient Traffic Classification of Encrypted and Compressed Packets," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 11, pp. 2916-2926, Nov. 2019.

(Chen and Guestrin, 2016) Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM signed international conference on knowledge discovery and data mining (pp. 785-794).

(Cui et. al., 2019) S. Cui, B. Jiang, Z. Cai, Z. Lu, S. Liu, and J. Liu, "A Session-Packets-Based Encrypted Traffic Classification Using Capsule Neural Networks," 2019 IEEE 21st International Conference on High-Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 2019, pp. 429-436.

(Danish, 2016) Danish, S., 2016. Data Preprocessing In Python | Sklearn Preprocessing. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2016/07/practical-guide-data-preprocessing-python-scikit-learn/

(Dimaano, 2019) Dimaano, F., 2019. What Is Machine Learning? [online] Medium. Available at: https://towardsdatascience.com/what-is-machine-learning-885aa35db58b

(Dordal, 2014) Dordal, P. L. (2014). An introduction to computer networks.

(Elsinghorst and Shirin, 2018) Elsinghorst, Dr. Shirin. “Machine Learning Basics - Gradient Boosting & XGBoost.” Shirin’s PlaygRound, www.shirin-glander.de/2018/11/ml_basics_gbm/.

(Foroushani and Heywood, 2015) Aghaei-Foroushani, V., & Zincir-Heywood, A. N. (2015, January). A proxy identifier based on patterns in traffic flows. In 2015 IEEE 16th International Symposium on High Assurance Systems Engineering (pp. 118-125). IEEE.

(Gil et. al., 2016) Gil GD, Lashkari AH, Mamun M, Ghorbani AA (2016) Characterization of encrypted and VPN traffic using time-related features. In: Proceedings of the 2nd international conference on information systems security and privacy (ICISSP 2016), pp 407–414

(Goodfellow et. al., 2016) Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

(Gupta, 2017) Gupta, P. (2017). Decision Trees in Machine Learning. [online] Towards Data Science. Available at: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

(H. Yao et al., 2019) H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, and Z. Han, "Capsule Network Assisted IoT Traffic Classification Mechanism for Smart Cities," in IEEE Internet of Things Journal, vol. 6, no. 5, pp. 7515-7525, Oct. 2019.

(Harliman and Uchida, 2018) Harliman, R., & Uchida, K. (2018). Data-and algorithm-hybrid approach for imbalanced data problems in deep neural network. International Journal of Machine Learning and Computing, 8(3), 208-213.

(Hinton et. al., 2011) Hinton, G. E., Krizhevsky, A., & Wang, S. D. (2011, June). Transforming auto-encoders. In International conference on artificial neural networks (pp. 44-51). Springer, Berlin, Heidelberg.

(Hinton et. al., 2018) Hinton, G. E., Sabour, S., & Frosst, N. (2018). Matrix capsules with EM routing.

(Hinton et. al., 2017) Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In Advances in neural information processing systems (pp. 3856-3866).

(Hinton et. al., 2014) Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

(Ioffe and Szegedy, 2015) Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

(Jaikaran, 2020) Jaikaran, Chris. Encryption: Frequently Asked Questions, report, Date Unknown; Washington D.C.. (https://digital.library.unt.edu/ark:/67531/metadc944683/: accessed February 6, 2020), University of North Texas Libraries, UNT Digital Library, https://digital.library.unt.edu; crediting UNT Libraries Government Documents Department.

(Jain et. al., 1996) A. K. Jain, Jianchang Mao and K. M. Mohiuddin, "Artificial neural networks: a tutorial," in Computer, vol. 29, no. 3, pp. 31-44, March 1996.

(James et. al., 2013) James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, pp. 3-7). New York: springer.

(Javaid, 2018) Javaid Nabi (2018). Machine Learning—Multiclass Classification with Imbalanced Dataset. [online] Medium. Available at: https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a

(Kho, 2018) Kho, J. (2018). Why Random Forest is My Favorite Machine Learning Model. Retrieved 18 July 2020, from https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706

(Kingma and Ba, 2014) Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

(Koo, 2018) Koo Ping Shung (2018). Accuracy, Precision, Recall, or F1? [online] Towards Data Science. Available at: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

(Kshitizrimal, 2019) Kshitizrimal. “Understanding Capsule Network Architecture.” Software.Intel.Com, 29 Apr. 2019, software.intel.com/en-us/articles/understanding-capsule-network-architecture.

(Kuhn and Johnson, 2013) Kuhn, M., & Johnson, K. (2013). Applied predictive modeling (Vol. 26). New York: Springer.
(Lakhtaria and Kamaljit, 2011) Lakhtaria, Kamaljit. (2011). Protecting Computer Network with Encryption Technique: A Study. 381-390. 10.1007/978-3-642-20998-7_47.
(Lawrence et. al., 2017) Lawrence, J., Malmsten, J., Rybka, A., Sabol, D. A., & Triplin, K. (2017). Comparing TensorFlow deep learning performance using CPUs, GPUs, local pcs, and cloud.
(LeCun et. al., 1995) LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10), 1995.
(Li et. al., 2018) T. Li, S. Chen, Z. Yao, X. Chen and J. Yang, "Semi-supervised network traffic classification using deep generative models," 2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Huangshan, China, 2018, pp. 1282-1288.
(Linuxmint.com) Linuxmint.com. (2020). [online] Available at: https://linuxmint.com/documentation/user-guide/Cinnamon/english_18.0.pdf.
(Liu et. al., 2018) Liu, Z. Cao, G. Xiong, G. Gou, S. Yiu, and L. He, "MaMPF: Encrypted Traffic Classification Based on Multi-Attribute Markov Probability Fingerprints," 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), Banff, AB, Canada, 2018, pp. 1-10.
(Lopez et. al., 2017) M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network Traffic Classifier with Convolutional and Recurrent Neural Networks for the Internet of Things," in IEEE Access, vol. 5, pp. 18042-18050, 2017.
(Lotfollahi et. al., 2017) M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian, “Deep packet: A novel approach for encrypted traffic classification using deep learning,” eprint arXiv:1709.02656, Sep. 2017
(Manish, 2018) Manish Pathak (2018). Using XGBoost in Python. [online] DataCamp Community. Available at: https://www.datacamp.com/community/tutorials/xgboost-in-python
(McGaughey et al. 2018) D. McGaughey, T. Semeniuk, R. Smith, and S. Knight, "A systematic approach of feature selection for encrypted network traffic classification," 2018 Annual IEEE International Systems Conference (SysCon), Vancouver, BC, 2018, pp. 1-8.
(Miller et. al., 2018) S. Miller, K. Curran and T. Lunney, "Multilayer Perceptron Neural Network for Detection of Encrypted VPN Network Traffic," 2018 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA), Glasgow, 2018, pp. 1-8.
(monkeyrunner, 2019) Monkeyrunner - Android Developers, 2019, https://developer.android.com/studio/test/monkeyrunner/.
(MOORE, 2005) MOORE A W, ZUEV D. Discriminators for Use in Flow-Based Classification [R]. Technical report, Intel Research, Cambridge, 2005.
(Myung-Sup et. al., 2004) K. Myung-Sup, Y.J. Won, H.J. Lee, 1.W. Hong and R. Boutaba "Flow-based Characteristic Analysis of Internet Application Traffic," In Proceedings of the 2nd Workshop on End-to-End Monitoring Techniques and Services, pp.62-67, San Diego, USA, October 2004
(N. Lu et. al., 2012) C. N. Lu, C. Y. Huang, Y. D. Lin, and Y. C. Lai, Session level flow classification by packet size distribution and session grouping, Computer Networks, vol. 56, no. 1, pp. 260-272, 2012.
(netflowmeter.ca) www.netflowmeter.ca/netflowmeter.html
(Obasi and Shafiq, 2019) T. Obasi and M. Omair Shafiq, "Towards comparing and using Machine Learning techniques for detecting and predicting Heart Attack and Diseases," 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 2393-2402.
(Ognjanovski, 2019) Ognjanovski, G., 2019. Everything You Need To Know About Neural Networks And Backpropagation—Machine Learning Made Easy…. [online] Medium. Available at:

https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a

(OpenLearn, 2020) OpenLearn. (2020). Network security. [online] Available at: https://www.open.edu/openlearn/science-maths-technology/computing-and-ict/systems-computer/network-security/content-section-5.3.2

(Oshana and Kraeling, 2019) Oshana, R., & Kraeling, M. (Eds.). (2019). Software engineering for embedded systems: Methods, practical techniques, and applications. Newnes.

(P. Wang et. al., 2018) P. Wang, F. Ye, X. Chen and Y. Qian, "Datanet: Deep Learning-Based Encrypted Network Traffic Classification in SDN Home Gateway," in IEEE Access, vol. 6, pp. 55380-55391, 2018.

(Paul, 2018) Paul, S., 2018. Ensemble Learning In Python. [online] DataCamp Community. Available at: https://www.datacamp.com/community/tutorials/ensemble-learning-python

(QUIC, 2018) V. Tong, "Network flow of quic," https://drive.google.com/drive/folders/1cwHhzvaQbiap8yfrj2vHyPmUTQhaYOj? usp=sharing, April 2018.

(Rajat, 2018) Rajat Harlalka (2018). Choosing the Right Machine Learning Algorithm. [online] Hacker Noon. Available at: https://hackernoon.com/choosing-the-right-machine-learning-algorithm-68126944ce1f

(Rosenblatt, 1957) Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory.

(Rumelhart et. al., 1986) Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

(Sagar, 2017) Sagar, S., 2017. Activation Functions In Neural Networks. [online] Medium. Available at: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

(Saha, 2018) Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. [online] Towards Data Science. Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

(Salman et al. 2018) O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi, "A Multi-level Internet Traffic Classifier Using Deep Learning," 2018 9th International Conference on the Network of the Future (NOF), Poznan, 2018, pp. 68-75.

(Schmoll and Zander, 2009) C. Schmoll and S. Zander, *NetMate - Version 0.9.5*. Germany: Fraunhofer FOKUS, 2009.

(Schott, 2019) Schott, M., 2019. Random Forest Algorithm For Machine Learning. [online] Medium. Available at: https://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb

(Scikit-learn, 2019) Scikit-learn.org. (2019). sklearn.tree.DecisionTreeClassifier — scikit-learn 0.22.1 documentation. [online] Available at: https://scikitlearn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

(scikit-learn, 2019) https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

(Shapira and Shavitt, 2019) T. Shapira and Y. Shavitt, "FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 2019, pp. 680-687.

(Shiravi et. al., 2012) A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, ``Toward developing a systematic approach to generate benchmark datasets for intrusion detection," Comput. Secur., vol. 31, no. 3, pp. 357374, 2012.

(Shung, 2018) Shung, K. (2018). Accuracy, Precision, Recall or F1? from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

(Singh, 2018) Singh, A. (2018). Ensemble Learning | Ensemble Techniques. Retrieved 18 July 2020, from https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

(Song et. al., 2020) Song, W., Beshley, M., Przystupa, K., Beshley, H., Kochan, O., Pryslupskyi, A., ... & Su, J. (2020). A Software Deep Packet Inspection System for Network Traffic Analysis and Anomaly Detection. *Sensors*, *20*(6), 1637.

(Tanner, 2019) Tanner, G. (2019). A guide to Ensemble Learning. [online] Medium. Available at: https://towardsdatascience.com/a-guide-to-ensemble-learning-d3686c9bed9a.

(TensorFlow, 2020) Carter, D.S., and S. (n.d.). Tensorflow — Neural Network Playground. [online] playground.tensorflow.org. Available at: https://playground.tensorflow.org/#activation=relu&batchSize=15&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=6

(Thathapudi et. al., 2012) Thathapudi, T. N., & Satyanarayana, S. D. (2012). *U.S. Patent No. 8,121,135*. Washington, DC: U.S. Patent and Trademark Office.

(Tong et al., 2018) V. Tong, H. A. Tran, S. Souihi, and A. Mellouk, "A Novel QUIC Traffic Classifier Based on Convolutional Neural Networks," 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.

(UNB, 2017) www.unb.ca. (n.d.). Tor 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. [online] Available at: http://www.unb.ca/cic/datasets/tor.html

(UNB, 2016) www.unb.ca.(n.d.). VPN 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. [online] Available at: http://www.unb.ca/cic/datasets/vpn.html

(USTC-TFC, 2016) USTC-TFC2016 dataset https://github.com/echowei/DeepTraffic

(Vu et. al., 2018) Vu, L., Thuy, H. V., Nguyen, Q. U., Ngoc, T. N., Nguyen, D. N., Hoang, D. T., & Dutkiewicz, E. (2018, September). Time Series Analysis for Encrypted Traffic Classification: A Deep Learning Approach. In 2018 18th International Symposium on Communications and Information Technologies (ISCIT) (pp. 121-126). IEEE.

(Vrána et. al., 2019) R. Vrána, J. Kořenek, and D. Novák, "Acceleration of Feature Extraction for Real-Time Analysis of Encrypted Network Traffic," 2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Cluj-Napoca, Romania, 2019, pp. 1-6.

(W. Wang et. al., 2017) W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, 2017, pp. 43-48.

(Will, 2018) Will, K., 2018. An Implementation And Explanation Of The Random Forest In Python. [online] Medium. Available at: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

(X. Zeng et. al., 2017) W. Wang, X. Zeng, X. Ye, Y. Sheng and M. Zhu, "Malware Traffic Classification Using Convolutional Neural Networks for Representation Learning" In the 31st Intl Conf on Information Networking (ICOIN), 2017

(Y. Zeng et. al., 2019) Y. Zeng, H. Gu, W. Wei, and Y. Guo, "$Deep-Full-Range$: A Deep Learning-Based Network Encrypted Traffic Classification and Intrusion Detection Framework," in IEEE Access, vol. 7, pp. 45182-45190, 2019.

(Y. Chen et al., 2019) Y. Chen, T. Zang, Y. Zhang, Y. Zhouz and Y. Wang, "Rethinking Encrypted Traffic Classification: A Multi-Attribute Associated Fingerprint Approach," 2019 IEEE 27th International Conference on Network Protocols (ICNP), Chicago, IL, USA, 2019, pp. 1-11.

(Yao et. al., 2019) Yao, H., Liu, C., Zhang, P., Wu, S., Jiang, C., & Yu, S. (2019). Identification of Encrypted Traffic Through Attention Mechanism-Based Long Short Term Memory. IEEE Transactions on Big Data.

(Z. Chen et. al., 2017) Z. Chen, K. He, J. Li, and Y. Geng, "Seq2Img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks," 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, 2017, pp. 1271-1276.

(Zhang et. al., 2018) F. Zhang, Y. Wang, and M. Ye, "Network Traffic Classification Method Based on Improved Capsule Neural Network," 2018 14th International Conference on Computational Intelligence and Security (CIS), Hangzhou, 2018, pp. 174-178.