Intrusion Detection System Using Deep Learning for Vehicular Ad Hoc Network

Betelhem Nigussie Mamo

A Thesis Submitted to the Department of Electronics and Communication Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Electronics and Communication Engineering

Office of Graduate Studies

Adama Science and Technology University

July 2023

Adama Ethiopia

Intrusion Detection System Using Deep Learning for Vehicular Ad Hoc Network

Betelhem Nigussie Mamo

Advisors:

Major Advisor: Dr. Rajeev K Shakya

Co-advisor: Dr.Javed Shaikh

A Thesis Submitted to the Department of Electronics and Communication Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Electronics and Communication Engineering

Office of Graduate Studies

Adama Science and Technology University

July 2023

Adama Ethiopia

Declaration

I hereby declare that this Master Thesis entitled “Intrusion Detection System Using Deep Learning for Vehicular Ad Hoc Network’’ is my original work. That is, it has not been submitted for the award of any academic degree, diploma or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation

Name of student	Signature	Date

Recommendation

We, the advisor(s) of this thesis, hereby certify that we have read the revised version of the thesis entitled “Intrusion Detection System Using Deep Learning for Vehicular Ad Hoc Network’’ prepared under our guidance by Betelhem Nigussie Mamo submitted in partial fulfillment of the requirements for the degree of Mater’s of Science in Electronics and Communication Engineering. Therefore, We recommend the submission of revised version of the thesis to the department following the applicable procedures.

Major Advisor	Signature	Date
Co-advisor	Signature	Date

Approval Page

We, the advisors of the thesis entitled “Intrusion Detection System Using Deep Learning for Vehicular Ad Hoc Network’’ and developed by Betelhem Nigussie Mamo, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Major Advisor	Signature	Date

Co-advisor	Signature	Date

Approval of Board of Examiners

We, the undersigned, members of the Board of Examiners of the thesis by Betelhem Nigussie Mamo have read and evaluated the thesis entitled “Intrusion Detection System Using Deep Learning for Vehicular Ad Hoc Network” and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Electronics and Communication Engineering.

Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the candidate’s Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGEMENT

Prior to all else, I'd want to thank God for his unprecedented blessings and for always being there for me. You have given me the confidence and bravery to believe in myself and improve my skills. I couldn't have done this without your faith in me. My name may be on the cover, but I cannot take credit for all that happened to get this thesis from idea to printing. As a result, I must express my gratitude to the persons mentioned elsewhere here.

I genuinely want to thank my advisor Dr. Rajeev Kumar Shakya for inspiring me to seek a master's degree and for providing me with ongoing support, advice, and assistance as I conducted a research technique. I am also grateful to my Co-advisor Dr. Javed Shaikh who inspired me to set high goals and to conduct this original thesis research.

I would like to thank Adama Science and Technology University for giving me the Opportunity to study at MSC. And I would like to thank Mr. Aron for helping me give the comment and suggestion on the proposal.

CONTENTS

CHAPTER PAGE

ACKNOWLEDGEMENT v
List of Tables iv
List of Figures v
List of Acronyms and Abbreviations vi
Abstract viii
Chapter One 1
1. Introduction 1
1.1 Background of the Study 1
1.2 Statement of Problem 2
1.3 General and Specific Objectives 3
1.3.1 General Objective 3
1.3.2 Specific Objectives 3
1.4 Significance of the Study 3
1.5 Workflow 4
1.6 Delimitation and Limitation of the Study 5
Chapter Two 6
2. Literature Review 6
2.1 Chapter Overview 6
2.1.1 Theoretical Background 6
2.2 Overview of VANET 6
2.2.1 What is VANET 6
2.2.2 Types of VANET Communication 8
2.2.3 Security Attacks and Threats in VANETs 9
2.2.4 Encrypted Internet Traffic 10
2.2.5 Intrusion Detection System (IDS) 10
2.2.6 Machine/Deep Learning Framework 11
2.2.7 Machine Learning 13
2.2.8 Deep Learning 13
2.2.9 Why is DL better than ML for IoT 13
2.2.10 Artificial Neural Network 14
2.2.11 Deep Neural Network 14

2.2.12 Tab Net DNN 15
2.2.13 ML and DL for the Domain of Cybersecurity 16
2.2.14 ML and DL Applications on IDS 16
2.2.15 Machine and Deep Learning Framework on IDS 17
2.3 Related Works 18
2.3.1 Summary and Critics 20
2.3.2 Identified Gaps 24
Chapter Three 25
3. Materials and Methods 25
3.1 Chapter Overview 25
3.1.1 System Design 25
3.1.2 Key Features for VANET's OBU Solutions 25
3.1.2.1 Lightweight 25
3.1.2.2 Distributive/Multilayered 26
3.1.2.3 Longevity 26
3.1.3 Advanced Features of AI 26
3.1.3.1 Interpretability 26
3.1.3.2 Explainable 27
3.1.3.3 Attentive/Attentiveness 27
3.1.4 VANET System Architecture 28
3.1.5 Tab Net: Attentive Interpretable Tabular Learning Model Architecture 29
3.1.5.1 Gated Linear Unit Layer 31
3.1.5.2 Feature Transformer 32
3.1.5.3 Attentive Transformer 32
3.1.5.4 Decision Step 33
3.2 Experimental Methods and Setups 34
3.2.1 Dataset Preparation 34
3.2.1.1 Dataset Description 35
3.2.1.2 Features 36
3.2.1.3 Data pre-processing 36
3.2.1.4 Data Mapping for Encrypted Internet Traffic 38
3.2.1.5 Data Partition for DL framework 40
3.2.1.6 Data split for TCP/IP layer 40

3.2.2 Proposed Modification on Fast.ai Tabular Training 41
3.2.3 Baseline Setups for Speed Test for PyTorch-Fast.ai 42
3.2.4 Experiment Environment Setup 42
3.2.5 Hyper-parameter 42
3.3 Evaluation Metrics 43
CHAPTER FOUR 45
4. Results and Discussions 45
4.1 Analysis of the Proposed Approach 45
4.2 Evaluation of the Deep Learning (DL) Framework 45
4.2.1 Performance Results of Theano-Keras 45
4.2.2 Performance Results of Tensor-Flow-Keras 46
4.2.3 Performance Results of PyTorch-Fast.ai 47
4.2.4 Comparison Results of DL Framework for IDS 47
4.3 Modified PyTorch-Fast.ai Evaluation 48
4.3.1 Performance Results of Baseline Fast.ai dataloader 48
4.3.2 Performance Results of Modified Fast.ai/Customized numpy dataloader 49
4.3.3 Comparison of the Results for Fast.ai Tabular Training 49
4.4 IDS Classifier Evaluation 50
4.4.1 Hyper parameter settings 50
4.4.1.1 Choosing a Good Learning Rate 50
4.4.1.2 Set Parameter Space and Value 51
4.4.2 Performance Results of each IDS classifier 51
4.4.2.1 Hyper parameter 51
4.4.2.2 Results of Feature Selection 52
4.4.2.3 Results of selective attention 53
4.4.3 IDS classifiers Comparison 55
4.4.4 Analysis of the IDS classifiers' Performance in Relation to Previous Work 55
CHAPTER FIVE 56
5. Conclusions and Future Works 56
5.1 Conclusions 56
5.2 Future Works 57
References 58

LIST OF TABLES

TABLE PAGE

TABLE 2.1: SUMMARY OF RELATED WORK 1. 21
TABLE 2.2: SUMMARY OF RELATED WORK 2. 22
TABLE 2.3: SUMMARY OF RELATED WORK 3. 23
TABLE 3.1: STATIC DISTRIBUTIONS OF ATTACK TYPE. 37
TABLE 3.2: NUMBER OF SAMPLES FOR EACH TYPE OF TRAFFIC DATA 38
TABLE 3.3: 4-CLASS LABELED DATASET 39
TABLE 3.4: NUMBER OF ATTACK SAMPLES FOR EACH TCP/IP LAYER. 41
TABLE 4.1: PERFORMANCE RESULTS FOR THEANO-KERAS. 45
TABLE 4.3: EVALUATION METRICS FOR PYTORCH-FAST.AI 47
TABLE 4.4: COMPARISON RESULTS OF DL FRAMEWORK. 48
TABLE 4.5: PERFORMANCE RESULTS OF BASELINE FAST.AI. 48
TABLE 4.6: PERFORMANCE RESULTS OF MODIFIED FAST.AI. 49
TABLE 4.7: COMPARISON PERFORMANCE BASED ON TIME/SPEED. 49
TABLE 4.8: COMPARE PERFORMANCE BASED ON MEMORY CONSUMPTION. 50
TABLE 4.9: PARAMETER SPACE AND VALUE OF TABNET. 51
TABLE 4.10: OPTIMAL HYPER-PARAMETER VALUE OF EACH IDS. 52
TABLE 4.11: PERFORMANCE COMPARISON AMONG PROPOSED IDS CLASSIFIERS. 55
TABLE 4.12: COMPARISONS BETWEEN THE PROPOSED IDS CLASSIFIERS AND THE CURRENT IDS CLASSIFIERS. 55

LIST OF FIGURES

FIGURE PAGE

FIGURE 1.1: GENERAL WORKFLOW DIAGRAM. 4
FIGURE 2.1: NETWORK AND HOST-BASED INTRUSION DETECTION SYSTEM. 11
FIGURE 2.2: ARTIFICIAL NEURAL NETWORK (2020 AL-GARADI ET AL). 14
FIGURE 2.3: DEEP NEURAL NETWORKS (BAU ET AL., 2020). 15
FIGURE 2.4: COMPLEX DEEP NEURAL NETWORK (GIESEN ET AL., 2020). 15
FIGURE 2.5: TAB NET ARCHITECTURE VISUALIZATION (GUGGER,2020). 16
FIGURE 2.6: FEATURE TRANSFORMATION COMPONENT (W. LI ET AL., 2019). 16
FIGURE 2.7: ATTENTIVE TRANSFORMATION COMPONENT (W. LI ET AL., 2019). 16
FIGURE 3.1: IOT SYSTEM ARCHITECTURE. 29
FIGURE 3.2: FLOWCHART OF PROPOSED WORK. 30
FIGURE 3.3: GATED LINEAR UNIT LAYER(POCHETTI, 2020). 31
FIGURE 3.4: FEATURE TRANSFORMER(POCHETTI, 2020). 32
FIGURE 3.5: ATTENTIVE TRANSFORMER(POCHETTI, 2020). 33
FIGURE 3.6: DECISION STEPS(POCHETTI, 2020). 34
FIGURE 3.7: ATTACK TRAFFIC COMPOSITION. 40
FIGURE 4.1: ACCURACY RESULTS ON OVER 10 EPOCHS. 46
FIGURE 4.2: ACCURACY RESULTS OVER 10 EPOCHS. 46
FIGURE 4.3: ACCURACY RESULTS OVER 5- FOLDS 47
FIGURE 4.4: OPTIMAL LEARNING RATE. 51
FIGURE 4.5: IMPORTANT FEATURES FOR IDS ON EVERY LAYER. 52
FIGURE 4.6: THE SIGNIFICANCE OF A FEATURE FOR APPLICATION LAYER IDS. 53
FIGURE 4.7: IMPORTANCE OF THE FEATURE FOR TRANSPORT LAYER IDS. 53
FIGURE 4.8: NETWORK LAYER FEATURE IMPORTANCE MASKS. 54

LIST OF ACRONYMS AND ABBREVIATIONS

AE	Auto Encoder
AIDS	Anomaly-Based Intrusion Detection System
AMD	Advanced Micro Device
ANIDS	Anomaly Network Intrusion Detection System
ANN	Artificial Neural Network
AT	Attentive Transformer
AU	Application Unit
AUC	Area Under Curve
BATMC	Bidirectional Attenuation Multiple Convolutions
BLSTM	Bidirectional Long Short-Term Memory
bot-IoT	Internet of Things Botnet
BRCG	Boolean Decision Rules via Column Generation
CEM	Contractive Explanation Method
CIC	Canadian Institute for Cybersecurity
CSE	Communications Security Establishment
DARPA	Defense Advanced Research Projects Agency
DBN	Deep Belief Network
DDoS	Distributed Denial of Service
DGC	Department Graduate Committee
DIDS	Distributed Intrusion Detection System
DNS	Domain Name Service
DOS	Denial-of-Service
DoS	Denial of Service
DS	Decision Step
DSRC	Dedicated short-range communications technology

DVWADamn	Vulnerable Web App
FAR	False Alarm Rate
FP	False Positive
FT	Feature Transformer
FTP	File Transfer Protocol
HFSA	Hybrid Feature Selection Algorithm
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineer
LIME	Linguistic Metadata
LSTM	Long Short-Term memory
MANET	Mobile ad hoc network
OBU	On-board unit
RCP	Resource Command Processor
RSU	Road-Side Unit
V2I	Vehicle Infrastructure Communication
V2V	Vehicle-to-Vehicle Communication
VANET	Vehicular ad hoc network
WAVE	Wireless Access in the Vehicular Environment

ABSTRACT

In recent years, there has been a lot of interest in study of vehicular ad hoc networks (VANETs). VANETs are essential to the development of self-driving and semi-autonomous vehicles since they increase comfort and safety. However, the security risks that are either present in ad hoc networks or specific to VANET pose significant difficulties. VANETs could be targets of numerous attacks. To defend the external communication system from intrusions, this study provides an intrusion detection system (IDS) for VANET that makes use of anomaly detection. In this research, we proposed a deep learning-based Attentive Interpretable Tabular Learning (Tab Net) architecture-based intrusion detection system for VANET. Deep learning Tab Net model exhibits strong interpretability and quick learning rates. The proposed system obtains with an accuracy of 98.12%, precision of 98.86% and a False Alarm Rate (FAR) of 0.78%. The principles of the Tab Net architecture are used for VANET security for the first time in research of VANETs. The developed methodology is tested on the real-time CIC-IDS 2018 dataset and the results of the experiment are contrasted with those obtained using other cutting-edge techniques.

Keywords: *CSE,CIC, IDS 2018 Data set, IDS Fast.ai, Network Security, VANET, Tab Net.*

CHAPTER ONE

1. INTRODUCTION

1.1 Background of the Study

Currently, the world is witnessing rapid product introductions and great expectations from rising technology. These technological gadgets simplify human life, business, and social activities. In the absence of permanent infrastructure, a mobile network known as VANET allows cars to connect with one another with the purpose of improving road safety by transmitting alerts between vehicles. OBU and Road-side unit (RSU)s make up VANET RSU.VANETs are essential to the development and adoption of fully and partially autonomous cars. Systems for internal and external communication are seen to be crucial parts of fully and partially autonomous vehicles. One of the most important goals of VANET is safety. Safety will decrease traffic, save lives, and reduce accidents. Other services, including Internet access, weather forecasts, and geolocation data, can enhance the trip experience in addition to safety by bringing comfort, convenience, and entertainment (Hamdi et al., 2020).

VANET technology is one of the emerging ones that has the potential to significantly alter business and communication practices. It predicts the best traits of people and will happen soon. The broadcast standard for the Dedicated Short-Range Communications Technology (DSRC), Wireless Access in the Vehicular Environment (WAVE), based on the Institute of Electrical and Electronics Engineers (IEEE) 802.11 protocol, has been incorporated by DSRC manufacturers into several cars. VANETs represents the transmission of data within a radio coverage area between vehicles and their RSUs, often known as vehicle-to-vehicle communication (V2V). VANETs enable wireless communication between vehicles through the DSRC, including V2V as well as vehicle infrastructure communication (V2I).

These two forms of vehicle communication are named V2V and V2I, respectively. In a pure wireless ad hoc network, V2V communication occurs without the aid of any permanent infrastructure.

SDN is a new network technology that provides a more flexible and scalable network architecture capable of responding quickly to changes in business and end-user

needs through simplified network management (Mladenov, 2019). Every day, the power of SDN is being shown, from small local area networks to public cloud designs. SDN typically shows remarkable success by offering higher reliability, efficacy, simplicity, and flexibility at a cheaper price. (Kandoi & Antikainen, 2015). Nonetheless, despite its numerous advantages, SDN security remains a source of concern among research communities.

VANETs are susceptible to a variety of attacks since cars are connected via a wireless network. The effectiveness of the VANET is significantly harmed by these attacks. which is extremely problematic for drivers. As a result, preventing VANET traffic from being manipulated, observed, or stripped of vital messages has been a key concern and one of academia's and industry's top concerns. This control over users' data could be utilized by their attacker to damage the network. One practical way for dealing with these difficulties of the VANET might be to design effective security mechanisms using the concept of "lightweight" and "interpretable". Many times, "interpretive Lightweight" solutions have shown their effectiveness in addressing discrepancies in very large, distributed networks. Because there are so many VANET device on a network, designing a security solution for each one is nearly impossible. However, securing data transferred between devices in a VANET network might be a viable approach.

In this thesis, we present a unique study involving a Tab Net design for a VANET system. Then, using DL algorithms, we monitored the VANET network data and classified it as "normal" or "attack" for each layer in the proposed architecture. The CSE-CIC-IDS2018 dataset (Sharafaldin et al., 2018,) which is a benchmark data set used in the majority of network data security ML research, is used in this study.

1.2 Statement of Problem

Due to resource constraints, OBUs on the VANET have limited processing power and storage capacity so A lightweight IDS model is necessary. One of the most difficult issues is to create a lightweight IDS model that is efficient in terms of processing power, training time, and has a greater intrusion detection rate and Encrypted network traffic is a problem/challenge, especially for Network-Based intrusion Detection System (NIDS)s. Other problem is models are tested on older

datasets, such as Knowledge Discovery Data (KDD) Cup '99 and Network Security Learning Knowledge Discovery Dataset (NSL-KDD). If a new IDS method for modern networks is tested with an old dataset, it is more likely to fail. Obviously, a model trained and validated using a newly proposed dataset will outperform a model trained and verified using an older dataset in the actual world. The problems that have been identified/raised in related papers are as follows:

1. Attack detection performance is poor.
2. Feature selection reduces accuracy.
3. On little data sets, multiple classifier methods are run.
4. Execution of various classifiers algorithms on diminished data sets.
5. False positives and false negatives are still an issue.

1.3 General and Specific Objectives

1.3.1 General Objective

The primary goal of this research work is to propose IDS system which use interpretive Tab Net deep learning model to detect old and recent attack with high computational speed, low memory consumption and high accuracy for resource limited systems (OBU).

1.3.2 Specific Objectives

- To modify PyTorch-fast.ai framework by understanding its architecture.
- To assess the performance of models for attack detection and classification.
- To evaluate variables like precision, recall, F-measure, and classification rate accuracy to compare the performance of the proposed approach to that of a different DL algorithm.

1.4 Significance of the Study

- speed up Tab Net model data block fetching by 30% by modifying the DL framework.

- We proposed a lightweight security solution which makes the system dynamic, scalable, and flexible, this would assist the model in increasing convergence speed while maintaining consistency.
- Furthermore, we applied a new method, the TabNet model, that has advanced key features for AI-Based security solutions.

1.5 Workflow

It has begun by conducting a study (review) of the present state of security and privacy in the NIDS and VANET after research gap formulation. The characteristics that are appropriate for designing an VANET solution was then framed (hypothesized). As the next phase, we provided a multilayered strategy that complies with the specified VANET features. We decided to use the VANET network data security issue as a proof-of-concept to create a clever solution. On network data, we applied the DNN technique to categories each sample as either normal or intrusive. we chose the CSE-CIC-IDS2018 benchmark dataset because it allows us to compare the outcome of this work to previous outcomes. Finally, the findings of the proposed VANET design at each layer were compared to the all-layers IDS and the existing one (literature). overall workflow to accomplish the specific objectives of this research are as seen in Figure 1.1 below.

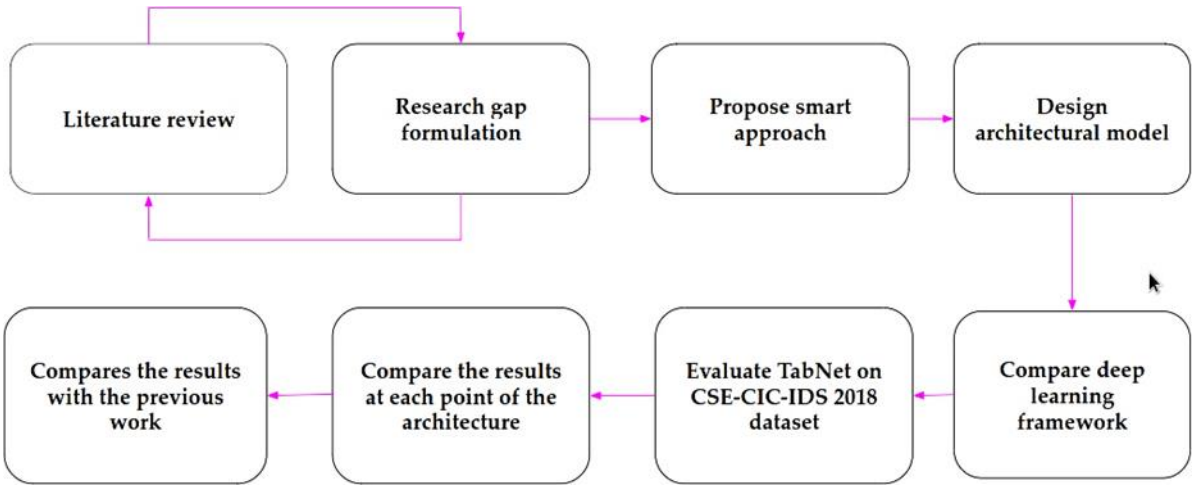

Figure 1.1: General Workflow Diagram.

The methodology followed to complete the task was:

- Review literature on intrusion detection systems, as well as the most recent advancements in machine and deep learning for VANET, to pinpoint a problem and find a solution.
- Recent CSE-CIC-IDS2018 data sets have been downloaded from the CIC.
- Compare the art of DL framework.
- Choose and modify the DL framework.
- Prepare the proposed framework for training.
- Propose a DL model based on the proposed modification for DL framework.
- Propose and design multi layered architecture approach to network security in an VANET system.
- Prepare the proposed model for training.
- Python programming was used to implement the proposed concept.
- Finally, validate the proposed model using different evaluation metrics.

1.6 Delimitation and Limitation of the Study

Because NIDS is such a broad field of study, it is critical to establish some form of job coverage boundary to achieve better results. As a result, the suggested research here is restricted to detection, and classification of an attack-type, but it doesn't include a preventive mechanism for those attacks. Additionally, the research makes network intrusion detection (simple) for a useful re- source constraint VANET device like OBU. We attempted to use the Google Colab environment, but the minimum machine that Colab provided was unable to run our experiments, especially the binary-class and multi-class classification tests, which needed more than 12 GB of random-access memory (RAM).

CHAPTER TWO

2. LITERATURE REVIEW

2.1 Chapter Overview

This chapter discusses the background literature and relevant works that were involved in the development of this research. It introduces theoretical concepts followed by its previous studies. Furthermore, this chapter addresses specific related works. The final section of this chapter also continues to highlight the gap identified in prior investigations.

2.1.1 Theoretical Background

This section contains background information on the evolution of the research. First, the concept of network intrusion and VANET are explained, followed by security and privacy concerns. Following that, there is an explanation of the VANET network. The network architecture of the DNN and Tab Net models is then discussed. The final items in this section highlight the utility of machine learning and deep learning for VANET security by giving examples from real-world applications. The primary objective of this section is to offer the reader a thorough understanding of the concepts and algorithms for those who are new to this field of study.

2.2 Overview of VANET

2.2.1 What is VANET

A VANET is an ad hoc network composed of multiple moving vehicles that communicate with one another via a wireless means and share critical information. Simultaneously, a small network is constructed, with each car acting as a node. Every other node receives the information that each node has. After communicating their own set of data, all nodes receive the data being transmitted by other nodes. Gao et al., 2019 Zeng et al., 2019. Now that new vehicles are being released to the market, they include on-board sensors that allow the vehicle to easily join and merge in the network and profit from VANET. One of the wireless technologies that VANET is expected to deploy is DSRC, a type of Wi-Fi. The three categories of primary systems components in VANET are as follows:

(a) On Board Unit

(b) Application Unit

(c) Roadside unit

(a) **On Board Unit** is Connecting with RSUs or other OBUs requires the usage of a WAVE device known as an OBU. It is frequently mounted on a vehicle. It has a user interface, a unique interface for interacting with other OBUs, a resource command processor (RCP), a read/write memory for storing and retrieving data, a network device for short-range wireless communication based on IEEE 802.11p radio technology, and more. The on-board system must have a user interface that works with driving Users are able to use the services and input data into the system thanks to it. To broadcast internal data and access external services, one needs the VANETS Communications with Externals Elements. To meet the needs for interacting with all the external components, it might be able to support a variety of wireless communication modes Alsarhan et al., 2021.

(b) **Application Unit** (AU) is a piece of hardware that is installed inside the car and communicates with the apps the service provider offers by utilizing the OBU's communication capabilities. Ordinary hardware, such as a personal digital assistant, can be used by the AU, as well as Internet-based security software. It is possible for the AU and OBU to share a single physical unit and to connect to one another wirelessly or over wires. It makes sense to distinguish between the AU and the OBU. The OBU, which oversees all networking and mobility tasks, is the only point of contact between the AU and the network. Gao et al., 2019.

(c) **Roadside Unit (RSU)** A piece of computer hardware known as a roadside unit is permanently installed next to a road or in a specific area, such as a parking lot or crossroads Zeng et al., 2018. Its purpose is to give passing vehicles local connectivity. The IEEE 802.11p radio technology-based specialized short-range communication network devices make up the +e RSU. RSUs have the ability to communicate with other network devices, particularly those connected to other infrastructure networks Zeng et al., 2018.

2.2.2 Types of VANET Communication

Due to the speed of the cars and the characteristics of radio propagation, a vehicular network is very dynamic. With motorway top speeds of 100 km/h and average urban area speeds of 50 km/h, vehicles move at high relative speeds. Topological changes happen frequently and swiftly since the automobiles can join or leave the network quickly. The variability in speed and propensity for asymmetric mobility patterns of nodes must be considered when developing algorithms. There are three primary types of communication or domains in VANET.

(a) Vehicle to vehicle (V2V)
(b) Vehicle to infrastructure (V2I)
(c) Roadside unit to Roadside unit (RSU2RSU)

Vehicle to vehicle Direct vehicular engagement is made possible by V2V communication, which is mostly utilized for dissemination and safety-related objectives. An OBU plus one or more AUs, which could be wired or wireless, make up this domain. To execute one or more sets of application providers using the OBU's communication capabilities, the OBU offers a communication link to the AU Gao et al., 2019.

Vehicle to infrastructure the capacity of a vehicle to interact with the infrastructure along the side of the road enables information and data gathering primarily. Data is transferred from one vehicle to another via a specialized routing protocol until it reaches the destination point when there is no direct connection between them. This is known as multi-hop vehicle to vehicle communication. By transmitting, receiving, and transferring data from one node to the RSU to carry out specific applications, the automobiles communicate with the RSU to extend the range of communication.

RSU to RSU provides for direct communication between two RSUs that are close to one another and is typically used when there is a need to send a V2I movement, information about a severe traffic jam, or an urgent safety message. Because RSU is overseen by the Trusted Authority directly as opposed to V2V/V2I, it is always

recognized as the trusted infrastructure. The following RSU2 re-receives the information and sends it to any route car nodes when a jam occurs within the coverage area of RSU1. A significant event that takes place during RSU-to-RSU communication is the handoff procedure, in which a vehicle from one RSU range joins another RSU or a new RSU.

2.2.3 Security Attacks and Threats in VANETs

Information accessibility is a crucial component of the VANET system since a lack of this characteristic could reduce the effectiveness of VANETs. We will describe the dangers and assaults in VANETs in this part.

(a) Denial-of-Service (DOS) Attacks: The attacker jams vehicle-to-vehicle communication and ultimately shuts down all channels of communication, is one of the frequent attacks in VANETs.

(b) Jamming Attack. Attacker uses a powerful signal with an analogous frequency to disrupt the communication channel in VANETs. Because it disregarded a legitimate safety alert, this attack is the most hazardous for safety applications. For any successful jamming attempt, the jammer can jam the valuable signal at the exact moment that an event occurs by taking some sort of action.

(c) Malware Attack: The VANET system can be breached by this assault using the wireless communications software and Mobile Computing components.

(d) Blackhole Attack: This is the primary assault that attacks availability in both VANETs and ad hoc networks. Typically, a registered VANET user is the one who launches this attempt. Although the suspect node decides not to engage in networking activity, it accepts packets from the network. Due to the malicious node, which pretends to contribute to the impractical event, this may disturb the routing table and prevent the necessary message from reaching the recipients.

2.2.4 Encrypted Internet Traffic

The execution of network traffic encryption has vastly enhanced communication security and user privacy. When trust technologies, such as Transport Layer Security (TLS), most internet users assume/think that third parties will not be able to access to their communications and companies may be certain that their transaction will be free of interference and eavesdropping. However, pervasive network traffic encryption has restricted network managers' capacity to monitor their infrastructures, limiting their ability to cope with hostile traffic and sensitive data ex-filtration, requiring them to rely on traffic decryption via proxies.

2.2.5 Intrusion Detection System (IDS)

In 1980, Jim Anderson introduced the idea of IDS for the first time. An IDS is a piece of software that keeps an eye out for suspicious activity on networks and systems and can aid in network security. IDS are classified as either active or passive depending on how responsive they are. A passive IDS only monitors network activity and informs users, but an active IDS is designed to stop malware attacks without the involvement of a human. Signature-based and anomaly-based IDS are two other types of IDS. The IDS employs the signature-based method, which involves accessing a database of acknowledged vulnerabilities and signatures. Every intrusion attack carries a signature that includes all the attack's details and is used to recognize and stop later attempts the database needs to be updated, which is this method's biggest flaw. often, as opposed to the behavior-based anomaly-based IDS, which detects anomalies by learning from baseline patterns to identify new intrusion threats. Alarms are set off whenever there are deviations from the baseline, which are categorized as attacks. Depending on its placement, the IDS can also be categorized shown in Figure 2.1, an IDS installed on a network segment is known as a NIDS; It is referred to as a Host-based intrusion detection system when installed on a workstation. Host-based IDSs might not be appropriate for research due to several drawbacks.

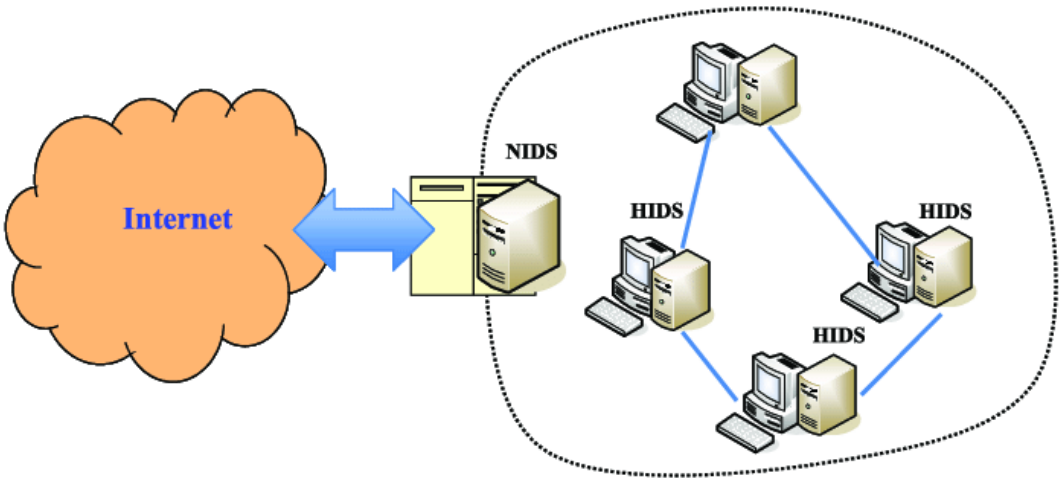

Figure 2.1: Network and Host-Based intrusion detection system.

2.2.6 Machine/Deep Learning Framework

Machine/DL frameworks are interfaces, libraries or tools that enable us to construct and deploy machine/DL models more simply and rapidly, without diving into the underlying techniques. They offer a straight- forward and succinct method for defining a model by utilizing a set of pre-built and optimized components. Instead of creating hundreds of lines of code, we may utilize an appropriate framework to assist us in speedily building such a model. A good DL framework should be optimized for performance, easy to comprehend and code, have excellent community support, parallelize processes to decrease computations, and automatically compute gradients.

Keras Framework: It is a portable Python-based high-level neural network API that may be used as a back end for Theano and Tensor-flow. It was initially developed as an Open-ended Neuro- Electronic Intelligent Robot Operating System component. For research and development, this interface offers a variety of advantages, with mobility being the most important. Switching between them just requires minor tweaks because Kera's was designed to support three major DL frameworks and maybe more in the future.

Tensor-Flow: It is a Google-created open-source platform that is widely utilized in commerce. It has a strong, flexible ecosystem of tools, libraries, and community resources that enables researchers to push the boundaries of machine learning and

developers to quickly build and distribute ML-powered apps. This framework supports Python, Java, JavaScript, C++, and the Internet of Things.

Theano The Python-based DL framework stands out for its speed. When coupled with a contemporary Graphical Processing Unit (GPU), this library's speeds surpass those of custom C programmers (Giesen et al., 2020 and Bergstra et al., 2012).

PyTorch: Due to its extensive support and focus on developing new features, PyTorch is a DL framework that is well-liked in academia and business. The largest cloud services are supported by this programming platform, which also permits extensive customization and optimization. Additionally, PyTorch can be used in C++, enabling further model building and optimization (Stevens et al., 2020).

Fast.ai: fast.ai The goal of Howard and Gugger, 2020 was to increase the accessibility of DL solutions for academics and developers from a variety of backgrounds. It uses PyTorch as its back end, which is another well-known deep learning system covered in the following section. For DL researchers, Fast.ai optimizes PyTorch and streamlines the experimental process.

The main reasons these frameworks were selected were their usability and applicability. Fast.ai and Keras-Tensor-Flow are both open-source and rigorously documented to reduce the time and effort needed to go from research to implementation. Fast.ai boasts of having training scripts that are only four lines long, demonstrating extreme simplicity. While Tensor-Flow includes numerous case studies from significant organizations like Intel, PayPal, GE-Healthcare, ARM, Twitter, etc., Howard and Gugger,2020. While Thea-no boasts incredible speeds, PyTorch displays its potential through its standing in the research communities. All these frameworks are also regularly updated and maintained. We determined that these frameworks were suitable to show how simple IDS training with DL.

2.2.7 Machine Learning

Machine learning (ML), a subfield of artificial intelligence, researches and develops systems using data knowledge. Howard and Gugger, 2020. There are many uses for machine learning, including regression, classification, prediction, and other tasks. There are three types of learning based on the use of labelled data: supervised, unsupervised, and semi-supervised. Logistic regression, Navies Bayes (NB) classifier, support vector machines (SVM), artificial neural networks (ANN), and other techniques are frequently used in machine learning (ML).

2.2.8 Deep Learning

DL is a sophisticated kind of ML that involves various degrees of data abstraction at various processing layers. Through back-propagation, DL can learn the complex dataset structures, and it shows how the machine modifies its internal parameters at each layer. Deep Belief Network (DBN), Convolutional neural networks (CNN), recurrent neural networks (RNN), and auto-encoder (AE) are some of the regularly utilized DL algorithms. A more complex kind of machine learning that uses varying levels of data abstraction across various processing layers Educating, 2020. Through the process of back-propagation, deep learning may learn the complex patterns in the dataset. The deep learning methods DBNs, AEs, CNNs, and RNNs are frequently used.

2.2.9 Why is DL better than ML for IoT

DL, also known as hierarchical learning, is a more advanced variant of ML in terms of data representation's structure and learning the fundamental difference between ML and DL is how performance changes as data amount increases. While ML requires the least amount of data, deep learning approaches require more data to identify patterns in the network. The structure and processing of the data at each layer will be deeper in an ANN with one or more hidden layers, resulting in deeper learning Popular architectures (DL) used in research. These include DBN, DNN, and RNN. When studying IoT data that is diverse and multimodal, DL may also be used. VANET devices that are frequently connected for extended time-of-live do not benefit from ML approaches in the long run. In DL, Tab Net can use previous

decision step learning to inform its next decision without requiring human involvement (2020 Al-Garadi et al).

2.2.10 Artificial Neural Network

To store and assess which input is crucial to the output, ANNs use a hidden layer. The hidden layer maintains relationships between the significance of input combinations and information about the relevance of input (Abiodun et al., 2018). Simple ANN are as shown Figure 2.2.

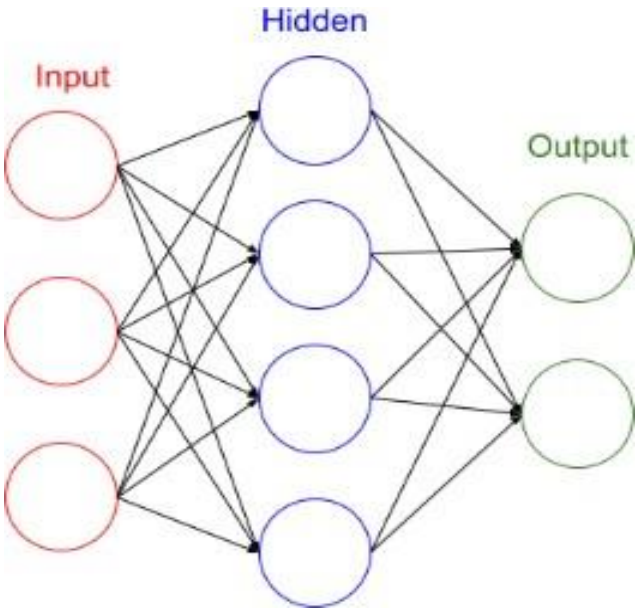

Figure 2.2: Artificial Neural Network (2020 Al-Garadi et al).

2.2.11 Deep Neural Network

Figure 2.3 shows that DNNs, on the other hand, utilise the ANN component. They contend that if that is so effective in updating/refining models. Because each hidden layer node establishes associations and assesses the appropriateness of the input to determine the output. Consequently, the deep web has a number of hidden levels. (Bau et al., 2020).

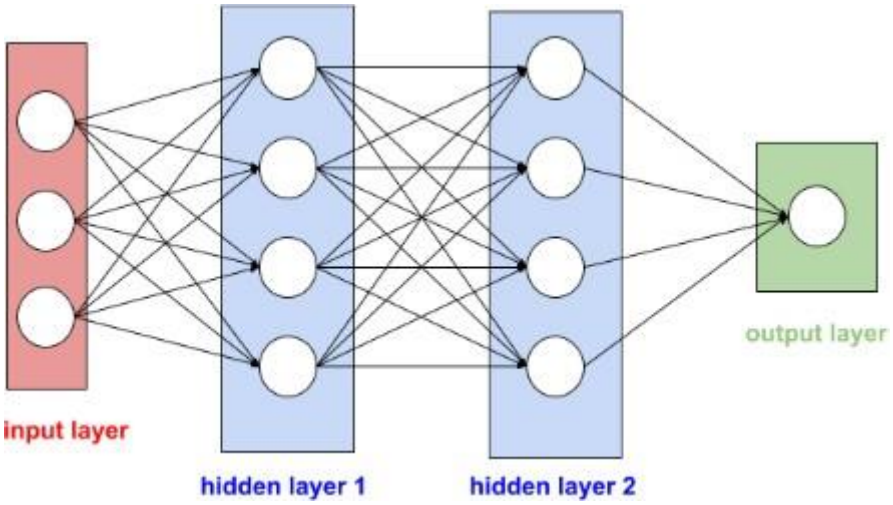

Figure 2.3: Deep neural networks (Bau et al., 2020).

2.2.12 Tab Net DNN

Tab Net is a DNN for tabular data and was designed to learn in a similar way than decision tree-based models, in order to have their benefits: interpretability and sparse feature selection. Since the learning capacity is applied to the most crucial elements, Tab Net uses sequential attention to choose the features from which to draw conclusions at each level of decision-making, improving interpretability and better learning. (Habibi et al.,2020).

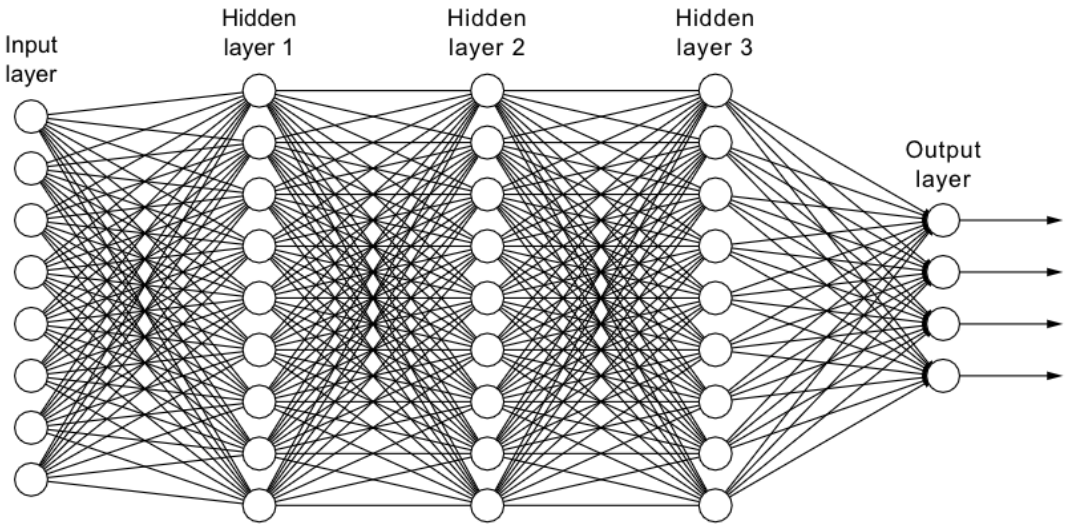

Figure 2.4: Complex deep neural network (Giesen et al., 2020).

The fact that multi-layered Tab Net shares Key elements are its hyperparameters, weights, and biases across layers to achieve optimal performance. Figure 2.5 shows

abstract view and components such as Feature Transformer (FT) and Attentive Transformer (AT) in 2.6 and 2.7, respectively. Their functions are discussed in subsection 3.1.5

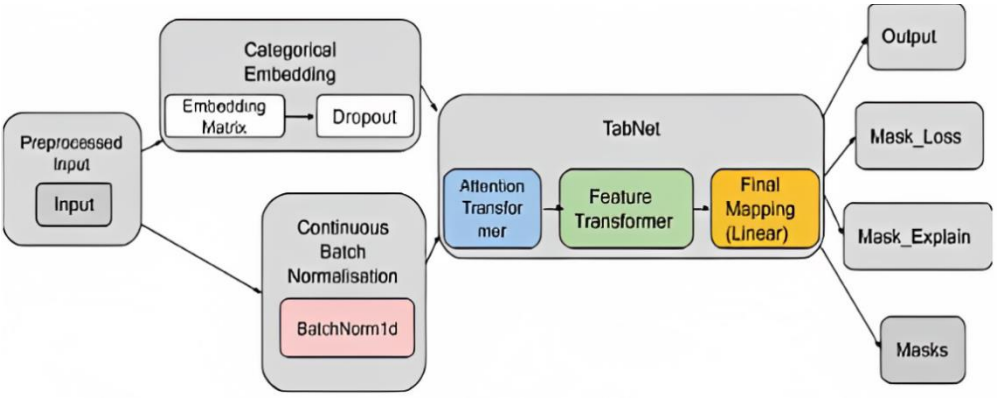

Figure 2.5: Tab net architecture visualization (Gugger,2020).

2.2.13 ML and DL for the Domain of Cybersecurity

Due to the rapid growth of cyber assaults and the availability of enormous volumes of harmful data in cyber infrastructures. ML may be used to recognize signatures, detect anomalies, detect scans, profile network traffic and mining data while maintaining privacy.

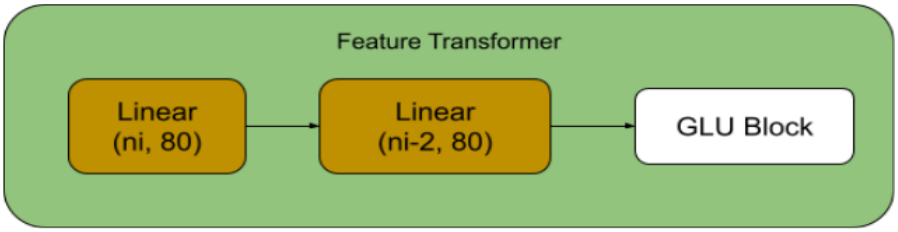

Figure 2.6: Feature transformation component (W. Li et al., 2019).

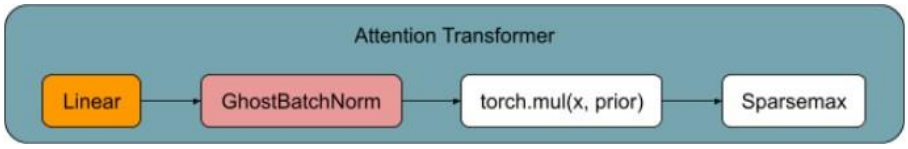

Figure 2.7: Attentive transformation component (W. Li et al., 2019).

2.2.14 ML and DL Applications on IDS

In the past, many researchers have employed automated techniques to find attacks. A supervised active learning system for intrusion detection was developed by (Y. Li and Guo, Y. Li and Guo, 2007). Their study achieved a false the false positive rate

in their study was 0.027, which can be improved Pu et al., 2020. On the KDD cup' 99 test datasets. (Shafak et al., 2019) obtained an accuracy of 84.12% and 68.62% using a Fuzzy-based semi-supervised approach. Although it is the most prevalent data type in real-world AI (since it is made up of any categorical and numerical variables) (Bughin et al., 2018), ensemble Decision Tree (DT) variations due to following reason (i) They are representationally effective for decision manifolds with roughly hyperplane bounds, which are common in tabular data. This is a problem in many real-world applications because they are easily interpreted in their basic form (for example, by tracking decision nodes) and have popular post-hoc explain ability methods for their ensemble form.

(ii) they are fast to train. DBNs, AEs, and limited Boltzmann machines are frequently employed to extract features from intrusion detection data sets. (Tao et al, 2016) Extracted the critical features necessary from the KDD cup '99 dataset using a supervised Fisher and Deep AE Tao et al., 2016. According to Devaraju and Ramakrishnan (2014), the same team produced better results with training accuracy and cost of the network being respectively 22.13% and 93.82%. Prior studies have all concentrated on small-sized random records, which are inappropriate for deep RNN. By using the TabNet DNN throughout the entire dataset, we tried to overcome this restriction given that it includes both categorical and numerical properties, tabular data is the most prevalent data type in real-world AI, yet DL for tabular data remains under-explored et al., 2018.

2.2.15 Machine and Deep Learning Framework on IDS

In the light of the anticipated acceleration and growth of computer threats, this paper analyses the value and promise of deep learning algorithms in the critical field of network intrusion detection. using the recent dataset CSE-CIC-IDS2018.Its employed and contrast three cutting-edge deep learning frameworks (Theano, PyTorch, fast.ai) for classifying typical network attack types and recognizing network intrusion data Our findings demonstrate the usefulness of various deep learning (DL) frameworks for identifying network intrusion traffic, with Fast.ai highly opinionated wrapper for PyTorch providing the highest detection and

categorization accuracy of about 99% with the least number of false positives (FP) and false negatives.

2.3 Related Works

Recently, several evaluations of the literature on machine learning techniques for intrusion detection have been published, some of which have a particular focus on deep learning techniques. The previous studies related to NIDS are discussed as follows:

(Su et al., 2020) Using the NSL-KDD dataset, Bidirectional Attenuation Multiple Convolutions (BATMC) for intrusion detection were developed by learning Bidirectional Long Short-Term Memory (BLSTM) in two phases. (Zhang et al., 2019) used a deep AEs model. It was compared with other deep learning, which was applied to intrusion detection method evaluations on the NSL-KDD dataset. The experiment shows better performance than the previous classification model such as RF and SVM.

(Solanki et al., 2020) developed a method for using data mining in network security settings such as intrusion detection framework, where Hybrid Feature Selection Algorithm (HFSA) is combined with the NB multinomial techniques. Additionally, they assess classification performance according to classification accuracy, precision, and recall, all of which are characteristics of the most recent identification methods. These maintain distribution execution which also significantly reduces computing time and cost.

To establish a safe network, Mohan et al. applied data mining classification algorithms for intrusion detection in the following study. For classification, the Random Tree (RT), NB, J48, and RF ML classifiers are employed. Comparison of strategies based on factors such as data accuracy, detecting interesting patterns, and extracting usable information (Mohan et al., 2020).

Shah et al. study presents multi-class Employing different machine learning techniques and neural networks, classification baselines may distinguish between legitimate network traffic and overt and covert network attacks. The Advanced Security Network Metrics and Tunnelling Obfuscation (ASNMTO) dataset provided

the baselines for this study. The dataset gathered both legitimate and spoofed malicious TCP/IP traffic on selected vulnerable network services.

Kumar and Singh, the paper suggests employing cutting-edge and promising technologies like blockchain to build a Distributed Intrusion Detection System (DIDS) on a dependable platform like cloud infrastructure. Additionally, it covers the research's use of the cloud computing infrastructure and block-chain in the DIDS architecture. DIDS server's performance with various load of data are shown. There is other more aspects to consider, like Communication lag, block-chain overhead, and the expense of implementation, and so on, which must be considered and analyzed (Kumar and Singh, 2020).

C. Chen et al. use the Grey Wolf Optimizer approach based on Particle Swarm Optimization (PSOGWO) method to improve the overall performance of intrusion detection based on SVM. The results show that the technique outperforms the baseline in terms of network intrusion detection. The "optimal detection model" of the SVM classifier is produced by combining the PSOG-WO and SVM algorithms (C. Chen et al., 2021).

"Dutt et al." Study offers Statical Modelling based Anomaly detection (SMAD) as the first layer of an IDS. It acts as the Innate Natural Immune System's (IS) interface, gathering the initial network information needed to pinpoint the vulnerability. Utilise the second layer, adaptive immune-based anomaly detection (AIAD), to identify the features of suspicious network packets for anomaly identification. It mimics the adaptive immune system by using T- and B-cells. cell energization for effective intrusion detection, studies were conducted on both real-time network traffic and the standard datasets KDD 99 and University of New South Wales (UNSW)-NB15. It pulls pertinent information from the header and payload portions. Using real-time traffic and common data sets, the SMAD model achieves a true positive rate of up to 96.04% and a true positive rate of over 97%. The AIAD model is used to assess the ability of the SMAD model to identify defects in highly suspect communications. In both real-time traffic and baseline data sets, the results demonstrate a high true positive rate of around 99 percent accuracy in identifying file-based and user-based anomalies (Dutt et al., 2020).

Swarna Sugi and Ratna article explains an IDS that detects and prevents security breaches in IoT networks using deep learning and machine learning. The attack detection model uses LSTM and KNN, and its performance is assessed using sensitivity, geometric mean, kappa statistic, and detection time. The effectiveness of the developed IDS is evaluated using Internet of Things Botnet (bot-IoT) datasets. (Swarna Sugi and Ratna, 2020).

In the paper Mane and Rao, 2021, Mane and utilized DNN to detect network breaches and offered a comprehensible AI framework to provide transparency throughout the machine learning process. This is achieved by utilizing Explainable AI algorithms, which aim to reduce the black box nature of ML models by elucidating how predictions are generated. Explanations give us quantitative parameters that affect the likelihood of a cyberattack and how severe it will be.

Marino et al. present a method for creating defenses for inaccurate classifications made by data-driven intrusion detection systems. In this study, a Generative Adversarial Network (GAN) is used to determine the smallest alterations (in the input features) necessary to correctly identify a given collection of erroneously classified examples. The significance of such changes is utilized to highlight the key explanations for the misclassification. The suggested technique offered sufficient explanations of the misclassifications' causes, with descriptions that are in line with professional understanding (Marino et al., 2018).

Although there are a number of excellent surveys that cover the growing body of work on this topic, there isn't a complete comparison of the various deep learning models in the literature, particularly for more recent intrusion detection datasets.

2.3.1 Summary and Critics

The literature analysis revealed that the current intrusion detection systems had issues with low accuracy, poor detection efficiency, a high false-positive rate, and an inability to handle new types of intrusions. OBU devices' constrained energy and computing capacity prevent them from implementing additional security features.

Table 2.1: Summary of related work 1.

Reference	Statement of the Problem	Proposed System	Feature Work	Result
(Su et al.)	Low accuracy and obsolete in the age of big data	BATMC	✗	Acc=84.25%, FP, FN
(Zhang et al.)	Requirement of adapt-ability and efficiency	Encoder of deep AE	Improve accuracy by modify classifier	Acc=79.74%, Prec=82.22%
(Solanki et al.)	To get NIDS' state of art	Hybrid combination of SVM and ML	Improve detection accuracy by modify classifier	✗
(Mohan et al.)	Large network traffic da-taset leads irrelevant fea-ture	Data Mining techniques	Optimization	Acc=90-99.78%
(Shah et al.)	Obfuscated network at-tack	AIDS	Hyper-parameter Optimization	Acc=95%
(Kumar and Singh)	Emerging Heterogeneous technologies and techniques need robust IDS	DIDS	✗	✗
(C. Chen et al.)	Poor classification Performance	PSOGWO	✗	ac=96%
(Dutt et al.)	Attack Payload Detection	IS-IDS	✗	Ac=97.1%,FP=2.7% TP=99%
(L. Chen et al.)	CNN complexity	CNN-Based novel NIDS	✗	Dete acc=96.81%, Test acc=99.56%
(Swarna Sugi and Ratna)	Improving security of emerging technology	DL-BasedIDS	✗	ac=92.3%,*UI*=
(Mane and Rao)	More complexity leads less interpreteablity	Explainable AI Framework	✗	Acc,FN,FP

(Marino et al.)	Black-box of DL based NIDS	GAN	✗	accuracy,FN,FP
(This Work)	1. OBU constraint 2. DL complexity	Multilayered Approach	class Balance	Acc,FN,FP

Table 2.2: Summary of related work 2.

Author	**Algorithm used**	**Dataset**	**Decryption Technique**	**Optimization**	**Metrics**
(Su et al.)	BLSTM	NSL-KDD	✗	✗	Acc,FP,FN
(Zhang et al.)	AE	NSL-KDD	✗	✗	accuracy
(Solanki et al.)	SVM+ML	NSL-KDD	✗	✗	accuracy
(Mohan et al.)	NB,RF,RT,J48	NSL-KDD	✗	✗	ac
(Shah et al.)	ANN,DT,KNN,RF, and SVM	ASNMTO	✗	✗	accuracy
(Kumar and Singh)	Block-chain	✗	✗	✗	accuracy
(C. Chen et al.)	PSOGWO	NSL-KDD	✗	PSOGWO	Accuracy,training time
(Dutt et al.)	SMAD,AIAD	KDD99 UNSW-NB15	✗	✗	accuracy
(L. Chen et al.)	CNN	CIC-CSE-IDS2017	✗	✗	accuracy
(Swarna Sugi and Ratna)	LSTM,KNN	bot-IoT	✗	✗	Detection time,geometric mean

(Mane and Rao)	SHAP,LIME,CEM	NSL-KDD	✗	✗	accuracy, shape value, weight
(Marino et al.)	GAN	NSL-KDD	✗	✗	accuracy, FN, FP
(Nguyen et al.)	DNN	CIC-CSE-IDS2018	✗	✗	accuracy, FN, FP
This Work	**TabNet**	**CIC-CSE-IDS2018**	*Nondecryptio*	**Bayesian**	accuracy,FN,FP

Table 2.3: Summary of related work 3.

Author	**Frame-work**	**Key Security features of VANET component**			**Key features of AI-Based security**		
		Lightweight	*Distributed*	*Longevity*	*Interpretative*	*Interpretive*	*Attentive*
(Su et al., 2020)	Keras	✗	✗	✓	✗	✗	✓
(Zhang et al., 2019)	Keras	✓	✗	✓	✗	✗	✗
(Solanki et al., 2020))	Keras	✗	✗	✗	✗	✗	✗
(Mohan et al., 2020)	Keras	✓	✗	✗	✗	✗	✗
(Shah et al., 2020)	Keras	✗	✗	✗	✗	✗	✗
(Kumar and Singh, 2020)	Keras	✗	✓	✗	✗	✗	✗
(C. Chen et al., 2021)	Keras	✗	✗	✗	✗	✗	✗
(Dutt et al., 2020)	Keras	✗	✗	✓	✗	✗	✗
(L. Chen et al., 2020)	Keras	✓	✗	✗	✗	✗	✗
(Swarna Sugi and Ratna, 2020)	Keras	✗	✗	✗	✗	✗	✗
(Mane and Rao, 2021)	Keras	✗	✗	✗	✗	✓	✗
(Marino et al., 2018)	Keras	✗	✗	✗	✗	✓	✗
(Nguyen et al., 2022)	**keras**	✗	✗	✗	✗	✗	✗
This Work	**Fast.ai**	✓	✓	✗	✓	✓	✓

2.3.2 Identified Gaps

Some of challenges (problems) the researcher doesn't address as shown at subsection 2.3.1 are.

(a) Gap 1: The shortcomings of earlier intrusion detection methods include low accuracy, poor detection efficiency, a high percentage of false positives, and an inability to handle new types of intrusions.

(b) Gap 2: Due to the resource constraints, a lightweight IDS is required (Implementation of advanced security features cannot be afforded by small device.

(c) Gap 3: IDS which use DL algorithms are untrusted as reasons behind their decisions are unknown.

(d) Gap 4: Encrypted network traffic can contain malicious information that cannot be detected by IDS, especially for network-based intrusion systems.

CHAPTER THREE

3. MATERIALS AND METHODS

3.1 Chapter Overview

This chapter discusses the proposed approach that are used to accomplish the research including system design approach, steps in the experiment and its setup, and finally model evaluation techniques and metrics.

3.1.1 System Design

This section presents a summary of the proposed design approach for developing intelligent models for VANET networks. It provides a clear security solution feature. Specifically, key features of VANET's OBU and core features of AI-Based security such as adaptable lightweight, multilayered/distributive and interpretability, explain ability and attentiveness, respectively. In this section, its present the innovated VANET network architecture that reduces complexity for the IDS classifier and the working concept of the proposed TabNet DNN.

3.1.2 Key Features for VANET's OBU Solutions

The data flow is protected by confidentiality, integrity, and authentication services, and the system is secured against disruptions and intrusions as part of the comprehensive VANET security solutions. Standard solutions may not be successful in an VANET system since it deals with a variety of multimodal data collected throughout time using several devices. It generates an interest in creating scalable solutions for usage on hardware with different memory capacity. As explained below, we came up with the critical aspects that are precisely necessary to manage VANET systems:

3.1.2.1 Lightweight

Because such devices run a low-end Operating System (OS), VANET's OBU devices, un- like smartphones and laptops, are incapable of executing anti-malware software. They lack the capacity to undertake innovative approaches to guard against sophisticated malware attacks, and additionally, they don't have enough RAM to store the malware's growing list of data storage.

3.1.2.2 Distributive/Multilayered

The variations in VANET's endpoint device capabilities underscore the concept of a multi-layered approach in VANET design. The VANET system is highly vulnerable to data, security and privacy, leaks because to its open conventional design. The system's robustness is enhanced by the multi-layered architecture's management of equipment and data at various levels. Data is generated by numerous types of devices in an VANET network processed, saved, and transported to numerous locations in different methods.

3.1.2.3 Longevity

The lifetime levels of the VANET's OBU differ from those of ordinary portable consumer gadgets. For instance, the device's life cycle in a smart city application would be 3 to 5 years. As a result, VANET solutions intended for those devices(products) should be able to continually learn from previous events without the need for human intervention. Applications for the VANET must function without the manufacturer and have long life cycles. Since it boosts Return on Investment (ROI), longevity is essential for creating demand for OBU system devices. Overseeing the maintenance of installed devices in VANET environment for an extended period is expensive for any organization.

3.1.3 Advanced Features of AI

Complex AI-based security solutions defend data flow with integrity, confidentiality, and authentication services while also protecting the system from disturbances and attacks. Standard solutions might not work in an AI-based system because DL solutions come in many different varieties. Smart solutions that can be interpreted by different deep network layers must be developed. To manage the AI-Based systems as detailed below, we developed advanced security measures.

3.1.3.1 Interpretability

The degree to which a machine learning model can accurately link a cause to an effect is known as interpretability. Modern ML algorithms are becoming more common in automated decision making, with applications in both the corporate and governmental sectors. As this trend continues, ML algorithms are being deployed with less and less human oversight, especially in areas where their conclusions may

have serious consequences for people’s lives. Among these fields are computerized credit scoring, medical diagnosis, employment, and autonomous driving, to name a few. Simultaneously, ML systems are growing increasingly complicated, making it challenging to study and comprehend how they arrive at conclusions. This growth in complexity—and the resulting lack of interpretability poses a fundamental difficulty for deploying ML algorithms in high-stakes settings, interpretability to enable assurance in modern machine learning systems.

3.1.3.2 Explainable

Explain ability has to do with how well the parameters, which are frequently hidden in Deep Nets, can defend the outcomes. IDSs that employ DL algorithms are currently not trustworthy because it is unclear what factors led to their judgements, it provides a Tab Net model that aids in obtaining crucial information for decision-making, instance-specific justifications, and relationships between features and system outcomes. The final decision is made by the network analyst by considering the model's prediction. Data scientists can learn what patterns the model has learned by looking at the explanations; if learned patterns are incorrect, they can choose a different set of features or make change in the dataset. In addition to prediction, we offer explanations where comparable cases are displayed for which the model's forecast is identical to the test instance. This allows the network analyst to recognize similarities between the examples and come to a final choice. End-user explanations aid in their understanding of which elements contribute to decision-making with what weight. to adjust the model's judgement by changing the values of the characteristics.

3.1.3.3 Attentive/Attentiveness

Selective attention is the process of focusing our attention on significant environmental stimuli while ignoring unimportant ones. Selective attention helps us weed out unimportant information and concentrate on what is important since there is a limit to how much information can be processed at any given time. This constrained attentional capacity has been compared to a bottleneck that obstructs the information flow. The bottleneck narrows as flow rate decreases.

3.1.4 VANET System Architecture

The requirements for VANET and AI-based security solutions were taken into consideration as we created an adequate architecture to thoroughly track intrusion detection operations. In a typical wireless system, an IDS either uses a signature-based approach or an anomaly-based method to investigate network data. All network data is gathered by the IDS deployed at a network node, which then classifies it as "normal" or "attack in addition to traditional methods, ML algorithms are applied to a dataset, and supervised learning is used to achieve categorization. However, this outdated method might not be suitable given the variability of VANET network architectures. The best intrusion detection security systems will be portable, multi-layered, and reliable. To increase performance, we built a multi-layered design and used Tab Net.

The VANET system is made up of numerous vehicles that are dispersed over a large area. To handle network data from all devices and respond quickly, a single IDS system needs to have enough memory due to the large number of devices and the great distance between them, the VANET network's performance would be poor in this case. To substitute one IDS for the entire system, we developed an architecture that enables four IDS based on malware attacks at each TCP/IP layer. Equipment included at the transport and network levels of the TCP/IP protocol, respectively, includes switches and routers The architecture is seen in Figure 3.1 below:

The "lightweight" and "longevity" properties must now be the main considerations in the creation of a VANET security solution. First, we chose an anomaly detection strategy, which we intended to implement and test several memory-effective ML methods. But when conventional ML algorithms are used, they are unable to achieve "Longevity" since they are unable to learn from the past. The creation of lightweight solutions that can learn through selective attention was made possible by the introduction of DL techniques like Tab Net DNN. As a result, the Tab Net DNN deep neural algorithm can satisfy both the essential characteristics of VANET solutions, "lightweight" and "longevity," as well as explain ability. In chapter 4, the findings at each layer are further discussed and contrasted to those at the single layer IDS.

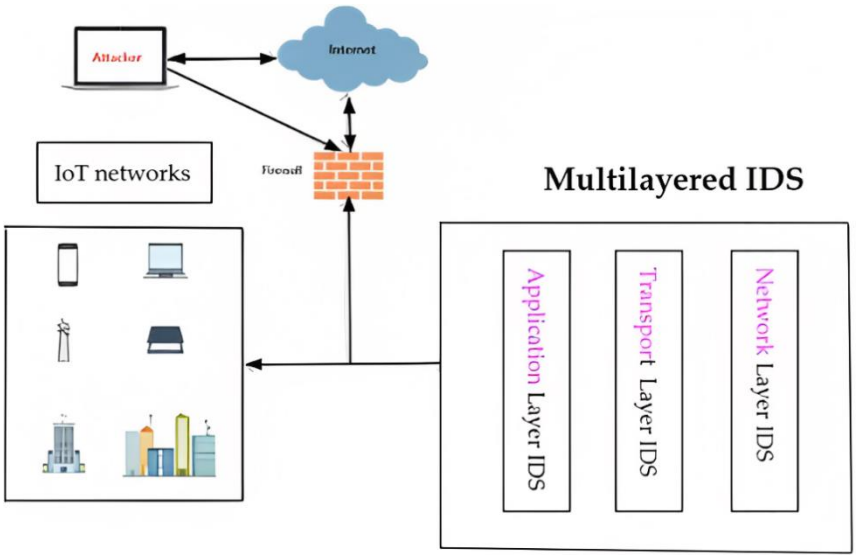

Figure 3.1: IoT system architecture.

3.1.5 Tab Net: Attentive Interpretable Tabular Learning Model Architecture

Since the neural network was founded on the extension of the perceptron, DNNs can be conceptualized as neural networks with numerous hidden layers, as was previously indicated. In terms of images, text, and audio, DNNs have been quite successful recently (He et al., 2016; Conneau et al., 2016; Amodei et al., 2016). For tabular data sets, however, ensemble tree models are still often used. These are dependent on:

• A decision manifold in the tree model (Polzlbauer et al., 2008) roughly approximates the hyperplane's border. Tabular data can be effectively represented in the tree model by effectively dividing the data along the border of the hyperplane.

• Good interpretability.

• Fast training speed.

Second, tabular data does not work with the DNN structure that was originally suggested. Conventional DNNs that use convolutional layers or multi-layer perceptions usually have too many parameters for tabular data.

They are insufficiently inductive, which prevents them from identifying the decision manifold for tabular data. Dependence on feature engineering is the fundamental drawback of the DT and its variation models. The ability of deep learning (DL) to

encode raw data into meaningful representations is a crucial factor in why these three areas are so successful for deep learning approaches. Tabular data can be efficiently encoded using end-to-end training using the back propagation technique, which can reduce or even do away with the requirement for feature engineering. Additionally, the expressiveness of the neural network model could operate better when the data set is bigger. Figure 3.3 shows that the TabNet encoder architecture is mainly composed of a FT, an AT, and feature masking at each decision step. The key components of the TabNet encoder architecture are an FT, an AT, and feature masking at each decision step. Both categorical and numerical data are included in the tabular data. To translate category features to numerical features, TabNet leverages original numerical data and trainable embedding (Grabovac and Cheng, 2018). Each decision step inputs the same B × D feature matrix; Input is the dimension of the feature, and BS is the batch size.

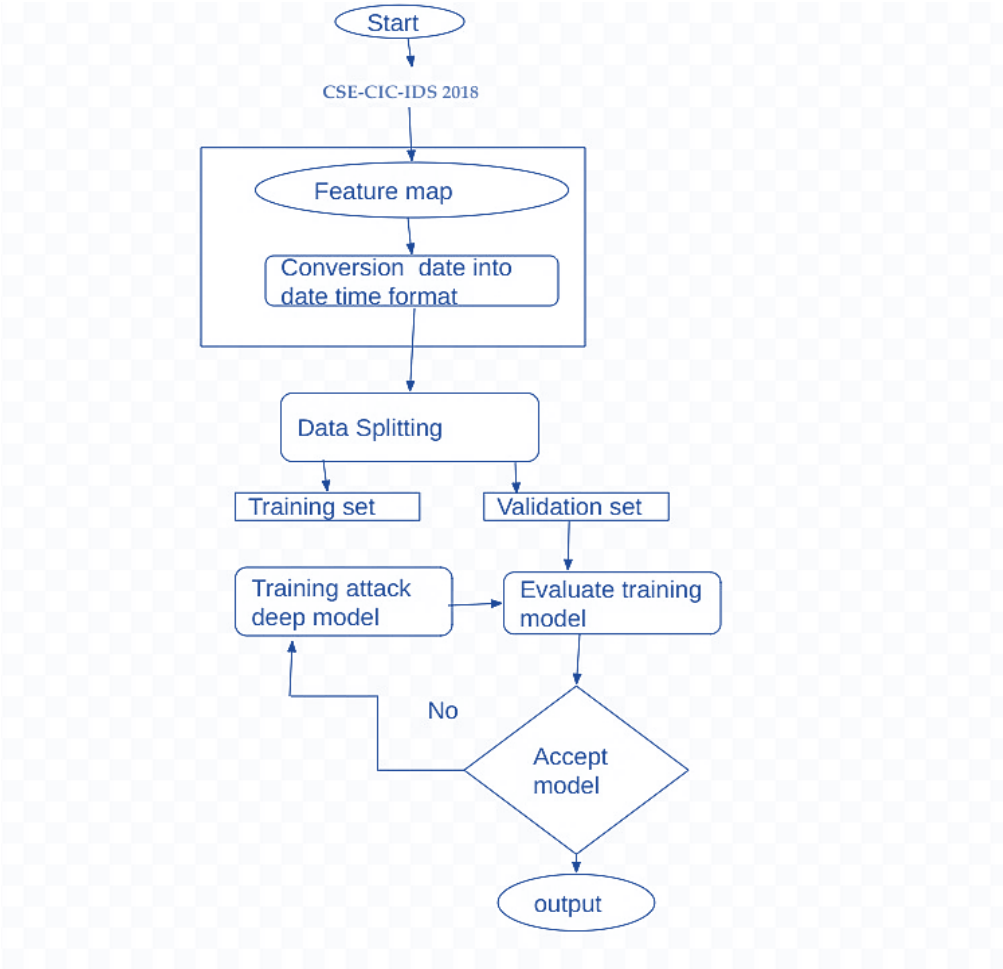

Figure 3.2: Flowchart of proposed work.

The processing of numerous decision steps serves as the foundation for TabNet's encoding. The output of the preceding decision step through the AT determines the characteristics of each decision phase. The output of this incorporates the feature representation that has been processed into the overall decision-making process.

3.1.5.1 Gated Linear Unit Layer

The Gated Linear Unit (GLU) layer, which divides a tensor of size nd + na from an input tensor, is the initial component of TabNet. The block is made up of a Fully Connected (FC) layer that maps the input to the end value of 2(nd + na), followed by the Ghost Batch Normalization (GBN) and GLU., which haves the shape to the final nd + na. The batch is divided into virtual BS pieces, each of which is subjected to conventional Batch Normalization independently before being concatenated back to the original batch. The generalization gap, which results in neural networks performing poorly for untrained data when trained on high batch sizes, is fixed by the GBN. By employing tiny portions of the batch instead of the entire batch, batch normalization transforms the input data into "Normally distributed" data with a 0 mean and 1 standard deviation.

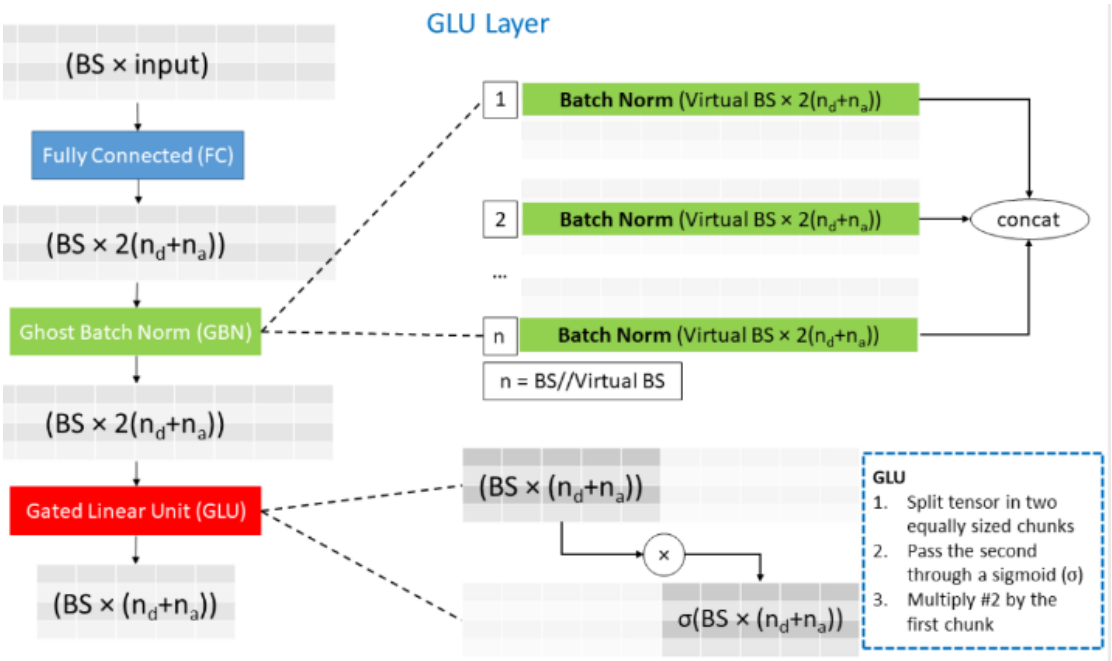

Figure 3.3: Gated Linear Unit Layer(Pochetti, 2020).

3.1.5.2 Feature Transformer

The FT, TabNet's second building block, is primarily made up of the GLU layer. This block consists of 4 GLU layers, the first 2 of which are shared by the entire network. Instead, the last 2 are unique for each FT, providing additional modelling versatility.

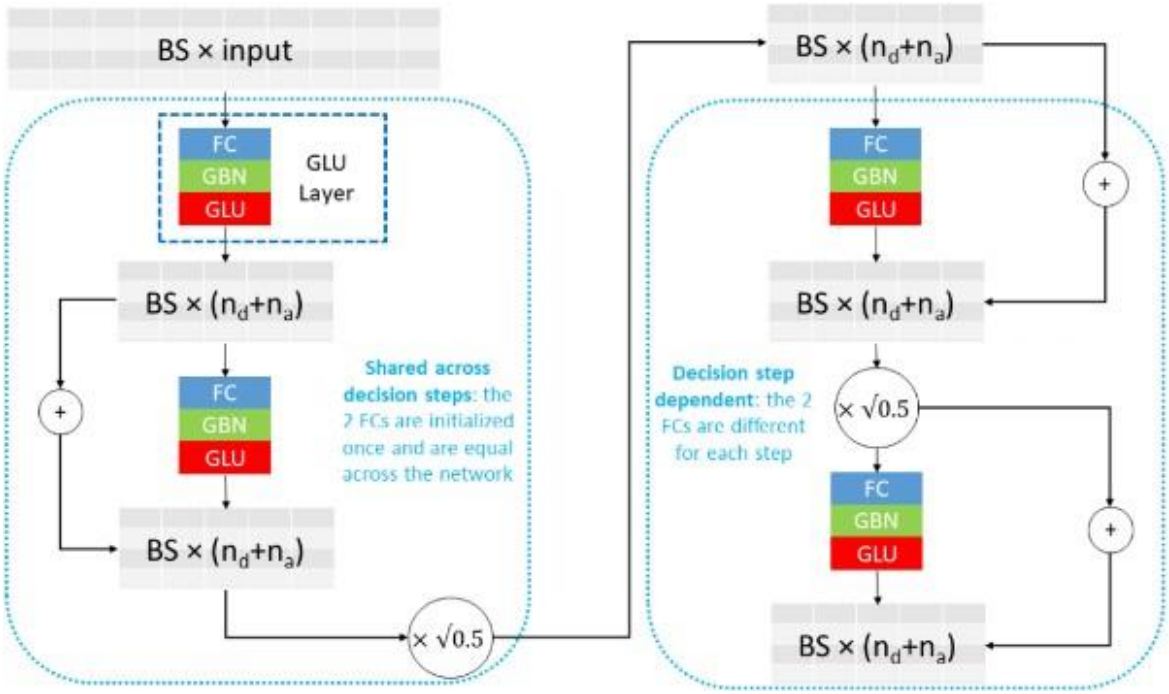

Figure 3.4: Feature Transformer(Pochetti, 2020).

3.1.5.3 Attentive Transformer

The AT is the following brick in the foundation of Tab Net. By imposing sparsity into the feature set and teaching the network to concentrate solely on certain variables, this block aims to provide the network an attention mechanism. An FC layer handles the learning component and grows a sizeable tensor as input. Instead, Sparsemax, a variant of Softmax that can reduce any probability below a certain threshold to zero, provides the sparsity bit. This operation causes some features of the final resultant tensor to be zeroed out. The output of the AT block and the prior tensor are used to spread the attention mechanism throughout the model. Prior,

which is initially set to 1s, is updated and multiplied both inside and outside of AT. It will be seen how that works in the following section.

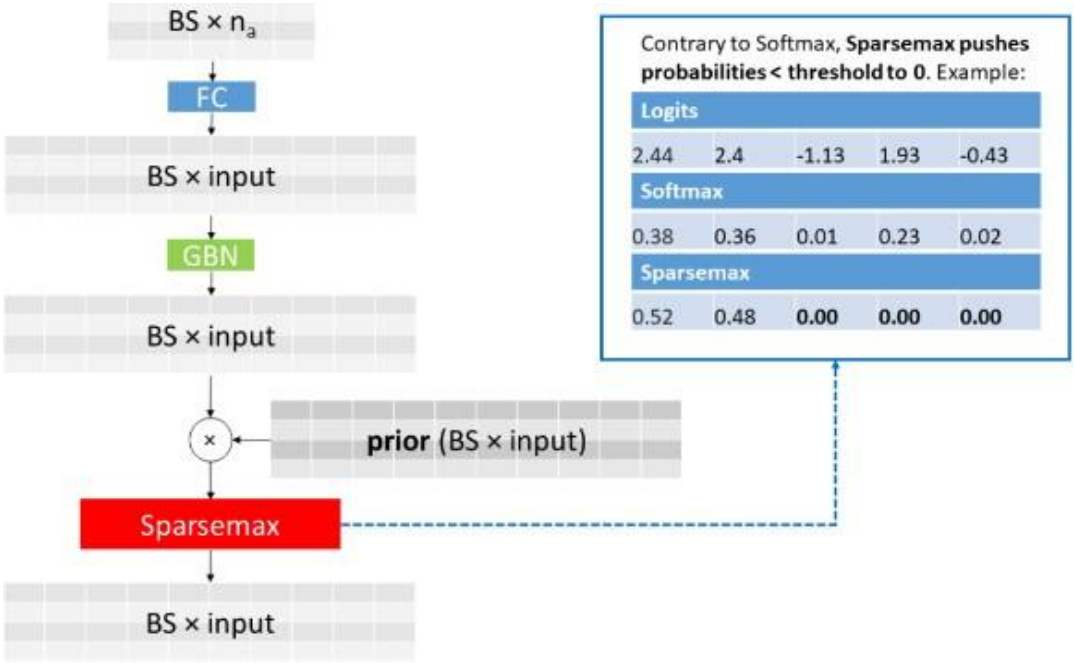

Figure 3.5: Attentive Transformer(Pochetti, 2020).

3.1.5.4 Decision Step

After defining the Feature and Attentive Transformers, we can begin assembling the components of a Decision Step (DS). We'll see that stacking subsequent DSs one after the other is the fundamental concept of TabNet, so let's start by examining one of those. Figure 5 shows the first step of the chain.

(a) There is no need to normalize unprocessed tabular data (x).
That is handled by the first batch normalization layer, which results in x1.x1 passes through FT to obtain x2.

(b) which is divided into two pieces; the attention mask m is produced by directing the att, na-shaped, into AT together with the preceding.

(c) M has two applications. To truly impose the attention mechanism by multiplication with x1, ready to be shipped to the next DS, and to update beforehand while being prepared for shipping.

(d) maskedx leaves travel along a conveyor belt with FT+Split+ReLU and end up with the nd-shaped res.

(e) As you can see from the diagram, the following DS receives 4 elements: x1, masked by the following m, previous, res, and att.

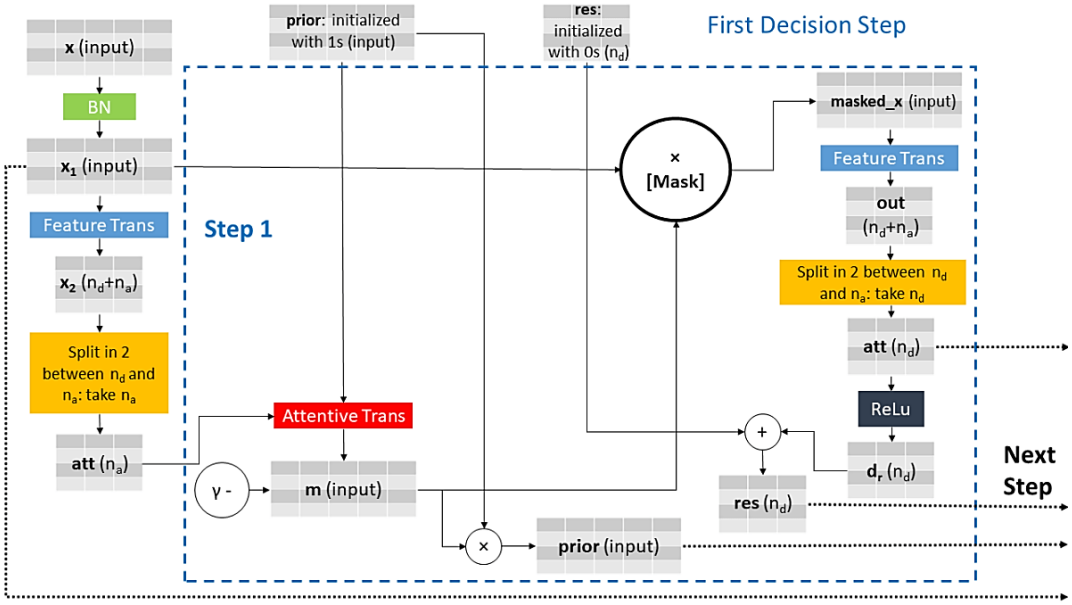

Figure 3.6: Decision Steps(Pochetti, 2020).

3.2 Experimental Methods and Setups

This section This section discusses the methodologies that are implemented to accomplish the research including data collection and preparation, selection of DL framework, and modification of training way of selected framework, hardware and software used, and hyper-tuning.

3.2.1 Dataset Preparation

The tabular dataset was downloaded from the CIC research center. Prior to the data being utilized as an input to the model, we conducted a thorough data analysis and prepared the data in the necessary format and utilized as an input to the model. Then we conducted a thorough data analysis and prepared the data in the necessary format. After that, we trained, validated, and tested the proposed approach by using a pre-pared dataset. In the training process, several hyperparameters and variables were optimized to get an optimal model. Then, the optimized model was tested using the validation dataset.

3.2.1.1 Dataset Description

As previously stated, (Sharafaldin et al., 2018) In order to train Anomaly Network Intrusion Detection System (ANIDS) models, the existing/previous literature relies on datasets with flaws and issues. We used an earlier dataset, CSE-CIC-IDS2018 (Sharafaldin et al., 2018,) which addresses some of the shortcomings of prior datasets.

The datasets include malicious network traffic produced by a number of network attacks, which are briefly explained below, as well as benign (normal) network traffic. In addition to benign (normal) network traffic, the datasets also contain malicious network traffic generated by a variety of network attacks, which are briefly described below.

Brute force attack: The dataset includes traffic statistics for the commonly used network services Secure Shell Protocol (SSH) and File Transfer Protocol (FTP), respectively, which are both subject to brute force attacks. traffic that Heartbleed generates.

DoS attack: A denial-of-service (DoS) attack is a type of network attack in which attackers send an excessive number of phone requests to a service or network in an effort to temporarily deny legitimate users access. A collection of Windows workstations was given Low Orbit Ion Cannons so they could execute DDoS attacks and generate the traffic (Sharafaldin et al., 2018) gathered.

Bot attack: bot attack Attackers use a botnet (a collection of compromised network devices that function as a single virtual network) to conduct a variety of nefarious Internet attacks, including spamming, phishing, providing backdoor access to compromised systems, and data theft and sniffing using key-loggers and sniffers (Sharafaldin et al., 2018b).

Web attack: The Cross-Site Scripting (XSS) (Brute Force-XSS), Structured Query Language (SQL)-Injection (SQL-Injection) Liu and Gupta, 2019 (SQL-Injection), brute force administrator and user passwords (BruteForce-Web), and other prominent attacks against modern web applications are included in the web attack category. To generate Web attack traffic, homegrown automated code developed in

the Selenium framework was used as the attacking tool and the victim service, respectively (Sharafaldin et al., 2018b).

Infiltration attack: carried performed from inside networks once certain systems have been breached by taking use of critical flaws in software applications like PDF Readers, Internet browsers, and so on. The authors used Windows, Ubuntu, and Mac devices on the target network to generate infiltration assault traffic, together with a Kali Linux attack machine (Sharafaldin and others 2018).

Benign traffic: The authors presented and deployed their own method for profiling the abstract behavior of human interactions to produce real-world benign or regular network traffic. The dataset is made up of more than 5 million benign traffic samples that simulate the actions of 25 online users as they utilize a variety of common apps and services, such as FTP, SSH, HTTP, HTTPS, HTTP, email, and others (Sharafaldin et al., 2018b).

3.2.1.2 Features

Each cleaned dataset has 79 characteristics, of which only two (Destination Port and Protocol) are categorical and are encoded using a 1-to-n scheme. The remaining 77 qualities are all numbers. Even though the original dataset has data with the best feature selection, we do not use this data in our research (Sharafaldin et al., 2018b).

3.2.1.3 Data pre-processing

After downloading the data, we examined it to understand its properties and cleaned it up as necessary. The dataset comprises feature-selected CSV files as well as the original traffic in Packet Capture (pcap) files, logs, and preprocessed and tagged Comma-Separated Values (CSV) files. We used labelled CSV files with 80 traffic characteristics that CICFlowMeter was able to collect. The dataset is broken up into seven CSV files and includes both normal (benign) and attack traffic, which is composed of a number of common attack types (as described in the section above). Due to the large number of available samples and in order to simplify trials, we removed samples containing Infinity, not a Number (NaN) or missing values. Changed from timestamps to Unix epoch numerical values the duplicated column headings in some data sets were analyzed and removed. Following the incident, a data cleanup procedure led to the loss of around 20,000 samples.

Table 3.1 summarizes the datasets that we use for our research after the data cleaning process. Samples Count After deducting the number of samples from each category listed in the previous column, the number of samples overall is shown in the remaining column.

Table 3.1: Static Distributions of Attack Type.

Original Dataset	**Attack Type**	**Remaining Sample**	**Dropped Sample**
02-14-2018	Benign	663,808	3824
	FTP-Brute Force	193,354	6
	SSH-Brute-force	187,589	0
02-15-2018	Benign	988,050	8027
	DoS-Golden Eye	41,508	0
	DoS-Slowloris	10,990	0
02-16-2018	Benign	446,772	1
	DoS-SlowHTTPTest	139,890	0
	DoS-Hulk	461,912	0
02-22-2018	Benign	1,042,603	5610
	Brute Force-Web	249	0
	Brute Force-XSS	79	0
	SQL-Injection	34	0
02-23-2018	Benign	1,042,301	5708
	Brute Force-Web	362	0
	Brute Force-XSS	151	0
	SQL-Injection	53	0

	Benign	235,778	2944
03-01-2018	Infiltration	92,403	660
	Benign	758,334	4050
03-02-2018	Bot Attack	286,191	0

Table 3.2: Number of samples for each type of traffic data

Traffic Attack Type	Number of samples
SSH-BruteForce	187,589
Infiltration	92,403
FTP-BruteForce	193,354
DoS-Slowloris	10,990
DoS-GoldenEye	41,508
DoS-SlowHTTPTest	139,890
BruteForce-Web	611
DoS-Hulk	461,912
SQL-Injection	87
Bot Attack	286,191
Benign	5,177,646
Total Attack	1,414,766

3.2.1.4 Data Mapping for Encrypted Internet Traffic

In the current investigation, the labeled data from each dataset will be cleaned and altered for following testing using binary classification. Binary categorization necessitates distinguishing between encrypted harmful communication and non- malicious traffic. According to the executive summary provided in Table 3.3,

the resulting data for the CSE-CIC-IDS2018 is shown in 4-class labelling categories, together with the related training and testing records as any other encrypted channel that can carry attack traffic. According to the executive summary provided in Table 3.3, the resulting data for the CSE-CIC-IDS2018 is shown in 4-class labelling categories, together with the related training and testing records.

Table 3.3: 4-class labeled dataset.

Internet Traffic	Normal	Attack
Encrypted	2066	1698
Non-Encrypted	860804	14841

Figure 3.7 depicts a further sub approximation of several protocols (TLS, SSH, HTTP, Internet Printing Protocol (IPPS), Lightweight Directory Access Protocol (LDAPS), FTP, etc.) forming malicious traffic in the non-encrypted and encrypted categories for each dataset. While the statistics indicate the variance in traffic volume (%) distribution across the aforementioned protocols, TLS and SSH are the most common throughout encrypted attack traffic. For the CSE-CIC-IDS2018, Each SSH and TLS protocol accounts for around 20–40% of all encrypted attack traffic. According to CSE-CIC-IDS2018, SSH and TLS account for around (20–40 percent) of encrypted attack traffic. Other important protocols in CSE-CIC-IDS2018 include HTTP, FTP, IPPS, and LDAPS (as well as a tiny proportion of Simple Network Management Protocol (SNMP), Simple Mail Transfer Protocol (SMTP), Teletype Network Protocol (Telnet), Domain Name Service (DNS), and others) in encrypted and non-encrypted mode. The training and testing data for encrypted and non-encrypted attack traffic were shared between each protocol category in almost equal amounts.

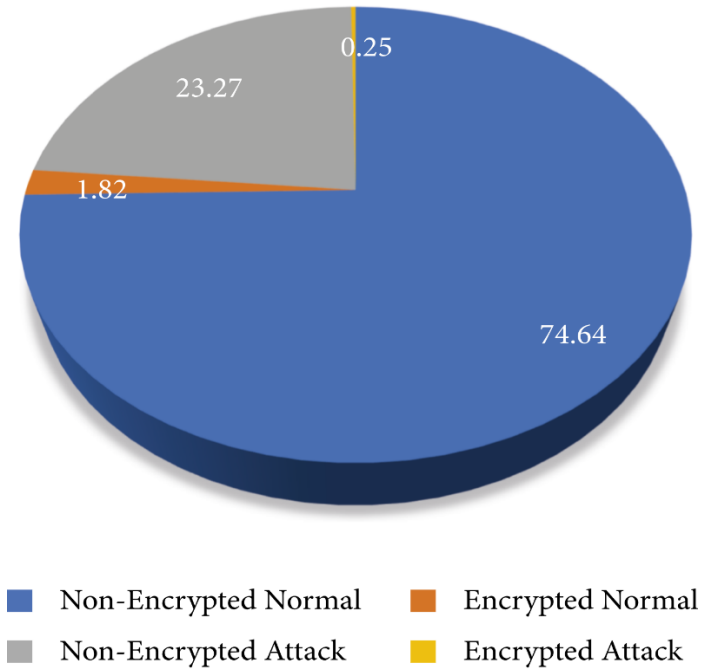

Figure 3.7: Attack traffic composition.

3.2.1.5 Data Partition for DL framework

It is experimented with a variety of modern DL frameworks, which are discussed in Chapter 4. N-fold cross-validation (usually 10-fold) and train-test split (often 70-30 or 80-20) are two methods frequently used to assess ML models. While train-test split is used when there are many samples in each category in the dataset, N-fold cross-validation is utilized when there are few or inconsistent samples in some categories. We applied both methods as necessary.

To evaluate binary-class and multi-class classification, we combined the entire set of original files and produced two additional datasets. For binary-class categorization, we assigned a value of 1 to all attack traffic and a value of 0 to all benign traffic.

3.2.1.6 Data split for TCP/IP layer

The dataset is organized into layers according to the different sorts of assaults at the various TCP/IP levels, as shown in the architectural diagram. Since no attack type in the data set fits into that category, Link Layer is not included in this analysis. There are 379,609 samples in the dataset for the Network layer and 490,251 samples in the dataset for all layers. Three category characteristics are included in the dataset,

and they must first be numerically encoded before being input into the algorithm model.

Table 3.4: Number of attack samples for each TCP/IP Layer.

Layers	Number of samples
Application Layer	5,043,980
Transport Layer	83379
Network Layer	1,465,052

3.2.2 Proposed Modification on Fast.ai Tabular Training

Fast.ai has introduced Tabular-Pandas, a potent tool to assist with processing your data while working with tabular data. It's incredibly convenient and beneficial because everything can be found in one location, all of your tables can be encoded and decoded at once, and only a small amount of memory is used on top of your Pandas data-frame. I wanted to maintain Tabular-Pandas in its current form because it's a fantastic pre-processing tool while also speeding up everything! As an alternative, we'll develop a new Dataset class and turn our Tabular-Pandas object into a NumPy array there. Why is that crucial? In comparison to Python, NumPy is a considerably quicker library that has been hyper-optimized to use as much C code as possible.

The Tabular-Pandas dataset from fast.ai is being converted to a Numpy Dataset as the initial alteration. The same procedure can be quickly completed with Numpy-dataset.ai The sole distinction between Numpy-Dataset and Tabular-Pandas is that Numpy-Dataset calls numpy () on the stored Data-Frame (cats, conts, and ys) to maintain the current state. We'll finally assign our cats to be np.long and our counts to be np.float32 in order to deal with categorical versus continuous variables (we also have our ys as np.int8, but this is because we're doing classification).

The second change is being made. NumPy can be integrated with fast.ai Data Loader. NumpyDataset does not explicitly support the Baseline for a fast speedup test.AI libraries are the benchmark for incredible speed up test coming from for

fast.ai libraries in a batch size, is added as we prepare to develop our Numpy Data-Loader by examining the fast.ai DataLoader in great detail.

3.2.3 Baseline Setups for Speed Test for PyTorch-Fast.ai

Utilizing the time, its module, which calculates the length of time it takes for specific loops in Python, speed tests are conducted. We run four alternative tests for our tests, including Fitting for 10 epochs (GPU only), Iterating through the full training dataset, iterating over the entire validation set, and One batch of the training data.

3.2.4 Experiment Environment Setup

The following hardware and software features were used in this experiment on an

(a) Hardware Tool Used

Advanced Micro Device (AMD) Ryzen 9 3900X Central Processing Unit (CPU) has 12-cores and 24 threads, operating at 3.80 GHz Memory (RAM): 64 GB

GPU: Two 11 GB Nvidia GeForce GTX 2080 Ti GPU

Operating system: kali-Linux

(b) Software Tool Used

To implement this research, python is used as a programming language and web browser such as Firefox, chrome. The Google Co-Lab is where the trials were conducted. The time measurements (training and testing times) were recorded on a device that met the criteria listed below. For certain CPU-based studies, a standard computer with 12 GB of RAM took several hours, but it ran out of memory for larger datasets. As a result, we purchased Google Co-lab Pro. We then conducted all our trials again to contrast the timing and accuracy.

3.2.5 Hyper-parameter

The performance of the network is significantly influenced by the hyperparameters employed in the DNN's architecture. The learning rate, number of hidden layers, number of units/cells in the hidden layer, and number of time-steps are the parameters having the biggest impact on the network's performance despite the fact that there are numerous hyper-parameters involved in the construction of a

DNN.fast.ai Tabular Model, the parameters emb_ szs, n_cont, out_sz, embed_p, and y_range are identical.

- n_d : The prediction layer's int dimension is typically between 4 and 64.
- n_a : The attention layer's int dimension is typically between 4 and 64.
- n_steps: int Number of successive steps in the network (usually between 3 and 10)
- gamma: float Float above 1, scaling factor for attention updates (usually between 1.0 to 2.0)
- momentum: float the momentum in every batch norm will be represented by a floating-point value between 0 and 1.
- n_independent: int Number of independent GLU layer in each GLU block (default 2)
- n_shared: int Number of independent GLU layer in each GLU block (default 2)
- epsilon: Avoid using log (0), and keep it at a very low value.

3.3 Evaluation Metrics

In machine learning, after training is completed, the model is evaluated using different evaluation metrics to test how the model generalizes for the new data Accuracy, precision, recall, and F1 score were the assessment measures employed in this study to assess the performance of the suggested model. Other than these metrics training accuracy, validation accuracy, training loss and validation loss were used to measure model per for in addition to these parameters, model performance during training was assessed using training accuracy, validation accuracy, training loss, and validation loss. during training.

- Accuracy is defined as the percentage of correctly identified samples to all samples. When the dataset is balanced, accuracy is an appropriate metric. However, in actual network contexts, normal samples are much more common than aberrant samples, therefore accuracy might not be the best statistic.

- Precision, which reflects the certainty of attack detection, is defined as the ratio of true positive samples to expected positive samples.
- Recall, often known as the detection rate, is the proportion of authentically positive samples to all positive samples. The model's capacity to detect the assault is indicated by the detection rate.
- F-measure is defined as the harmonic average of the precision and the recall. The false-negative rate (FNR) is defined as the ratio of false-negative samples to total positive samples. In attack detection, the FNR is also called the missed alarm rate.

CHAPTER FOUR

4. RESULTS AND DISCUSSIONS

4.1 Analysis of the Proposed Approach

In this chapter, we will go through the tools used in the design process, as well as the integration points for submodules and the deployment of the solution in a suitable environment. This chapter attempts to describe the experimental results of the recommended approach. Furthermore, a comprehensive set of experiments was conducted to determine the individual performance of the suggested/proposed approach and model. The experimental results are briefly discussed in this section, and the proposed model performance is compared to existing approaches.

4.2 Evaluation of the Deep Learning (DL) Framework

It's used the DL frameworks on and recorded performance metrics like accuracy, loss, time, and so on. It's experimented with each framework's many parameters in order to develop the best deep learning classifier to serve as the baseline classifier.

4.2.1 Performance Results of Theano-Keras

Figure 4.1 shows the accuracy results for experiments with the Binary class over a period of ten epochs.

Table 4.1: Performance results for Theano-Keras.

DL framework	Class	Accuracy (%)	Loss (%)	Time (min)
Theano-Keras	Binary-class	92.01	18.52	105.90
	Multi-Class	95.105	17.43	632.02

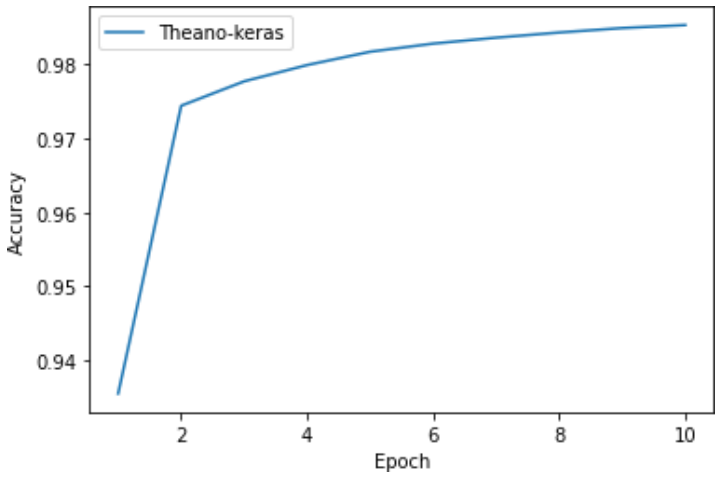

Figure 4.1: Accuracy results on over 10 epochs.

4.2.2 Performance Results of Tensor-Flow-Keras

Figure 4.2 shows the accuracy results for examines using the Binary class over a period of ten epochs.

Table 4.2: Evaluation Metrics for Tensor-Flow-Keras.

DL framework	Class	Accuracy (%)	Loss (%)	Time (min)
Tensor-Flow-Keras	Binary-class	92.04	18.54	105.97
	Multi-Class	95.135	17.58	632.35

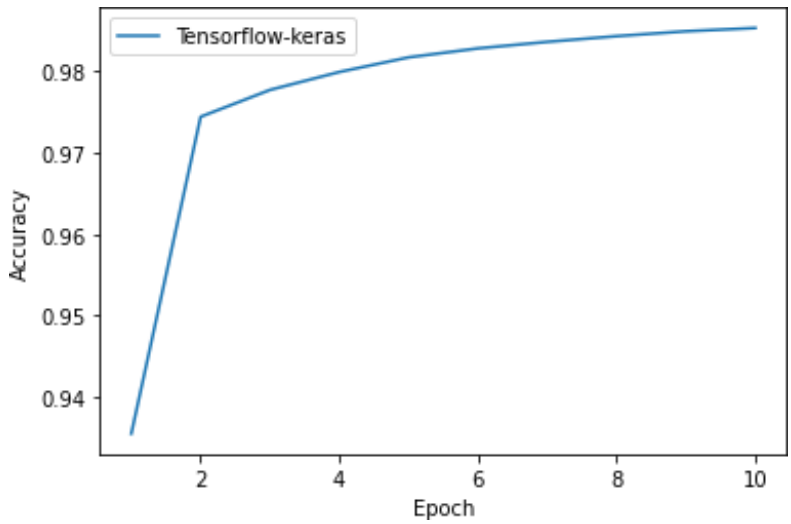

Figure 4.2: Accuracy results over 10 epochs.

4.2.3 Performance Results of PyTorch-Fast.ai

The accuracy results for experiments using the Binary-class are shown in Figure 4.3.

Table 4.3: Evaluation Metrics for PyTorch-Fast.ai

DL framework	Class	Accuracy (%)	Loss	Std-Dev (%)	Time (min)
PyTorch-Fast.ai	**Binary-class**	**96.68**	**0.37692**	**0.05**	**675.98**
	Multi-Class	**98.31**	**7.06169**	**0.16**	**683.12**

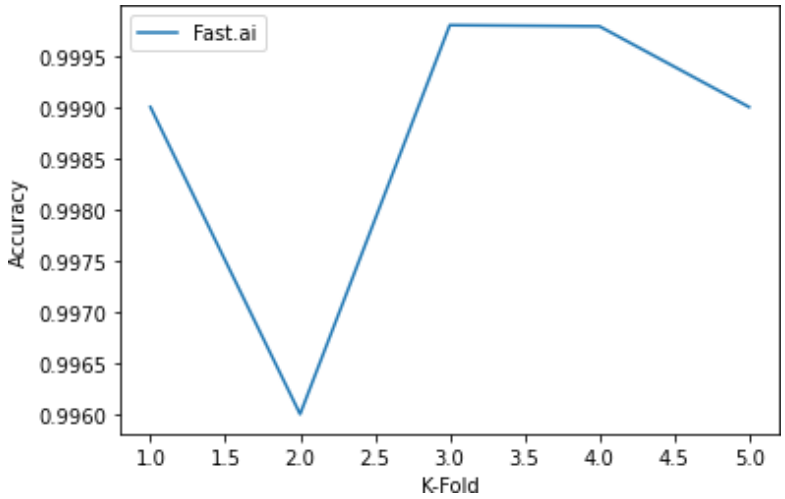

Figure 4.3: Accuracy results over 5- folds

4.2.4 Comparison Results of DL Framework for IDS

Table 4.4 summarizes the performance results for each dataset using CPU. For individual files, Theano results are fairly comparable to those of Keras-TensorFlow, thus we believed that Keras-Theano would not likely produce better results as fast.ai. Fast.ai is essentially an additional layer on top of PyTorch that extends the training utilities with a concept called "callbacks" (which Keras also has but PyTorch lacks). This gives your neural network a lot of new functionality, including methods for visualising your data, more ways to load and split data, inferring the number of classes from the dataset you provide, and callbacks, which PyTorch lacks.

Table 4.4: Comparison results of DL framework.

DL framework	Class	Accuracy (%)	Loss (%)	Time (min)
Theano-Keras	Binary-class	92.01	18.52	105.90
	Multi-Class	95.105	17.43	632.02
Tensor-Flow-Keras	Binary-class	92.04	18.54	105.97
	Multi-Class	95.135	17.58	632.35
PyTorch-Fast.ai	Binary-class	96.68	0.37692	675.98
	Multi-Class	98.31	7.06169	683.12

4.3 Modified PyTorch-Fast.ai Evaluation

It's tested a modified fast.ai to see how fast it was and how much memory it consumed. Run a speed test on the entire dataset to evaluate performance improvements. On network layer datasets, space complexity (memory consumption) was tested to confirm that optimized consumes less memory than baseline fast. Given the limited resources of small devices and IoT, simple and lightweight systems are required.

4.3.1 Performance Results of Baseline Fast.ai dataloader

Baseline Fast.ai uses Pandas Object in data-loader mid-level API.

Table 4.5: Performance results of Baseline fast.ai.

Baseline Fast.ai	**Devices**	**First Bach**		**Per batch**		**Per**	
		Train	Valid	Train	Valid	Epoch	Ten Epoch
Time(*S*)	CPU	878.8*ms*	8.65*ms*	20.3 *s*	4.45 *s*		
	GPU	877*ms*	10*ms*	20.3 *s*	4.46 *s*	96 *s*	960 *s*
Memory (*MB*)	CPU	1497.14 *MB*	1496.71 *MB*	1502.07 *MB*	1502.10*MB*		
	GPU	1484.67 *MB*	1503.86 *MB*	1505.44 *MB*	1505.47*MB*	4932.7 *MB*	49327.00 *MB*

4.3.2 Performance Results of Modified Fast.ai/Customized numpy dataloader

Table 4.6: performance results of Modified fast.ai.

Modified Fast.ai	Devices	First Bach		Per batch		Per	
		Train	Valid	Train	Valid	Batch	Ten Epoch
Time	CPU	24.8 *ms*	3.56 *ms*	1.3 *s*	326 *ms*		
	GPU	25.4 *ms*	3.79 *s*	2.3 *s*	567 *ms*	69.6 *s*	696*s*
Memory (*MB*)	CPU	1491.96 *MB*	1491.96 *MB*	1492.48 *MB*	1492.50 *MB*		
	GPU	3152.18 *MB*	3154.20 *MB*	3154.72 *MB*	3154.72 *MB*	4932.7 *MB*	49327.00 *MB*

4.3.3 Comparison of the Results for Fast.ai Tabular Training

To summarize, one can first reduced the time it took to retrieve a single batch of data by converting everything from Pandas to NumPy. Following that, we created a custom DataLoader that could handle these NumPy arrays and cause the speedup it has been observed. This research paper helps you understand how the interior DataLoader can be integrated with NumPy and speeds up your tabular training. As shown in table 4.7, the proposed system outperforms the baseline in terms of speed on customized data loaders.

Table 4.7: Comparison performance based on time/speed.

Fast.ai	Devices	First Bach		Per batch		Per	
		Train (*sec*)	Valid (*Sec*)	Train (*sec*)	Valid (*Sec*)	Epoch(*sec*)	Ten epoch(*sec*)
Baseline Fast.ai	CPU	878.8 *ms*	8.65 *ms*	20.3 *s*	4.45 *s*		
	GPU	877 *ms*	10 *ms*	20.3 *s*	4.46 *s*	96 *s*	960 *s*
Modified Fast.ai	CPU	24.8 *ms*	3.56 *ms*	1.3 *s*	326 *ms*		
	GPU	25.4 *ms*	3.79 *s*	2.3 *s*	567 *ms*	69.6 *s*	696*s*

As shown in Table 4.8, the proposed system outperforms the baseline for CPU base sensor node in terms of speed on customized dataloaders. But not for GPUs because Numpy is designed for the most recent CPU architecture.

Table 4.8: Compare performance based on memory consumption.

Fast.ai	Devices	First Bach		Per batch		Per	
		Train (*sec*)	Valid (*Sec*)	Train (*sec*)	Valid (*Sec*)	Epoch(*sec*)	Ten epoch(*sec*)
Baseline fast.ai	CPU	1497.14 *MB*	1496.71 *MB*	1502.07 *MB*	1502.10 *MB*		
	GPU	1484.67 *MB*	1503.86 *MB*	1505.44 *MB*	1505.47 *MB*	4932.7 *MB*	49327.00 *MB*
Modified fast.ai	CPU	1491.96 *MB*	1491.96 *MB*	1492.48 *MB*	1492.50 *MB*		
	GPU	3152.18 *MB*	3154.20 *MB*	3154.72 *MB*	3154.72 *MB*	4932.7 *MB*	49327.00 *MB*

4.4 IDS Classifier Evaluation

In this section, it offers detailed results and evaluation measures for each IDS classifier obtained using TabNet DL architecture.

4.4.1 Hyper parameter settings

It took 15 sets of experiments for each layer to achieve the best results. It was found that this was a problem of binary classification, with the classifier categorising each sample as either "normal" or "attack." By considering every pertinent metric for the specified research challenge, we selected the top models.

4.4.1.1 Choosing a Good Learning Rate

The learning rate finder plots the relationship between lr and loss. Train the learner over several iterations. Begin with a very low startlr and gradually increase it with each mini-batch until it reaches a very high endlr. The loss will be recorded at each iteration by the recorder. Plot those losses against the learning rate to determine the best value before it diverges.

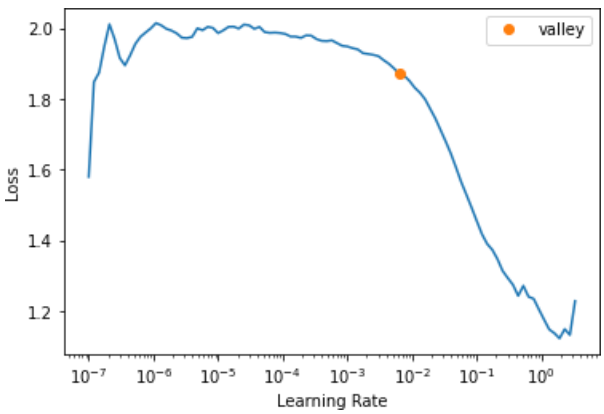

Figure 4.4: Optimal learning rate.

4.4.1.2 Set Parameter Space and Value

TabNet's hyper-parameter settings are shown in Table 4.9. N_d, N_a, and N_steps are important parameters that determine the model's capacity. N_steps ranging from 3 to 10 is a reasonable parameter for most data sets, and N_d = N_a is a reasonable choice 2020, Ark and Pfister. Reducing N_d, N_a, and N_steps is an effective way to reduce over-fitting while maintaining accuracy.

Table 4.9: parameter space and value of TabNet.

Hyper-parameter	Values Ranges
N_d	(0,8)
N_a	(0,8)
N_steps	(0,4)
Optimizer	Adam
Gamma	1.5
lr	3e-2

4.4.2 Performance Results of each IDS classifier

4.4.2.1 Hyper parameter

Models are built in 8 hours and 30 minutes by the package bayes_opt. The best accuracy is 0.9812.

Table 4.10: optimal hyper-parameter value of each IDS.

Hyper-parameter	All	Transport	Application	Network
N_d	4.887	6.878	4.717	7.157
N_a	1.198	6.998	3.167	3.867
N_steps	1.535	0.2767	1.417	2.871

4.4.2.2 Results of Feature Selection

The implemented outcome of Feature Selection As illustrated in Figures 4.5, 4.6,4.7, and 4.8, the likelihood of being attacked is determined by a variety of factors; therefore, how to better select features will become an important factor influencing model performance. To select features, we use TabNet's Instance-wise feature selection method. Figures 4.5, 4.6, and 4.7 depict feature importance masks (which show which features are selected at the i ^{the} step.

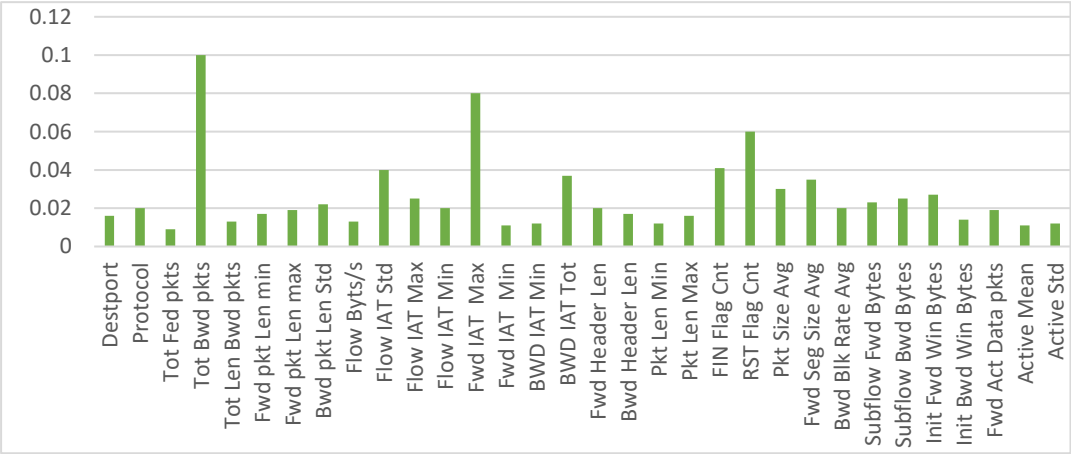

Figure 4.5: Important features for IDS on every layer.

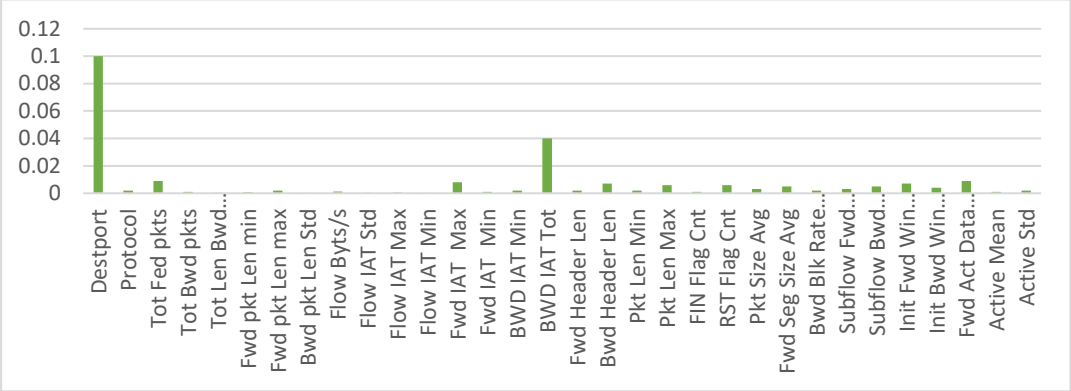

Figure 4.6: The significance of a feature for application layer IDS.

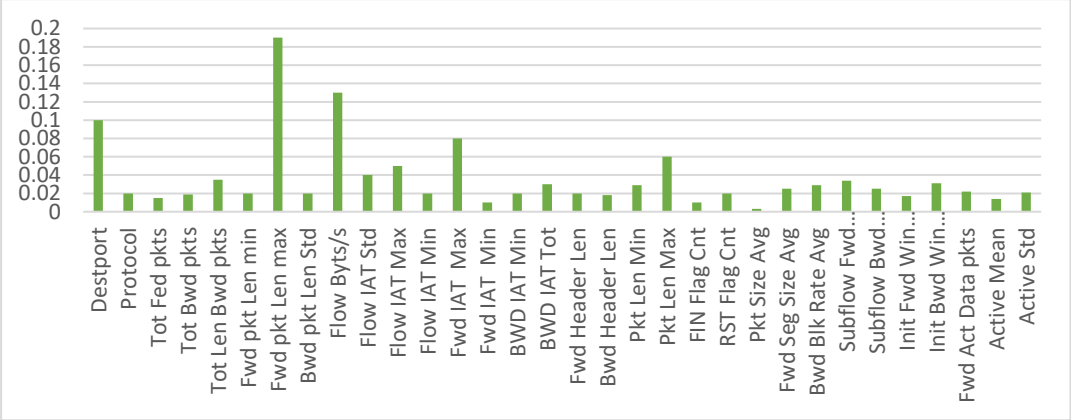

Figure 4.7: Importance of the feature for Transport layer IDS.

4.4.2.3 Results of selective attention

Figures 4.12,4.9,4.9, and 4.11 show between the first and third choice steps, which features are chosen. The feature was given greater consideration during this decision-making process, as indicated by the brighter color. Figures 4.12 and 4.9 illustrate how the instance-wise idea will result in a new weight being assigned to each feature for each decision step. The aggregate feature importance masks in the CSE-CIC_IDS2018 dataset, demonstrating global instance-wise feature selection. Brighter colors indicate a greater value. The masks for each input instance are represented by a row.

The figure depicts the output masks of 20 different input instances. The CSE-CIC_IDS2018 has 80 features, and the output is designed so that it only depends on

the bright value of the features. Each column in the figure represents a different feature, such as the dst protocol feature in the first column.

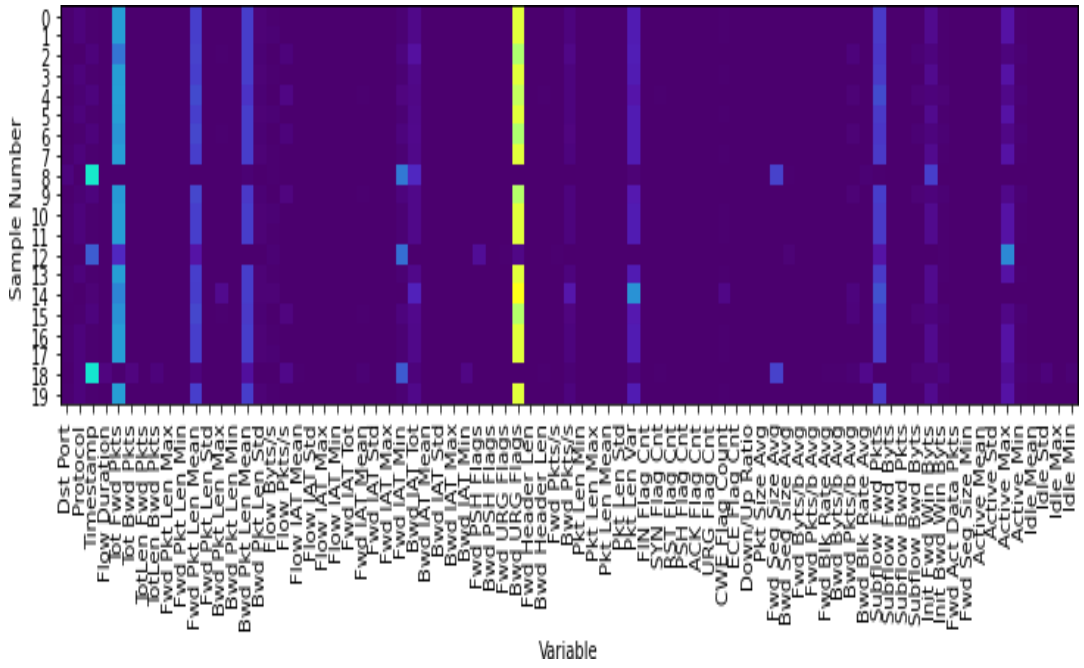

Figure 4.8: Masks of features of importance for the application layer.

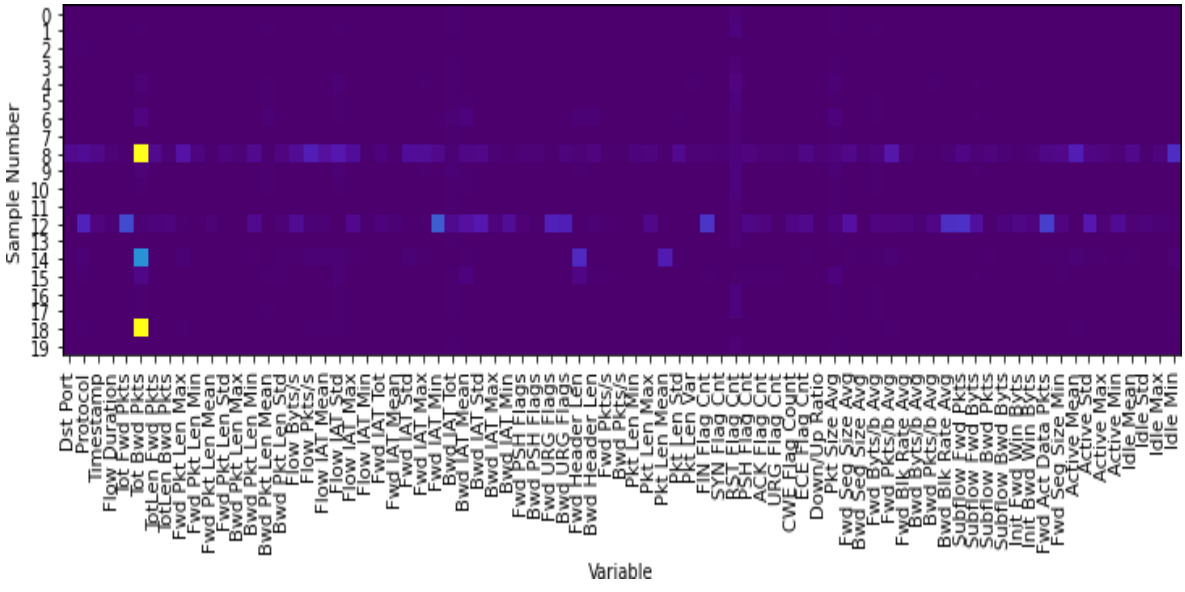

Figure 4.9: Network layer features importance masks.

4.4.3 IDS classifiers Comparison

When the ideal outcomes of all IDS classifiers are considered. It is found that the All-Layers IDS classifier performs worse than the individual layer IDS classifiers in terms of training accuracy and training time. Table 4.11 compares the outcomes.

Table 4.11: Performance comparison among proposed IDS classifiers.

IDS	Accuracy (%)	Precision (%)	False Alarm Rate (FAR)(%)	Recall (%)	F1-score (%)
All layer IDS	98.12	98.86	0.78	98.34	98.60
Application Layer IDS	98.6	98.93	0.07	98.6	98.76
Transport Layer IDS	98.2	98.83	0.24	980.6	98.44
Network layer IDS	98.3	98.16	2.0	98.33	98.24

4.4.4 Analysis of the IDS classifiers' Performance in Relation to Previous Work

Table 4.12 illustrates the findings of a subsequent study and comparison with previous research on intrusion detection using ML and DL techniques. It can be demonstrated that our study outperformed all previous studies.

Table 4.12: comparisons between the proposed IDS classifiers and the current IDS classifiers.

Algorithm Used	Accuracy (%)	Precision (%)	FAR(%)	Recall (%)	F1-score (%)
BLSTMSu et al., 2020	84.25	✗	✗	✗	✗
AEZhang et al., 2019	79.74	82.22	✗	✗	✗
RF,NBRT,J48 Mohan et al., 2020	90	✗	✗	✗	✗
ANN,DT,KNNShah et al., 2020	95	✗	✗	✗	✗
SMADDutt et al., 2020	97.1	✗	✗	2.7	✗
CNNL. Chen et al., 2020	96.81	✗	✗	✗	✗
LSTM,KNNSwarna Sugi and Ratna, 2020	92.3	✗	✗	✗	✗
Proposed TabNet Arık and Pfister, (All layer IDS)	98.12	98.86	0.78	98.34	98.60

CHAPTER FIVE
5. CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

This is unique in that Tab Net DL is being used for VANET security for the first time. We thoroughly examined the VANET architecture, as well as privacy and security problems. The focus of this thesis has been reduced because of our research to network data security. We implemented a simple IDS design in the VANET network. We suggested placing IDS classifiers at each layer based on TCP/IP layer design and attack classifications. As a result, the size of each classifier's data set was decreased while the accuracy, re-call, training time, and FAR were all improved. At each IDS classifier, DL methods were used to categorize the data. This method produced exceptional results, outperforming previous literature work. Furthermore, unlike previous studies, we conducted the experiment using the CIC-CSE-IDS2018 dataset, as shown in section 3.2.1.1. It is essential for dynamic VANET networks that the training times for Transport Layer IDS, Application Layer IDS, and Network Layer IDS are roughly half that of All Layer IDS. According to section 4.4.3, every layer IDS performs better than all authors known IDS classifiers in terms of accuracy and FAR, with 98.12% and 0.78%, respectively.

DL's application to VANET to build security solutions is still in its early stages, but we see great potential. Because the VANET deals with users' personal data as well as industrial information, it is critical to implement strong security solutions. Because VANET generates a large amount of heterogeneous data, this is feasible using the ML and DL concepts. On the dataset, we used TabNet DL architectures. This study concentrated on handling VANET devices (on-board units) with constrained processing power and modest data sizes. Applying this research to vast volumes of real-time VANET data could expand it. Not an evolution, but a revolution is what the VANET is. As VANETs advance, security problems increase. If VANETs are secure, which AI can deliver, they are a benefit to society.

5.2 Future Works

In VANET vehicle communication i.e., vehicle to vehicle communication and in vehicle to infrastructure communication they use Wi-Fi network for transmitting data from source to destination, and in Wi-Fi network used some protocols that are HTTP, HTTPS, FTP, SSH, and email protocols. Additionally, the CICIDS2018 dataset was created using all these protocols, allowing us to use it now. Real-world datasets created by real-world driverless automated cars will be used in future work for intrusion detection systems in VANET.

REFERENCES

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018a). Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSP.

Gao, Y., Wu, H., Song, B., Jin, Y., Luo, X., & Zeng, X. (2019). A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network. IEEE Access, 7, 154560–154571.

Zeng, Y., Qiu, M., Zhu, D., Xue, Z., Xiong, J., & Liu, M. (2019). Deepvcm: A deep learning based in- trusion detection method in vanet. 2019 IEEE 5th intl conference on big data security on cloud (BigDataSecurity), IEEE intl conference on high performance and smart computing,(HPSC) and IEEE intl conference on intelligent data and security (IDS), 288–293.

Alsarhan, A., Al-Ghuwairi, A.-R., Almalkawi, I. T., Alauthman, M., & Al-Dubai, A. (2021). Machine learning-driven optimization for intrusion detection in smart vehicular networks. Wireless Per- sonal Communications, 117, 3129–3152.

Zeng, Y., Qiu, M., Ming, Z., & Liu, M. (2018). Senior2local: A machine learning based intrusion de- tection method for vanets. Smart Computing and Communication: Third International Conference, SmartCom 2018, Tokyo, Japan, December 10–12, 2018, Proceedings 3, 417–426.

Giesen, J., Laue, S., & Mitterreiter, M. (2020). Optimization frameworks for machine learning: Examples and case study. it - Information Technology, 62(3-4), 169–180. https://doi.org/doi:10.1515/itit- 2019-0031

Bergstra, J., Bastien, F., Breuleux, O., Lamblin, P., Pascanu, R., Delalleau, O., Desjardins, G., Warde- Farley, D., Goodfellow, I. J., Bergeron, A., & Bengio, Y. (2012). Theano: Deep learning on gpus with python.

Stevens, E., Antiga, L., & Viehmann, T. (2020). Deep learning with pytorch. Manning Publications. Howard, J., & Gugger, S. (2020). Deep learning for coders with fastai and pytorch. O'Reilly Media.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., & Saurous, R. A. (2017). Tensorflow distributions. arXiv preprint arXiv:1711.10604.

Learning, D. (2020). Deep learning. High-Dimensional Fuzzy Clustering.

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for internet of things (iot) security. IEEE Communications Surveys & Tutorials, 22(3), 1646–1685.

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of- the-art in artificial neural network applications: A survey. Heliyon, 4(11), e00938.

Bau, D., Zhu, J.-Y., Strobelt, H., Lapedriza, A., Zhou, B., & Torralba, A. (2020). Understanding the role of individual units in a deep neural network. Proceedings of the National Academy of Sciences, 117(48), 30071–30078.

Elbrächter, D., Perekrestenko, D., Grohs, P., & Bölcskei, H. (2019). Deep neural network approximation theory. arXiv preprint arXiv:1901.02220.

Habibi, M., Starlinger, J., & Leser, U. (2020). Deeptable: A permutation invariant neural network for table orientation classification. Data Mining and Knowledge Discovery, 34(6), 1963–1983.

Dua, S., & Du, X. (2016). Data mining and machine learning in cybersecurity. CRC press.

Li, W., Tug, S., Meng, W., & Wang, Y. (2019). Designing collaborative blockchained signature-based intrusion detection in iot environments. Future Generation Computer Systems, 96, 481–489.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsu- pervised anomaly detection. In Applications of data mining in computer security (pp. 77–101). Springer.

Putchala, M. K. (2017). Deep learning approach for intrusion detection system (ids) in the internet of things (iot) network using gated recurrent neural networks (gru).

Jose, A. C., & Malekian, R. (2017). Improving smart home security: Integrating logical sensing into smart home. IEEE Sensors Journal, 17(13), 4269–4286.

Laufs, J., Borrion, H., & Bradford, B. (2020). Security and the smart city: A systematic review. Sustainable cities and society, 55, 102023.

Luo, X., Liu, J., Zhang, D., & Chang, X. (2016). A large-scale web qos prediction scheme for the industrial internet of things based on a kernel machine learning algorithm. Computer Networks, 101, 81–89.

Esmalifalak, M., Liu, L., Nguyen, N., Zheng, R., & Han, Z. (2014). Detecting stealthy false data injection using machine learning in smart grid. IEEE Systems Journal, 11(3), 1644–1652.

Abie, H., & Balasingham, I. (2012). Risk-based adaptive security for smart iot in ehealth. Proceedings of the 7th International Conference on Body Area Networks, 269–275.

Srivastava, G., Parizi, R. M., & Dehghantanha, A. (2020). The future of blockchain technology in health- care internet of things security.

Li, Y., & Guo, L. (2007). An active learning based tcm-knn algorithm for supervised network intrusion detection. Computers & security, 26(7-8), 459–467.

Pu, G., Wang, L., Shen, J., & Dong, F. (2020). A hybrid unsupervised clustering-based anomaly detection method. Tsinghua Science and Technology, 26(2), 146–153.

Ashfaq, R. A. R., Wang, X.-Z., Huang, J. Z., Abbas, H., & He, Y.-L. (2017). Fuzziness based semi- supervised learning approach for intrusion detection system. Information Sciences, 378, 484– 497.

Bughin, J., Seong, J., Manyika, J., Chui, M., & Joshi, R. (2018). Notes from the ai frontier: Modeling the impact of ai on the world economy. McKinsey Global Institute.

Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888.

Tao, X., Kong, D., Wei, Y., & Wang, Y. (2016). A big network traffic data fusion approach based on fisher and deep auto-encoder. Information, 7(2), 20.

Kim, J., Kim, H., et al. (2017). An effective intrusion detection classifier using long short-term mem- ory with gradient descent optimization. 2017 International Conference on Platform Technology and Service (PlatCon), 1–6.

Kim, J., Kim, J., Kim, H., et al. (2015). An approach to build an efficient intrusion detection classifier.

Journal of Platform Technology, 3(4), 43–52.

Staudemeyer, R. C., & Omlin, C. W. (2013). Evaluating performance of long short-term memory recur- rent neural networks on intrusion detection data. Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, 218–224.

Devaraju, S., & Ramakrishnan, S. (2014). Performance comparison for intrusion detection system using neural network with kdd dataset. ICTACT Journal on Soft Computing, 4(3).

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

Shavitt, I., & Segal, E. (2018). Regularization learning networks: Deep learning for tabular datasets.

arXiv preprint arXiv:1805.06440.

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Modeling tabular data using conditional gan. arXiv preprint arXiv:1907.00503.

Kim, S.-Y., Yun, S.-W., Lee, E.-Y., Bae, S.-H., & Lee, I.-G. (2020). Fast packet inspection for end-to-end encryption. Electronics, 9(11), 1937.

Ning, J., Huang, X., Poh, G. S., Xu, S., Loh, J.-C., Weng, J., & Deng, R. H. (2020). Pine: Enabling privacy- preserving deep packet inspection on tls with rule-hiding and fast connection establishment. European Symposium on Research in Computer Security, 3–22.

Bakhshi, T., & Ghita, B. (2021). Anomaly detection in encrypted internet traffic using hybrid deep learn- ing. Security and Communication Networks, 2021.

Dixit, P., & Silakari, S. (2021). Deep learning algorithms for cybersecurity applications: A technological and status review. Computer Science Review, 39, 100317.

Su, T., Sun, H., Zhu, J., Wang, S., & Li, Y. (2020). Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset. IEEE Access, 8, 29575–29585.

Zhang, C., Ruan, F., Yin, L., Chen, X., Zhai, L., & Liu, F. (2019). A deep learning approach for network intrusion detection based on nsl-kdd dataset. 2019 IEEE 13th International Conference on Anti- counterfeiting, Security, and Identification (ASID), 41–45.

Solanki, S., Gupta, C., & Rai, K. (2020). A survey on machine learning based intrusion detection system on nsl-kdd dataset. Int. J. Comput. Appl, 176, 36–39.

Mohan, L., Jain, S., Suyal, P., & Kumar, A. (2020). Data mining classification techniques for intrusion detection system. 2020 12th International Conference on Computational Intelligence and Communi- cation Networks (CICN), 351–355.

Shah, A., Clachar, S., Minimair, M., & Cook, D. (2020). Building multiclass classification baselines for anomaly-based network intrusion detection systems. 2020 IEEE 7th International Conference on Data Science and AdvancedAnalytics(DSAA),759–760. https://doi.org/10.1109/DSAA49011. 2020.00102

Kumar, M., & Singh, A. K. (2020). Distributed intrusion detection system using blockchain and cloud computing infrastructure. 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), 248–252. https://doi.org/10.1109/ICOEI48184.2020.9142954

Chen, C., Song, L., Bo, C., & Shuo, W. (2021). A support vector machine with particle swarm optimiza- tion grey wolf optimizer for network intrusion detection. 2021 International Conference on Big Data Analysis and ComputerScience(BDACS),199–204. https://doi.org/10.1109/BDACS53596. 2021.00051

Dutt, I., Borah, S., & Maitra, I. K. (2020). Immune system based intrusion detection system (is-ids): A proposed model. IEEE Access, 8, 34929–34941. https://doi.org/10.1109/ACCESS.2020.2973608

Chen, L., Kuang, X., Xu, A., Suo, S., & Yang, Y. (2020). A novel network intrusion detection system based on cnn. 2020 Eighth International Conference on AdvancedCloudandBigData(CBD),243–247. https://doi.org/10.1109/CBD51900.2020.00051

Swarna Sugi, S. S., & Ratna, S. R. (2020). Investigation of machine learning techniques in intrusion de- tection system for iot network. 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), 1164–1167. https://doi.org/10.1109/ICISS49785.2020.9315900

Mane, S., & Rao, D. (2021). Explaining network intrusion detection system using explainable ai frame- work. arXiv preprint arXiv:2103.07110.

Marino, D. L., Wickramasinghe, C. S., & Manic, M. (2018). An adversarial approach for explainable ai in intrusion detection systems. IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, 3237–3243.

Nguyen, X.-H., Nguyen, X.-D., Huynh, H.-H., & Le, K.-H. (2022). Realguard: A lightweight network intrusion detection system for iot gateways. Sensors, 22(2), 432.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778.

Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2016). Very deep convolutional networks for text classification. arXiv preprint arXiv:1606.01781.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. International conference on machine learning, 173–182.

Wang, W., Guan, X., Khan, M. T., Xiong, Y., & Wei, D.-Q. (2020). Lmi-dforest: A deep forest model to- wards the prediction of lncrna-mirna interactions. Computational Biology and Chemistry, 107406.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30, 3146–3154.

Polzlbauer, G., Lidy, T., & Rauber, A. (2008). Decision manifolds—a supervised learning algorithm based on self-organization. IEEE transactions on neural networks, 19(9), 1518–1530.

Grbovic, M., & Cheng, H. (2018). Real-time personalization using embeddings for search ranking at airbnb. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 311–320.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018b). Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp, 1, 108–116.

Liu, Z., & Gupta, B. (2019). Study of secured full-stack web development. CATA, 317–324. Arık, S. O., & Pfister, T. (2020). Tabnet: Attentive interpretable tabular learning. arXiv.

Hamdi, M. M., Audah, L., Rashid, S. A., Mohammed, A. H., Alani, S., & Mustafa, A. S. (2020). A review of applications, characteristics and challenges in vehicular ad hoc networks (VANETs). 2020 international congress on human-computer interaction, optimization and robotic applications (HORA),

Pochetti, F. (2020). Tabular Deep Leaning: TabNet deep-dive. https://francescopochetti.com/tabular-deep-leaning-tabnet-deep-dive/