A HYBRID DEEP LEARNING MODEL FOR INTRUSION DETECTION IN CLOUD-BASED IMPLANTABLE MEDICAL DEVICES

JAMES KIRIMI

A Thesis Submitted to the Graduate School in Partial Fulfilment of the Requirements for the Award of the Degree of Master of Science in Computer Science of Chuka University

CHUKA UNIVERSITY

NOVEMBER 2025

DECLARATION AND RECOMMENDATION

Declaration

This thesis is my original work and has not been submitted for an award of diploma or conferment of degree in any other University.

Signature: ..... Date: 29/10/2025

James Kirimi
SM22/63235/23

Recommendation

This thesis has been examined, passed and submitted with our approval as University supervisors.

Signature: ..... Date: 29/10/2025

Dr. Edna Chebet Too, PhD
Chuka University

Signature: ..... Date: 29/10/2025

Prof. David Gitonga Mwathi, PhD
Chuka University

DEDICATION

This work is dedicated firstly to God, whose unfailing wisdom and strength have guided me through every challenge and triumph of this journey. I also dedicate this thesis to my supervisors, Dr. Edna Chebet Too and Prof. David Gitonga Mwathi. Their expert guidance, constant encouragement, and relentless pursuit of academic excellence have shaped each chapter of this research and pushed me to reach new heights. Further, I dedicate this work to the Computer Science department at Chuka University. The motivation, guidance, and support I received from the department were instrumental in shaping my research interests, refining my competencies, and achieving the skills needed to develop the deep autoencoder.

Finally, I owe this work to my family and friends, whose unwavering support, patience, and understanding sustained me through moments of difficulty and celebration alike. Your belief in me had been a source of enduring motivation. May this modest contribution help safeguard the lives of those who depend on implantable medical devices.

ACKNOWLEDGEMENTS

First, I give thanks to God Almighty for sustaining me through each stage of this research and for opening doors of opportunity that I could not have imagined. I am deeply indebted to my supervisors, Dr. Edna Chebet Too and Prof. David Gitonga Mwathi, for their expert advice, constructive criticism, and unwavering support. Their insights into cybersecurity and machine learning methodologies have been invaluable. I extend my gratitude to the faculty and staff of the Department of Computer Science, especially my lecturers, whose teaching laid the foundation for this work. Their dedication, sacrifices, and immense help and guidance were important in shaping my skills, refining my competencies in the field, and strengthening my programming skills. Heartfelt thanks go to my family, my parents, siblings, and extended relatives, for their love, encouragement, and the sacrifices they made so I could pursue this degree. I also appreciate the moral support of my friends, whose faith in me never wavered. Finally, I acknowledge the institutions and laboratories that provided access to computing resources and clinical data necessary for this study.

Disclaimer:

While many individuals and organizations have generously contributed time, expertise, and facilities, I alone remain responsible for any errors or omissions contained herein.

ABSTRACT

The rapidly evolving technologies in the healthcare sector, such as implantable medical devices (IMDs), require advanced security solutions that leverage the intelligence capabilities of these technologies while ensuring optimal safety and reliability. The IMD technology redefines healthcare service delivery by offering timely interventions, minimally invasive treatment options, and continuous patient condition monitoring to improve quality of life. Despite these achievements, IMDs face unauthorised access, data manipulation, and denial-of-service attacks, which conventional security solutions are limited in handling due to resource constraints within IMD ecosystems. As a result, different machine learning and deep learning frameworks have been proposed for real time threat detection. However, they still suffer from overfitting, slow inference, and excessive resource demands, hindering their effective integration into the IMD ecosystem. The study's primary goal was to design and develop a hybrid of deep autoencoders, convolutional neural networks, and long short-term memory (LSTM) strategies to provide a comprehensive detection model that reduces inference time for deployed models while enhancing performance. Autoencoders provide the fundamental architecture of the detection model, while convolutional neural networks are used in the encoder and decoder for simplicity and to capture nonlinear data effectively. The Long Short-Term Memory captures temporal dependencies in the model, enhancing overall detection capabilities. The study adopted an experimental approach, developing a hybrid deep autoencoder model to test its performance against convolutional neural networks, long short-term memory, and other conventional machine learning techniques. The results demonstrate that the hybrid model outperformed standalone models, achieving high accuracy scores across the datasets. The best model in the ICU dataset achieved 100% accuracy, precision, recall, and F1 score, and a false positive rate of 0.00%. The WUSTL had an accuracy of 79.32%, a recall of 79.92%, a precision of 79.41%, a specificity of 79.24%, and a false positive rate of 20.59%. The Edge IIoT dataset had a recall, F1, and accuracy of 96.87%, a precision of 96.94%, a specificity of 96.88%, and a false-positive rate of 3.12%. The model’s inference time was substantially reduced compared to the standard deep autoencoder model across the datasets, providing a lightweight detection environment for the intrusion detection system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AUC:	Area Under the Receiver Operating Characteristics Curve
CM:	Confusion Matrix
CNN:	Convolutional Neural Network
DAE:	Deep Autoencoders
FPR:	False Positive Rate
IMD:	Implantable Medical Devices
IoT:	Internet of Things
LSTM:	Long short-term memory
M-IoT:	Medical Internet of Things
ROC:	Receiver Operating Characteristics
MAE:	Mean Absolute Error
MSE:	Mean Squared Error
RMSE:	Root Mean Squared Error

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

The healthcare sector has widely implemented Implantable Medical Devices (IMDs) to expand patient access to services, improve remote patient monitoring, and enhance the management of chronic conditions. The performance of these devices is optimized through cloud and edge computing paradigms, reducing the need to manage data locally within existing healthcare infrastructures (Firouzi *et al.*, 2020). A leading implementation of these M-IoT devices is the IMD, a subcutaneous device that provides in vivo treatment. This phenomenon implies that the devices are deployed within the organism, enabling enhanced patient monitoring and condition response, and optimizing control for various diseases (Asghari *et al.*, 2019). Through this deployment, healthcare facilities remotely monitor patients, enabling real-time, effective responses based on device data.

This service delivery approach is convenient for senior citizens who experience significant issues with their organs and bodily functions. For instance, these individuals may have complications with their blood glucose, requiring continuous monitoring to respond to increases or decreases beyond acceptable levels (L. Wang *et al.*, 2021). Further, these patients may have complications with their hearts, requiring an implantable cardiovascular defibrillator to remedy the situation. This phenomenon requires these patients to regularly visit the healthcare facility when complications occur with their devices or organs. However, such hospital visits are often extensive, inconvenient, and impractical. This situation makes it challenging to ensure adequate healthcare services for patients outside hospitals. However, IMDs address these challenges by providing remote patient monitoring, offering a robust environment for identifying and resolving medical issues. This feature enhances the quality of life and optimizes service delivery, even for patients outside the facility.

Although this design approach refines the performance of the healthcare sector by improving patient-centered service delivery, it introduces new security challenges that compromise the efficiency of existing controls. Hackers may target the cloud platforms

or deployed IoT devices, gaining access to the underlying healthcare information infrastructures (Bhuiyan *et al.*, 2021). Such attacks are intended to access the patients' sensitive information, such as their payment details for credit fraud and identity theft, or addresses and personal information for cyber-bullying, blackmail, and extortion. Alternatively, hackers may compromise the IoT communication framework through advanced attacks, disrupting the efficiency of these ecosystems and hindering their ability to meet deployment requirements. These attacks divert attention from identity thefts, system intrusions, or other sophisticated attacks targeting the underlying information infrastructure. The attacks are considered intrusions because attackers can access sensitive information resources on the target devices illegally. Such security breaches adversely affect the sector as the deployed devices play a critical role in service delivery (Bhuiyan *et al.*, 2021). For example, the compromise of connected pacemakers jeopardizes the patient's safety, potentially aggravating the situation or leading to death. Given the critical security outcomes for the sector, such as cyber-attacks, there is a need for advanced security management frameworks that effectively deter attacks before they occur or contain them in their formative phases to protect users.

Heuristic and signature-based strategies have been proposed to handle these security needs, providing reasonable security management frameworks for the infrastructure. The major setback with these techniques is their need for updates to enhance relevance and viability (Y. Wang *et al.*, 2023). This phenomenon compromises the strategy's ability to handle zero-day malware. Machine learning-based approaches, such as autoencoders, have been proposed to address these limitations of the heuristic and signature-based technologies. The autoencoders leverage neural networks to identify and classify attacks targeting deployed Implantable Medical Devices, enabling timely detection of security incidents (Telikani & Gandomi, 2021). This detection model is supported by Aloul *et al.* (2021), who acknowledge the significance of cloud and edge computing in achieving these security objectives. Deep autoencoders have also been proposed to enhance detection accuracy for security breaches, substantially resolving the limitations of the conventional machine learning approaches (Siddiqui & Boukerche, 2021). This phenomenon demonstrates progress in addressing the core security challenges facing the Implantable Medical Devices ecosystem.

While these studies demonstrate the promise of autoencoders and deep autoencoders in addressing challenges in conventional security solutions, they also highlight three key limitations that offer opportunities for the current study. First, these studies leverage a general dataset to assess the features of the proposed models. They used datasets such as NSL-KDD, UNSW-NB15, and KDDCUP-99 for modelling threat detection in the Internet of Things (Shehadeh *et al.*, 2023). While these datasets are globally acclaimed for their comprehensiveness, they fail to provide the deployment-specific scenarios for the IMD ecosystems (Türk, 2023). Using domain-specific datasets yields higher accuracy and better generalization than using general datasets (Edwards *et al.*, 2020). This phenomenon improves the detection efficiency of the developed model, resulting in optimal performance in deployment scenarios. Additionally, this use of non-IoT-relevant datasets compromises the design considerations adopted for the proposed IDS solutions. The researchers emphasize the need for a host-based IDS solution installed on the monitored host devices (Shehadeh *et al.*, 2023). While such approaches are feasible with conventional computing solutions, they fail to account for the resource-constrained environments of IoT ecosystems. These connected platforms aim to enhance access to computing resources by reducing the need for complex processing frameworks (Kuchuk & Malokhvii, 2024). The outcome of such a platform is a resource-constrained system incapable of implementing a conventional host-based IDS.

Thirdly, the increasing adoption of Implantable Medical Devices in healthcare underscores the need for rapid threat-detection frameworks that enable healthcare service providers to identify attacks in real time and safeguard patients. Conventional security solutions fail to provide these functionalities because they rely on conventional datasets to train their models and neglect design considerations for deployment settings. This situation is complicated by the growing complexity of the IMD devices' attack surface, despite increased reliance on analytics for real-time decision support (Yogev *et al.*, 2023). Consequently, there is a need for a comprehensive, lightweight solution that addresses the design constraints of the IMD ecosystem while meeting industry requirements for healthcare security. This study sought to address these concerns by using an experimental approach to develop a hybrid deep autoencoder model for intrusion detection using the IoT Healthcare Security, WUSTL EHMS 2020, and Edge-

IIoT datasets. Using the two medical IoT datasets and the Industrial Internet of Things (IIoT) dataset improved model training. At the same time, the experimental strategy provided a hands-on assessment of the proposed framework to determine whether it meets the design goals. The outcome of this strategy was an expanded security framework that drew on insights from these attack datasets. The performance metrics used to test the developed solution were inference time, accuracy, precision, recall, and F1 scores. These metrics provided a comprehensive framework for evaluating the developed model against conventional deep learning strategies for intrusion detection.

1.2 Statement of the Research Problem

Although autoencoder-based models have been proposed for threat detection in Implantable Medical Devices, they have failed to meet their objectives due to limitations in accuracy and speed. First, these detection models suffer from overfitting due to a lack of IMD-specific datasets, limiting their generalization and accuracy. Secondly, there are inference time issues with the existing solutions. These detection models rely on slow autoencoders with long detection times, potentially allowing hackers to compromise devices before threats are identified. Lastly, conventional solutions lack robust detection capabilities due to resource constraints in implementation environments. The IMD devices adopt a minimalist design, reducing the processing and storage resources required for comprehensive security solutions. This phenomenon underscores the need for a lightweight, real-time solution that aligns with the IMD ecosystem while providing comprehensive threat detection capabilities. The lightweight nature of the intrusion detection system enables it to work with the resource-constrained IMD ecosystem. On the other hand, the real-time functionality provides a fast detection environment that allows prompt identification of threats in the IMD ecosystem. Delayed threat detection in the IMD ecosystem has devastating impacts on patients and healthcare service providers, as hackers may compromise patient devices before detection, leading to critical health implications, or gain unauthorized access to the underlying medical infrastructure and stage subsequent attacks. Consequently, a hybrid solution is needed to leverage the strengths of the existing autoencoder and deep autoencoder models while addressing their limitations, providing a quick, efficient, and lightweight detection environment for the IMD

ecosystem. Such a model addresses the accuracy and inference-time challenges faced by existing solutions, thereby improving accuracy and accelerating inference.

1.3 Objectives of the Study

The study was structured around defined objectives that established the framework for developing the intrusion detection system. The objectives established both the overarching research goal and the specific areas of focus required for the general objective. The general objective outlined the study's primary purpose. In contrast, the specific objectives detailed the technical and performance-oriented tasks needed to implement it, focusing on the development, optimization, and evaluation of the hybrid intrusion detection system for Implantable Medical Devices.

1.3.1 General Objective

The study aimed to design and develop a hybrid model integrating a deep autoencoder, a convolutional neural network, and a long short-term memory for intrusion detection in cloud-based implantable medical systems.

1.3.2 Specific Objective

i. To develop a hybrid intrusion detection model for IMD ecosystem using deep autoencoders, convolutional neural networks and long short-term memory.
ii. To optimize the developed model by integrating features from the convolutional neural networks and long short-term memory.
iii. To evaluate the inference speed, memory utilization and detection accuracy of hybrid model against autoencoder, long short-term memory and convolutional neural networks.

1.4 Research Questions

i. How can deep autoencoders, autoencoders, convolutional neural networks, and long short-term memory be used to develop a hybrid detection model for cloud-based IMDs while optimizing detection accuracy, inference speed, and memory utilization?

ii. How should convolutional neural networks, long short-term memory, and autoencoder models be integrated into a deep autoencoder model to optimize the performance of hybrid intrusion detection for cloud-based IMDs?

iii. How does the performance of the developed hybrid intrusion detection model incorporating convolutional neural networks, long short-term memory, and autoencoder compare with autoencoder, long short-term memory, and convolutional neural networks regarding speed, memory utilization, and detection accuracy?

1.5 Justification of the Study

The cloud-based implantable medical devices technology was selected due to its growing role in contemporary healthcare service delivery. First, cloud-based IMDs include diverse medical devices that deliver life-saving interventions and therapies to patients (Asghari *et al.*, 2019). These IMDs have revolutionized the management of numerous medical conditions by providing targeted interventions tailored to individual patient needs. Secondly, cloud-based IMDs improve the quality of life for patients living with chronic medical conditions (Asghari *et al.*, 2019). For individuals with conditions like heart failure, chronic pain, or urinary incontinence, IMDs offer relief from debilitating symptoms and functional limitations. The intervention for these patients does not require the healthcare provider's physical presence, since the devices are connected via cloud infrastructure.

Thirdly, cloud-based IMDs provide minimally invasive treatment options for patients who may not be candidates for traditional surgical procedures or who wish to avoid the risks and recovery time associated with invasive surgeries (Asghari *et al.*, 2019). IMDs offer patients greater comfort, faster recovery, and reduced healthcare costs by providing less invasive treatment alternatives. Further, cloud-based IMDs are instrumental in the long-term management of chronic medical conditions, providing sustained therapeutic benefits and disease control over extended periods (Firouzi *et al.*, 2020). IMDs empower patients to better manage their conditions by providing long-term disease management solutions. Lastly, the use of cloud infrastructure in the cloud-based IMD ecosystem presents both security challenges and opportunities. For instance,

the cloud platform introduces new attack surfaces for healthcare infrastructure, as hackers may compromise virtualized resources to access underlying information systems. On the other hand, cloud platforms offer highly scalable computing frameworks that are easy to implement and enable the implementation of elaborate security solutions that address the limitations of conventional IMD devices. Such computing opportunities offer alternatives for strengthening security posture in IMD ecosystems and healthcare information infrastructures.

The study provided a reasonable solution to the security challenges of cloud-based Implantable Medical Devices. First, the study contributed to the Cybersecurity Body of Knowledge, particularly in the context of IMD ecosystems. The extensive literature review on the subject, the assessment of security governance models for the IMD ecosystems, and the analysis of their strengths and weaknesses expanded the understanding of cybersecurity issues. Secondly, the study presented a hybrid intrusion detection model for the IMD ecosystem, enabling healthcare institutions to manage threats to their computing systems effectively. This feature offered a practical solution to the healthcare sector by addressing security needs and handling emerging threats (Kong *et al.*, 2022). Lastly, the findings obtained from the research were generalizable to the M-IoT and IoT ecosystems, enabling other industries to implement the developed threat detection model. Thus, the study provided a comprehensive environment for handling societal security issues.

1.6 Scope of the Study

This study was limited to designing and deploying a hybrid threat detection model for the Implantable Medical Devices ecosystem, targeting senior citizens using two or more IMDs to manage their condition. This phenomenon implies that the research focused on examining the current cybersecurity strategies implemented in the IMD sector to enhance resilience and responsiveness, and on assessing the features of those models. The IMDs considered in this scenario were connected to a wearable controller via a personal area network and communicated using low-energy protocols such as Bluetooth. The controller was connected to the healthcare facility's cloud portal and sent data to it

via the 4G network. Also, this device had Wi-Fi capabilities, enabling it to transmit data using Wi-Fi when the access point is in range.

The study focused on designing and optimizing an intrusion detection model based on deep autoencoders, convolutional neural networks, and long short-term memory. These components were integrated through feature-level fusion. This approach involved parallel feature-extraction modules that used a deep autoencoder to identify compact anomaly representations. The CNN part of the encoder and decoder was intended to identify spatial features within the traffic, while the LSTM bottleneck focused on temporal features. The lightweight Multilayer Perceptron classification head integrated into the model was designed to produce a final prediction. This feature-fusion approach reduced the controller's inference time while shifting the heavy analysis to cloud-based model training(Sinha *et al.*, 2025). This model was intended to identify threats to the IMD ecosystem at the application layer, specifically the Constrained Application Protocol (CoAP), and to notify administrators of such attacks through the interactive dashboard.

The developed model was intended to provide near-real-time threat detection capabilities by reducing inference latency while maintaining relatively high accuracy, precision, and recall. This detection design aligned with the IMD ecosystem’s goal of providing prompt identification of threats to safeguard patient safety and ensure timely response to potential attacks. The research evaluated the performance of the developed model against other models, such as convolutional neural networks, autoencoders, long short-term memory, multilayer perceptrons, random forests, and gradient boosting. The study did not focus on other aspects of cybersecurity in the IMD or healthcare infrastructure. This limited scope enhanced comprehensive coverage of the core security issues within the established time.

1.7 Assumptions of the Study

i. The Implantable Medical Devices implemented in the healthcare sector were managed from a centralized location within the healthcare institution.

ii. The IMD devices implemented in the healthcare sector communicated using the standardized Transmission Control Protocol and the Constrained Application Protocol.

iii. The healthcare organizations implementing the solution had an adequate traffic-monitoring environment, which enhanced the IMD's performance after implementation.

1.8 Operational Definition of Terms

Dataset: A dataset is the collection of data that identifies specific characteristics of the measured entity in a typical production environment.

Framework: A software platform that offers tools, libraries, and infrastructure for building, training, and deploying machine learning models

Generalization: The ability of a machine learning model to make accurate predictions on new data that it was not exposed to during the training process.

Model: A mathematical algorithm that learns from data and makes predictions based on the attributes derived from the dataset.

Testing: evaluating the machine learning model's performance and generalization capabilities, if the proposed model is deployed in the production environment.

Training: The process through which a machine learning model learns patterns and relationships in the dataset.

CHAPTER TWO
LITERATURE REVIEW

2.1 Development of the Hybrid Detection Model for Implantable Medical Devices

The evolution of Implantable Medical Devices (IMDs) within the broader Internet of Things (IoT) ecosystem has brought medical benefits but also heightened cybersecurity concerns. These devices have increasingly enhanced connectivity by leveraging embedded and interconnected systems and cloud-based infrastructure for remote monitoring and data exchange. However, this advancement has introduced new security threats and expanded the existing ones, posing significant challenges in establishing robust security frameworks for the infrastructure. This phenomenon raises the need for a comprehensive understanding of the enabling technologies for the IMD ecosystem, including the IoT, agent-based systems, and cloud computing architectures. Understanding these technologies, their strengths, and vulnerabilities offers an effective framework for developing a suitable intrusion detection solution for the IMD ecosystem.

2.1.1 Internet of Things Technology

The Internet of Things (IoT) technology provides a revolutionary computing environment that reduces the computational requirements for implementing robust systems. This functionality is achieved by reducing the computational and power requirements of computing systems while maintaining a reasonable data-processing communication environment that facilitates data transmission to a centralized controller (Ray, 2018). The resulting systems are regarded as embedded systems, comprising a minute computational environment and adopting a minimalistic design. This computational framework enables the implementation of computing systems in extreme conditions where conventional systems are infeasible. The implemented devices maintain their functionality and efficiency by communicating with remote or on-site controllers that handle the infrastructure's core communication and data processing requirements (Ray, 2018). This feature has enhanced the expansive adoption of these computational solutions across society, especially in the healthcare sector. Additionally, these devices are integrated into the existing enterprise information infrastructure through customizable control frameworks that facilitate information flow between the core information infrastructure and the IoT ecosystem (Goyal *et al.*, 2021). This feature

provides a comprehensive infrastructure management environment, enabling robust governance of deployed systems and consistent security operations across the organization.

The Internet of Things architecture is designed to facilitate the seamless data flow between devices and systems. It typically consists of four main layers: perception, network, middleware, and application. The Perception Layer is the foundational segment that facilitates seamless data flow between interconnected devices and systems (Sobin, 2020). Positioned at the forefront of the IoT ecosystem, this layer deploys a network of strategically placed sensors and actuators to collect diverse data from the surrounding physical environment. These sensors are meticulously designed to capture information from fundamental variables, such as temperature and humidity, to more dynamic aspects, such as motion and light (Sobin, 2020). However, the significance of the Perception Layer extends beyond mere data gathering; it serves as the critical interface between the physical world and the digital realm. The layer offers core functionalities of the IoT ecosystem, converting real-world phenomena into digital signals that can be comprehensively processed, analyzed, and transmitted (Ray, 2018). As the initial point of contact for data acquisition, the Perception Layer provides the raw input necessary for subsequent layers to perform intricate analyses, interpret patterns, and ultimately trigger informed, intelligent actions within the broader IoT infrastructure.

The Network Layer plays a critical role as the communication backbone within the intricate architecture of the IoT, shouldering the responsibility of transmitting data collected by sensors from the Perception Layer to centralized servers within the implementing organization or the cloud-based servers. This layer acts as a crucial intermediary, facilitating the efficient and secure transfer of information across the IoT ecosystem (Firouzi *et al.*, 2020). Its significance lies in orchestrating data transmission over diverse communication protocols, such as Message Queuing Telemetry Transport (MQTT) or Constrained Application Protocol (CoAP). These protocols are carefully selected to ensure reliable data transfer, while accounting for critical factors such as bandwidth, latency, and energy consumption. The Network Layer thus establishes a

robust, scalable infrastructure, serving as the connective tissue that enables data to traverse the network seamlessly (Firouzi *et al.*, 2020). Its efficiency in data transmission is integral to the timely and accurate conveyance of information, ensuring that data reaches its designated destination for further processing, analysis, and utilization within the broader scope of IoT applications and functionalities.

The Middleware Layer is a critical bridge between the network and application layers. The layer ensures seamless interoperability between the diverse devices and systems within the expansive IoT ecosystem. The Middleware Layer's responsibilities include data normalization, protocol translation, and routing, collectively contributing to the efficient processing and dissemination of information (Bhuiyan *et al.*, 2021). It harmonizes the complexities inherent in the data generated by interconnected devices, making it accessible and coherent for subsequent layers. The layer's significance is underscored by its ability to streamline the complexities of data management, providing a structured framework that simplifies access to and use of the voluminous datasets generated by the myriad interconnected devices (Bhuiyan *et al.*, 2021). This streamlined approach enhances data processing efficiency and reduces the burden on developers and applications, facilitating a more cohesive, intelligent integration within the IoT ecosystem.

The highest layer of the IoT architecture is the Application Layer, where specific IoT business services are developed and implemented. The layer harnesses processed and refined data from lower layers for monitoring, control, analytics, and automation (Kassab & Darabkh, 2020). This approach enables the layer to extract meaningful insights from extensive datasets, enhancing informed decision-making at the corporate level. The Application Layer translates raw data into actionable intelligence, shaping the corporate trajectory within the implementation sector. Whether optimizing energy consumption in an intelligent building, predicting maintenance needs in an industrial setting, or enhancing healthcare through remote patient monitoring, this layer becomes the center of innovation. Its inherent versatility empowers developers to develop applications tailored to the specific needs of various industries, thereby realizing the full potential of interconnected devices and systems within the IoT ecosystem (Kassab

& Darabkh, 2020). The Application Layer epitomizes the culmination of IoT's transformative journey, where data is collected and leveraged intelligently to drive meaningful advancements across a spectrum of fields. This organization of the layers is presented in Figure 2.1.

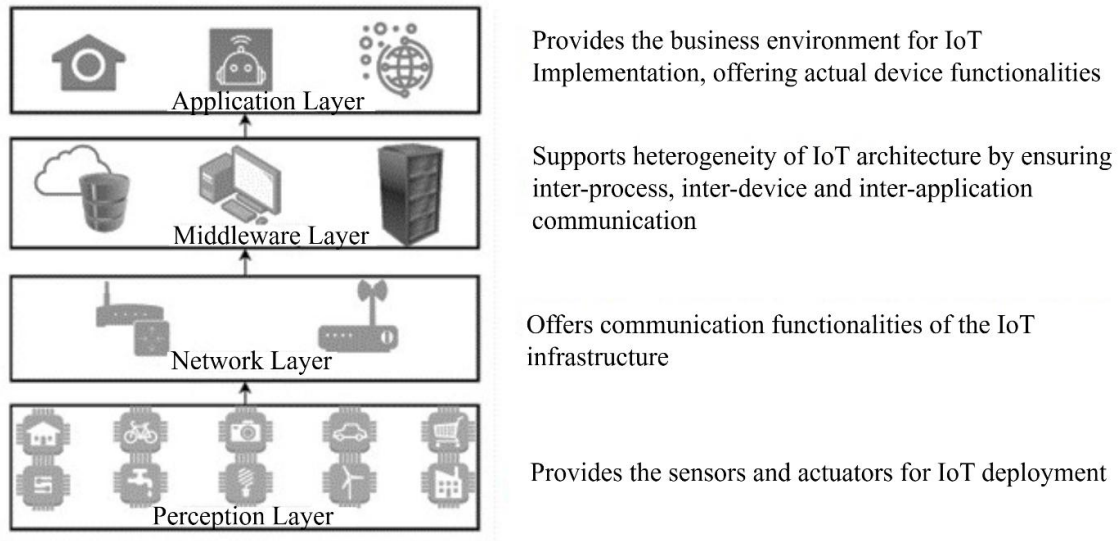


Figure 2.1: Internet of Things' Architecture (Kassab & Darabkh, 2020)

The success of implementing the Internet of Things depends on the enabling technologies that empower the seamless operation of interconnected devices. These technologies create a robust, responsive infrastructure that enables deployed devices to share data with controllers and respond to control signals. These enabling technologies include Wi-Fi, Bluetooth, Zigbee, RFID, and cellular networks, which are critical for facilitating efficient data transmission among devices (Dizdarević *et al.*, 2019). This interconnectedness ensures a continuous flow of information, enabling devices to communicate and collaborate intelligently. Data analytics is incorporated into the architecture to enable advanced analysis of data from the environment. This feature employs advanced technologies, including machine learning and artificial intelligence algorithms. The analytical tools sift through the large volumes of data generated by IoT devices, extracting meaningful insights (Dizdarević *et al.*, 2019). This capability not only supports informed decision-making but also enables predictive analytics, allowing organizations to anticipate trends, optimize processes, and enhance overall system efficiency across diverse industries.

2.1.2 Agent-Based Systems and Internet of Things Devices

Agent-based systems are autonomous computational systems that interact in a shared environment, enabling them to achieve specific business goals. These systems are deployed in dynamic, uncertain environments and rely on sensors and actuators to respond to such environments (Alkhabbas *et al.*, 2020). This response is defined by the agent's decision, enabling it to perceive its surroundings. Typically, multiple agents operate in a shared environment, each performing a designated role. A central registry is used to manage these devices, monitor their performance, and provide a remediation mechanism for deviant ones (Forestiero & Papuzzo, 2021). The role performed by the individual agents is linked to the collective objective of the deployed devices, enabling them to achieve the business goal. The resulting agents constitute a multi-agent system and exhibit intricate relationships among themselves (Savaglio *et al.*, 2020). This relationship is essential in providing a robust solution that achieves the shared business objective. The autonomous nature of these systems and their intelligent response to the environment enhance their performance in dynamic environments, reducing the need for human intervention in extreme deployment settings.

Agent-based computing has redefined the Internet of Things by endowing devices with intelligent capabilities, enabling the creation of autonomous and adaptive systems. The IoT devices are handled like multi-agent systems, with these systems demonstrating distributed intelligence in their deployment scenarios. The distributed nature of multi-agent systems enables them to extensively monitor the environment through sensors and respond appropriately using actuators (Alkhabbas *et al.*, 2020). The response to these environmental changes defines the devices' performance, shaping their adaptability to evolving settings. Additionally, individual IoT devices make decisions based on their perceptions of the environment, thereby shaping the overall coherence of the multi-agent system (Savaglio *et al.*, 2020). This phenomenon results in a distributed environment in which individual IoT devices respond differently within the IoT ecosystem, driven by different deployment considerations. A central device coordinates the performance of these IoT devices and provides a remediation framework for non-compliant devices (Forestiero & Papuzzo, 2021). This central device is considered the controller and is responsible for linking the IoT ecosystem to

the business information infrastructure. This resulting interrelationship between the IoT ecosystem and core business infrastructure controls the exchange of information across these segments. Thus, agent-based computing provides the core functionalities of the IoT ecosystem.

While agent-based systems offer opportunities to strengthen the application of IoT devices in the healthcare sector, they face significant challenges due to their design constraints. These IoT-based agent systems have limited computing resources, affecting the implementation of robust AI-based decision-making frameworks (Gupta, 2025). Inability to deploy such decision-making solutions compromises the efficiency of these systems in handling cybersecurity threats targeting their infrastructure, leading to security lapses. This phenomenon is further complicated by the inability of edge computing platforms to provide a robust resource environment for advanced AI-based threat detection solutions to meet the security needs of these systems (Chinta, 2024). As a result, there is a need for a lightweight security solution that can be easily implemented at the controller, close to the device, while ensuring high accuracy. This research seeks to provide such a solution, enabling the deployed devices to handle security threats related to their infrastructure effectively. Such an approach offers a lightweight solution for the IMD ecosystem without compromising accuracy or inference speed, meeting real-time detection expectations in deployment settings.

2.1.3 Medical Internet of Things and Implantable Medical Devices

The Internet of Things technology is extensively implemented in the healthcare sector as Medical Internet of Things (M-IoT), optimizing the quality of care. The M-IoT devices are deployed as wearable devices to monitor the patient's biomarkers or location, or as internal devices to monitor core organs like the heart and lungs. The healthcare service provider implements and manages these devices, with the patients being the primary recipients. The deployment of M-IoT devices optimizes healthcare service delivery, enabling medical professionals to respond promptly to patient issues (Kong *et al.*, 2022). The outcome of such deployment is improved quality of life and greater efficiency in healthcare services, optimizing managed care services for the sector. For instance, seniors with dementia are less likely to find their way back home after a walk.

This phenomenon is attributed to their memory challenges, affecting their recall of their residence. In such instances, discovering the individual becomes challenging for the nursing officer or the family. However, M-IoT addresses this challenge through location services, enabling the care provider to quickly locate patients and ensure they take their medication and meals as required (Habibzadeh *et al.*, 2020). Thus, M-IoT provides a robust platform for improving the quality of care.

Implantable medical devices are a crucial aspect of modern healthcare, providing innovative solutions for monitoring, diagnosing, and treating various medical conditions. The devices are inserted into the human body, providing an in vivo healthcare management environment that enables rapid response to ailments and continuous monitoring of the patient's condition (L. Wang *et al.*, 2021). These devices have sensors that track vital signs, detect anomalies, and collect real-time data on specific health indicators. For instance, cardiac implants such as pacemakers continuously monitor heart rhythms, providing healthcare providers with valuable information about a patient's cardiovascular health. This continuous monitoring enables early detection of issues, allowing for timely intervention and personalized care (L. Wang *et al.*, 2021). Furthermore, the devices contribute significantly to the diagnostic process by providing accurate, timely information. For example, neurostimulators and deep-brain stimulators are used to understand neural responses for therapeutic and diagnostic purposes. Implantable diagnostic devices may include sensors that detect specific biomarkers or changes in physiological conditions, aiding in diagnosing diseases and conditions ranging from neurological disorders to diabetes. Although the healthcare service provider manages these devices, they have semi-autonomous functionalities. This phenomenon implies that the devices may respond to patient needs without human intervention. However, the healthcare service provider may implement remediation measures to prevent the devices from going rogue.

The architecture for these devices leverages the Internet of Things framework, providing connected systems that monitor physiological processes. This phenomenon implies that the IMD devices rely on IoT devices' processing, storage, and communication functionalities to perform their functions. These devices leverage

sensors to monitor activities of critical organs, such as heart rate and blood sugar levels (Taleb *et al.*, 2021). The data collected from this monitoring is sent to the processing unit, which analyses the signals. The embedded memory stores historical data collected from the tracking and analysis, while the battery powers the entire device. When two or more devices are deployed on the individual, a wireless body area network (WBAN) is established to monitor their communication and functionalities (Taleb *et al.*, 2021). In such instances, these deployed devices function as multi-agent systems, each performing its designated role in overall patient health and safety. An external controller is implemented as a wearable device that coordinates the devices in the WBAN network. This controller acts as the bridge between the WBAN and the core healthcare infrastructure (L. Wang *et al.*, 2021). For instance, it monitors the traffic between the healthcare facility and the WBAN network. This monitoring regulates requests directed to the WBAN devices to safeguard their performance and efficiency. Further, it facilitates communication between the IMD devices and remote servers in the cloud or the healthcare data center. It enables the appropriate packet segmentation to enhance communication over the Internet Protocol-based network (Taleb *et al.*, 2021). Figure 2.2 illustrates how these devices are implemented, facilitating communication between the patient and the healthcare facility.

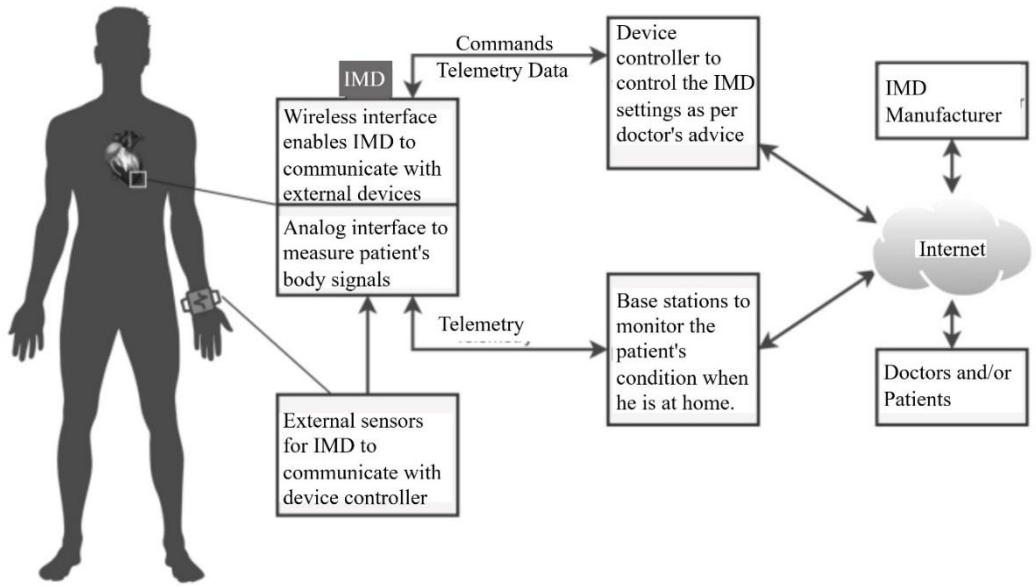


Figure 2.2: Deployment of IMD (L. Wang *et al.*, 2021)

There are diverse types of Implantable Medical Devices designed to address specific patient needs. These devices include cardiac, neurological, and insulin delivery implants. Cardiac implants encompass a range of devices designed to address heart-related issues. For instance, connected pacemakers are implanted to regulate heart rhythms by delivering electrical impulses (Siddiqi *et al.*, 2020). Also, implantable cardioverter defibrillators monitor heart rhythms and administer shocks to restore normal rhythm during life-threatening arrhythmias. Similarly, cardiac resynchronization therapy devices are designed to improve ventricular synchronization in patients with heart failure, thereby enhancing overall cardiac function. Neurological implants manage conditions affecting the nervous system (Siddiqi *et al.*, 2020). For example, deep-brain stimulators are implanted to alleviate symptoms of movement disorders like Parkinson's disease. The spinal cord stimulators help manage chronic pain by sending electrical signals to the spinal cord. At the same time, cochlear implants provide sound perception to individuals with severe hearing loss or deafness, significantly improving their quality of life. Implantable insulin delivery pumps offer a revolutionary approach to diabetes management. These devices are implanted under the skin to provide a continuous insulin supply, offering more precise control over blood sugar levels (Easttom & Mei, 2019). This technology reduces the burden of constant monitoring and injections for individuals with diabetes, enhancing their overall well-being.

Implantable medical devices are designed to deliver targeted, personalized patient treatment and are characterized by biocompatibility, durability, and specific size and shape considerations. Biocompatibility ensures that the body's materials are well-tolerated, minimizing the risk of rejection or adverse reactions. This feature reduces the overall risk associated with the implemented device, ensuring it addresses specific patient issues to optimize healthcare management (Easttom & Mei, 2019). Durability is crucial for long-term functionality, and the devices' size and shape are tailored to fit within the designated anatomical space. These considerations reduce the complexity of the deployed device, lowering its overall side effects on the body due to size and heat emissions. Further, it ensures that patient safety is incorporated into the design and deployment considerations, as the devices are tailored to minimize deployment

requirements with negligible disruption toody functions (Tarricone *et al.*, 2020). The devices are integrated with the broader Medical IoT ecosystem, providing enhanced healthcare capabilities. For instance, these devices optimize remote patient monitoring by enabling in vivo monitoring enabling real-time data collection. Thus, these devices offer a robust telehealth environment, revolutionizing healthcare service provision.

Patient interaction with data in Implantable Medical Devices is a critical aspect that shapes healthcare management and patient outcomes. Firstly, the collected data involves continuously monitoring physiological processes using sensors embedded in the IMDs (Sun *et al.*, 2018). For instance, a patient with a cardiac pacemaker may have their heart rhythm continuously monitored. The patient then uses this collected data for self-monitoring and for healthcare professionals to diagnose and plan treatment. Patient interaction with IMD data extends beyond mere observation to active participation in their healthcare. They leverage data from IMDs to make informed decisions about their lifestyle, medication adherence, and treatment adjustments (Kwarteng & Cebe, 2022). For example, a diabetic patient may use data from an insulin pump or continuous glucose monitor to adjust insulin dosages based on real-time glucose readings. The healthcare service provider relies on this data to enable remote patient monitoring and to provide timely responses to patient needs. For example, the healthcare facility may recalibrate the dosage administered to the patient based on the changing medical condition. Despite these interventions, the IMD devices remain autonomous in the multi-agent environment.

Challenges are associated with these devices despite their benefits for healthcare service provision. Firstly, these devices leverage the Internet of Things technology, which adopts a minimalistic design framework. This phenomenon compromises the ability to implement comprehensive security solutions as the computational resources are limited (Shah & Chircu, 2018). Balancing security requirements with minimizing power consumption and device size presents a significant challenge for IMD designers. Secondly, the interconnected nature of IMDs within the broader IoT ecosystem increases their susceptibility to cyber threats and vulnerabilities (Easttom & Mei, 2019). For instance, wireless communication interfaces used in IMDs for data transmission

may be vulnerable to interception or hacking, potentially compromising patient privacy and safety. Further, the decentralized nature of IMD design approaches poses significant challenges in ensuring optimal governance of the deployed devices (Forestiero & Papuzzo, 2021). The IMD devices interact autonomously, without centralized control, which complicates the implementation of robust security measures. While the controller may provide a semblance of monitoring in the WBAN network, its functionality is limited compared to that of typical local area networks due to the intelligence of the IMD devices. These factors compromise the effective governance of deployed IMD devices.

2.1.4 Cloud Computing Technology and the Internet of Things

The design limitations of Internet of Things devices compel them to leverage cloud computing, providing a robust, remote storage and processing environment for intensive data analytics in business operations. IoT devices are engineered to be compact, energy-efficient, and lightweight, thereby limiting their computational capabilities and storage capacity (Abou-Nassar *et al.*, 2020). These devices prioritize efficiency and portability, limiting onboard data storage and processing resources. For example, wearable IoT devices such as fitness trackers are small and have limited battery life, making it challenging to include large-scale on-board storage. As a result, it becomes difficult to implement large-scale physical memory in these devices. This phenomenon compromises the ability to deploy large-scale solutions in these systems. For instance, due to storage and processing constraints, comprehensive data analytics solutions for multi-agent systems are infeasible in the typical IoT ecosystem. This feature reduces these devices' optimal functionality despite their promises to redefine business operations.

The cloud infrastructure provides a suitable framework for offloading resource-intensive tasks, such as data storage and processing. Cloud technology is a revolutionary computing strategy that focuses on remote access and the provisioning and management of computing resources (Azimi *et al.*, 2018). This provisioning approach is enabled by virtualization technology, which abstracts underlying hardware resources to provide a highly scalable, customizable, and flexible business computing

environment. Virtualization abstracts hardware resources, allowing users to access them through software-defined environments without needing to understand the location or identity of individual hardware components (Azimi *et al.*, 2018). The virtualized environment eases the management of computing resources by fostering collaboration and remote access, enabling users to access them regardless of location. The distributed nature of this virtualization technology establishes the cloud infrastructure, allowing users to access virtualized resources through a managed platform. These services are customized to meet specific user requirements, enhancing the flexibility and scalability of the underlying virtualization technology.

Integrating the Implantable Medical Devices with the cloud infrastructure provides a robust healthcare management framework that enhances service delivery. Cloud computing offers a scalable, elastic infrastructure that complements the dynamic, distributed nature of Implantable Medical Devices (Longras *et al.*, 2020). This architecture offloads data storage and processing analytics, enabling the deployed Medical IoT devices to perform their functions optimally. This approach allows these devices to conserve local resources and operate more efficiently. The cloud services facilitate real-time data analysis, enabling healthcare organizations to extract meaningful insights from the vast amounts of data generated by IMDs (Rahmani *et al.*, 2018). The scalability of cloud infrastructure accommodates the exponential growth of IMDs and M-IoT devices, ensuring that the system can handle increasing data loads. Additionally, cloud-based services enable seamless remote management, updates, and monitoring of IMDs, enhancing overall system flexibility and responsiveness (Rahmani *et al.*, 2018). The cloud's accessibility, cost-effectiveness, and ability to handle diverse workloads make it a critical enabler for realizing the full potential of IMDs and M-IoT applications in the healthcare sector.

The integrated Implantable Medical Devices and cloud infrastructure comprise three main segments, with each component addressing specific computing and resource needs in the healthcare sector. The first segment is the device layer, which is the physical device implemented in the patient (Farahani *et al.*, 2017). This section comprises IMD devices, such as pacemakers, implantable cardioverter-defibrillators,

and neurostimulators. The layer provides in vivo patient monitoring, data collection, and condition tracking (Farahani *et al.*, 2017). The second layer is the fog node, which provides processing capabilities near the patient. The section comprises the controllers with storage, networking, and processing capabilities, enabling the implemented IMDs to handle large processing requirements without increasing complexity (Farahani *et al.*, 2017). For instance, it could include a wearable device with storage and processing capabilities to enable the deployed IMD to temporarily store and process data before redirecting it to the cloud platform. As a result, the layer interfaces the implanted devices with the cloud-based servers, reducing overall latency in cloud-based data processing while ensuring optimal performance of these devices. This approach strengthens healthcare data management by providing a robust, reliable framework for responding to patient needs. The last section is the cloud layer, which comprises the remote healthcare management infrastructure implemented to leverage the IMD devices (Farahani *et al.*, 2017). Data from other healthcare devices and systems, such as electronic health records and corporate data, is integrated at this layer, providing a seamless healthcare management environment. The organization and relationship of these layers are shown in Figure 2.3.

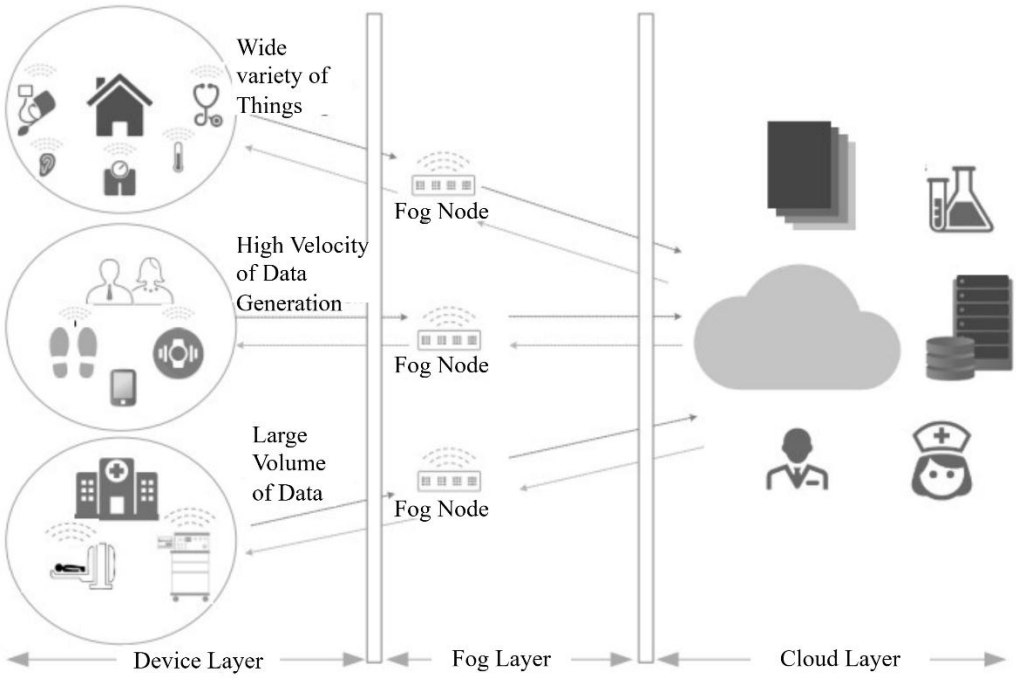


Figure 2.3: Healthcare cloud architecture (Farahani *et al.*, 2017)

2.1.5 Risk Classification

Risk classification is an integral aspect of threat management for Implantable Medical Devices, as it determines the priority of threat response. This approach is essential for determining the appropriate resources to mitigate threats, including financial and human resources. Risk classification in security systems involves identifying the degree of risk, the probability of occurrence, and its implications for the information infrastructure. There are two risk classification approaches: qualitative and quantitative. Qualitative risk management involves the subjective assessment of risks based on their perceived impact and likelihood. This approach relies on expert judgment and categorizes risks into predefined levels, such as low, medium, or high. Quantitative risk management utilizes numerical values and statistical techniques to quantify risks in terms of their probability of occurrence and potential impact. This approach provides a more rigorous, data-driven assessment, enabling more precise risk prioritization. Each approach develops a risk register, prioritizes risks, and identifies possible mitigation measures based on risk level and threat nature.

The selected risk classification approach is the qualitative risk management framework. This risk-scoring mechanism is chosen due to its simplicity and ease of implementation in the intrusion detection system. This phenomenon makes it effective to integrate the risk scores into the classification report. Implementing this risk-scoring mechanism involves determining the frequency of attacks from a single Internet Protocol (IP) address or network address relative to the overall attacks identified in the system. First, the number of requests to the device within a minute is determined for the IMD ecosystem. This number is the sum of all benign and malicious service requests to the IMD devices within the ecosystem. Secondly, the number of malicious requests to the IMD ecosystem within a minute is obtained. This number denotes the illegitimate service requests that compromise the confidentiality, integrity, or availability of information services on the IMD ecosystem. These illegitimate requests are identified by analyzing the nature of traffic targeting the specific IMD device. For instance, if the request that bypasses the conventional access mechanism does not provide a sequence number that is consistent with the previous messages, or involves unauthorized modification of the message, it is considered invalid. Also, the number of benign cases

within a minute is obtained. A benign request is the traffic that conforms to the access requirements established for the ecosystem. The third phase is to get the percentage of malicious requests among all requests within a minute. This percentage indicates the likelihood of an attack. It is expressed using the following formula:

$$\text{Percentage of attacks} = \frac{\text{Number of malicious cases}}{\text{Number of malicious cases + Number of benign cases}} \quad \text{(i)}$$

$$(P_A) = \frac{C_m}{(C_m) + (C_b)} \quad \text{(ii)}$$

The percentages obtained from this calculation are used to create the frequency distribution table for the attacks. This frequency distribution table is instrumental in developing the risk classification scheme for the intrusions. For instance, higher percentages are considered more severe, while lower percentages are considered less severe. This percentage-based risk classification provides a reasonable risk-prioritization scheme that determines suitable mitigation measures based on attack occurrence rates. For example, higher risk percentages require immediate mitigations and extensive resource allocation, while the lowest risk percentages are handled after the critical ones are resolved. This approach ensures optimal security governance across the infrastructure by effectively distributing resources within the IMD ecosystem. Additionally, recommendations for the attacks are proposed based on these percentages and the threat's nature, providing a comprehensive threat management framework for the IMD ecosystem. This risk-scoring strategy is presented in Table 2.1.

Table 2.1: Risk scoring table

Percentage	Risk Value	Risk Score
81-100	5	Critical
61-80	4	High
41-60	3	Medium
21-40	2	Low
1-20	1	Negligible

The risk scoring table is used to create the risk classification framework. This framework identifies the risk implications on the IMD ecosystem and the priority level for the classified risks. The critical risk score had the highest impact on the IMD

infrastructure. This situation makes risk management the highest priority in the IMD ecosystem. The negligible risk score had the lowest impact and priority in risk management. This classification scheme is shown in Table 2.2.

Table 2.2: Risk classification framework (Chandra *et al.*, 2022)

Risk Score	Risk Description	IMD Implications	Priority Level
Critical	Critical compromise of the IMD infrastructure	Severe data unavailability	1
High	Significant compromise of the IMD infrastructure	Operational delays	2
Medium	Moderate compromise of the IMD infrastructure	Intermittent disruptions	3
Low	Relatively low compromise of the IMD infrastructure	Minor operational disruptions	4
Negligible	Very low or unnoticeable compromise of the IMD infrastructure	Minimal operational impact	5

2.1.6 Validation and Evaluation Metrics

Validation approaches and evaluation metrics are integral in evaluating the effectiveness and reliability of intrusion detection systems. Validation strives to assess the model's ability to handle unseen data (Varoquaux & Colliot, 2023). Validation prevents overfitting, in which a model fails to generalize to new datasets. On the other hand, evaluation metrics focus on assessing the model's prediction behaviour on the target dataset (Mwitta *et al.*, 2024). This assessment is based on the model's ability to identify the classification groups required for the specific dataset. Validation and evaluation approaches play a critical role in assessing IDS solutions as they provide a comprehensive mechanism for determining the model's performance. For instance, validation approaches ensure the model does not overfit and performs well on unseen datasets, enabling it to handle zero-day attacks in deployment settings effectively. The evaluation metrics help determine the model's suitability for the target dataset by assessing its performance during training or testing.

Various validation techniques have been proposed to ensure robust model evaluation during dataset refinement. One such approach is cross-validation, which focuses on

model development and dataset validation. Cross-validation methods such as k-fold and leave-one-out cross-validation are widely employed (Cheng *et al.*, 2021). These techniques partition the dataset into multiple subsets, enabling iterative training and testing, providing a comprehensive assessment of model performance, and mitigating the risk of overfitting. Also, the holdout method involves splitting the dataset into distinct training and testing sets (Pal & Patel, 2020). This approach may introduce variance from random data selection, thereby affecting the reliability of the evaluation. Stratified sampling techniques address imbalanced datasets by ensuring proportional representation of each class in the training and test sets. This strategy minimizes bias, thereby enhancing the model's generalization capabilities.

Two evaluation strategies were considered in assessing the proposed IDS solution. The first approach is the time required for the model to classify threats, presented as either training or inference time. Training time is the time a model takes to classify the dataset during training, and inference time is the time it takes to classify the test dataset. Assessment of training and inference time is essential for determining resource utilization, as it directly maps to computational costs, model deployment, and energy consumption. Mwitta *et al.* (2024) evaluated per-image inference latencies and used these findings to determine the model that was suitable for real-time robotic deployment. This phenomenon implies that inference time plays a critical role in shaping real-time IDS solutions, like those deployed in IMD ecosystems. In separate research, Fernandez *et al.* (2025) showed that inference-time optimization affected energy use in the developed model. These researchers noted that optimizing inference time reduced total energy use by up to 73%, enhancing the sustainable deployment of Large Language Models (LLMs) and the energy efficiency of deployed solutions (Fernandez *et al.*, 2025). This finding demonstrates that reducing inference time while maintaining other considerations enables lightweight solutions to deliver near-real-time detection capabilities in mission-critical scenarios such as the IMDs.

The second evaluation approach for the IDS is based on the confusion matrix. The confusion matrix is a critical tool for evaluating machine learning algorithms. This tool provides a snapshot of the model’s performance, effectively capturing the essential

performance features that determine the suitability of the developed model. The confusion matrix comprises four key components: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). The TP is the number of correctly identified malicious requests, while TN is the number of correctly identified benign requests in the IMD ecosystem. The FP is the number of benign requests that are identified as malicious. At the same time, FN is the number of malicious requests identified as benign within the IMD environment. Table 2.3 shows this confusion matrix.

Table 2.3: Confusion matrix

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

Different metrics were derived from the confusion matrix, providing an expanded assessment of the proposed model. The first metric was accuracy, which measured the model's correct classifications relative to the actual classifications (Arias-Duart *et al.*, 2023). This metric assessed the model's ability to correctly identify incidents as either malicious or benign using the provided dataset. The second metric was precision, which evaluated the model's positive predictions against the actual number of positivecases (Arias-Duart *et al.*, 2023). This metric checked the model's ability to avoid false positives in the dataset. It is also known as the False Positive Rate, as it measures the model's false positive rate. The third metric was recall, which evaluated the number of positive predictions the model made in the dataset (Arias-Duart *et al.*, 2023). The metric accurately assessed the model's ability to predict threats in the Implantable Medical Devices ecosystem. This assessment, also known as the sensitivity (True Positive Rate), compared the number of positive predictions with the actual positive instances in the dataset. The last metric was the F1 score, which provided the harmonic mean between recall and precision (Arias-Duart *et al.*, 2023). This metric evaluated the model's overall performance using recall and precision, assessing its efficiency in both true-positive and false-positive scenarios. These metrics and their corresponding formulae are provided in the equations below:

$$\text{Inference Time} = \text{Time After Testing} - \text{Time Before Testing} \quad \text{(iii)}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{(iv)}$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{(v)}$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{(vi)}$$

$$\text{F1 Score} = \frac{2TP}{2TP + FP + FN} \quad \text{(v)}$$

2.1.7 Implantable Medical Devices and Healthcare Cybersecurity Landscape

Security challenges associated with Implantable Medical Devices stem from the design constraints of the IoT infrastructure. IoT devices' limited processing and storage functionality and the heterogeneous nature of their communication frameworks make it challenging to effectively implement conventional security solutions (Farahani *et al.*, 2017). This phenomenon renders such security solutions unsuitable for the IMD ecosystems, exposing the infrastructure to security breaches (Butt *et al.*, 2019). There are diverse categories of these security attacks, each addressing specific issues in the design considerations for the IMDs. The first category is denial-of-service, a security breach intended to compromise the performance of deployed devices. These attacks exhaust existing resources to prevent legitimate use of the device, rendering its implementation irrelevant for addressing patient needs. This attack poses critical issues for patients due to the device's vital role. For instance, in 2010, researchers noted that any IMD was prone to denial-of-service attacks, which reduced the battery life of the targeted device (Hassija *et al.*, 2021). This attack compromised the device's ability to meet its deployment goals, disconnecting patients from remote healthcare services. In 2016, Marin and others demonstrated a successful attack on the ICDs, compromising their availability (Hassija *et al.*, 2021). In a separate 2018 review, Marin and other researchers noted that neurostimulators were susceptible to multiple messages without a serial number, leading to battery drain and compromise of the target devices.

The second category is integrity attacks, designed to compromise the information relayed between communicating nodes. These attacks target the personal area network to compromise communication between in vivo devices and the controller, or between the controller and the remote cloud location. The success of such attacks negatively impacts the response to the patient's condition, aggravating the medical situation. For instance, in 2015, Billy Rios controlled Hospira's Symbiq infusion pump, potentially leading to over- or underdosing for patients (Hassija *et al.*, 2021). A separate review identified a replay attack on an Implantable Cardioverter Defibrillator, where the attacker replayed a previously sent message (Hassija *et al.*, 2021). This attack compromised patients' safety and privacy, jeopardizing the sector's security goals. In 2018, the Department of Homeland Security identified vulnerabilities in Medtronic cardiovascular defibrillators, emphasizing the possibility of an attacker taking complete control of the device post-implantation (Hassija *et al.*, 2021). Thus, these attacks pose a significant challenge in ensuring the end-to-end integrity of the messages relayed in the IMD ecosystem.

The last category is privacy attacks, which aim to compromise the confidentiality of information exchanged between communicating devices. Confidentiality in healthcare service delivery is critical, as patient records must be protected at all costs. Privacy attacks on the IMD ecosystem target the design limitations that deter the implementation of elaborate encryption and access control mechanisms for the devices. This phenomenon potentially exposes sensitive patient information, enabling hackers to misuse such information. For example, the Nuffield Council on Bioethics highlighted the risk of privacy and security compromise by hacking pacemakers in 2019, exposing patients' sensitive data (Hassija *et al.*, 2021).

While these security attacks adopt divergent approaches, they leverage the communication protocols implemented in the IMD ecosystem. This situation implies that attackers send malicious packets over the internet to mount denial-of-service, integrity, or privacy attacks, with the receiving device being compromised due to the lack of an effective detection mechanism to identify and flag such packets. Thus, these security incidents collectively underscore the urgent need for robust cybersecurity

measures in developing and deploying IMDs to safeguard the well-being and privacy of patients who rely on these critical medical devices. These attacks are summarized in Table 2.4.

Table 2.4: Summary of IMD Attacks

Threat/Attack	Vulnerabilities Highlighted in Attacks	Probability of Attack Occurrence	Severity/Impact of Attack on Occurrence
Denial of Service (DoS)	Exhaustion of resources (e.g., battery life)	High: Documented occurrences (e.g., 2010 general IMD susceptibility, 2016 ICD attacks, and 2018 neurostimulator attacks)	High: Compromising device performance and availability disconnects patients from essential remote healthcare services.
Integrity Attacks	Compromised personal area network communication and controller-to-cloud communication links	High: Multiple documented cases (e.g., 2015 Hospira's Symbiq infusion pump, replay attacks on Implantable Cardioverter Defibrillator, and 2018 Medtronic cardiovascular defibrillator vulnerabilities)	High: Negatively impacts patient care, safety, and privacy, potentially leading to incorrect medical responses.
Privacy Attacks	Lack of encryption and access control mechanisms	High: Documented cases (e.g., 2019 pacemaker hacking risk identified by the Nuffield Council on Bioethics)	High: Exposes sensitive patient information, risks misuse of personal data

2.1.8 Cybersecurity Solutions for Implantable Medical Devices

Conventional security models are designed to address these threats in the enterprise environment, enabling healthcare service providers to ensure the safety and reliability of deployed devices substantially. These solutions are hybrid, integrating multiple features to address security threats. This design approach enhances comprehensiveness in data protection. The outcome of such consideration is increased accuracy in threat detection, leading to better performance than the individual approaches (de Souza *et al.*, 2020). Such improved performance enables the organization to diversify the detection capabilities, leading to better results, enhanced data protection, and increased efficiency of the security solution. Further, hybrid security solutions provide granular control and

increased flexibility in security governance (Naseer, 2020). Hybrid security solutions enable organizations to implement granular control over information resources and explicitly define user interactions across the infrastructure. Further, it provides an adaptive security governance environment that enhances flexibility and scalability (Naseer, 2020). As new security issues emerge, the solution is redefined to address these issues.

Diverse hybrid security solutions are proposed to address the threats facing Implantable Medical Devices. For example, Thamilarasu *et al.* (2020) proposed effective security models using user authentication and authorization, data encryption, device monitoring, and backup services. Integrating these solutions into IoT ecosystems redefines data access and resource management strategies in the healthcare sector, improving resilience and incident response (Thamilarasu *et al.*, 2020). Further, cloud computing frameworks have been proposed as robust security solutions for the IMD and IoT ecosystems. This computing paradigm enables healthcare organizations to expand security functions for deployed IMD devices by leveraging edge and cloud computing to manage the resources required for advanced security controls (Thamilarasu *et al.*, 2020). The machines are then integrated with these security models, providing robust security governance in the healthcare sector. This approach substantially addresses the resource limitations of the IMD devices, enabling the deployment of extensive security models as if they were natively running in the IMD ecosystem. Other proposed security solutions are secure communication protocols that allow robust authentication and validation. For instance, constrained application protocol, message queue telemetry transport, and extensible messaging and presence protocol have been proposed for secure communication in the Medical IoT ecosystem (Bhuiyan *et al.*, 2021). These protocols are intended to enhance the safety of the communication process, extending features implemented in the underlying infrastructure.

Despite these advances, there are challenges to the commercial security strategies adopted for the implantable medical device infrastructure. First, these models emphasize resilience, ignoring the infrastructure's responsiveness to security breaches through real-time detection. This phenomenon implies that healthcare institutions are

investing heavily in preventing attacks on infrastructure, but have not defined strategies to respond if existing controls are bypassed. The absence of such expansive security governance strategies exposes the IMD ecosystems to massive corporate losses if hackers succeed in compromising access controls. For instance, if hackers bypass the encryption frameworks implemented in the cloud infrastructure by attacking the underlying cloud resources, they have unrestricted access to the hosted information resources (Selvaraj & Sundaravaradhan, 2020). Similarly, if hackers compromise the user authentication and authorization mechanisms by attacking the access control frameworks, they gain elevated privileges on the target systems.

Such elevated access to the system enables attackers to compromise it further, leading to three possible impacts. First, hackers may disrupt the normal performance of deployed IMD devices, affecting healthcare service delivery (El-Bakkouri & Mazri, 2020). Secondly, the attacker may compromise patient privacy during the breach, exposing sensitive medical records. Thirdly, affected patients and organizations may file lawsuits targeting the healthcare service provider, seeking compensation for the damages incurred in the attack. Such incidents underscore the need for comprehensive, real-time threat-detection frameworks that enable healthcare and IMD solutions to identify attacks targeting infrastructure quickly. This prompt identification and real-time monitoring allow safe responses when preventive controls fail, strengthening cybersecurity resilience and guaranteeing patient safety and infrastructure reliability (Catuogno & Galdi, 2024). However, the absence of such capabilities in these conventional security solutions exposes the infrastructure to attacks that bypass the perimeter controls (Messinis *et al.*, 2024). Thus, these solutions fail to meet the security expectations of the IMD ecosystem, exposing the critical systems to breaches that jeopardize patient safety.

The resource constraints and real-time monitoring requirements for the IMDS are unfeasible with these conventional security solutions. The design considerations of these conventional security solutions focus on resource-rich environments in standard computing systems. For instance, these frameworks require extensive computing resources to implement and maintain, which is infeasible in the IMD ecosystems due

to their resource-constrained nature (Slimani *et al.*, 2024). This phenomenon makes it challenging to safeguard the IMD ecosystem despite significant investments in security frameworks. As a result, these solutions require redesign to incorporate lightweight approaches that address resource constraints within the IMD ecosystem (Chandekar *et al.*, 2025). These limitations of existing conventional IMD solutions raise the need for advanced artificial intelligence solutions that automate threat monitoring in the IMD ecosystem, while providing lightweight functionality without compromising their accuracy or inference speed. These advanced AI security solutions leverage hybrid approaches that combine multiple models to expand the threat-detection framework.

2.2 Optimization of the Hybrid Intrusion Detection Model

Intrusion detection systems for healthcare settings need to detect threats in real-time, while ensuring optimal performance in terms of detection accuracy and inference time. This performance depends on the activation functions used in the machine learning models and on their optimization with respect to different parameters. Model activation enables the machine learning model to handle the input and generate the output based on the normalization parameters integrated through the function. On the other hand, optimization enhances predictive accuracy, reduces inference time, and minimizes the computational resources required for the model to run successfully. Understanding the fundamentals of these functions helps in developing robust machine learning models that enhance threat detection in healthcare settings.

2.2.1 Activation and Optimization Overview

Activation functions introduce nonlinearity to the model's output, which is critical for improving performance. These functions are applied to each neuron's output in a layer, allowing the network to learn complex patterns and relationships in the data (Dubey *et al.*, 2022). They play two critical functions in developing a robust intrusion detection model for the Implantable Medical Devices ecosystem. Firstly, they enable the model to capture non-linear relationships within IMD data, which is crucial for detecting subtle anomalies or intrusion patterns (Hayou *et al.*, 2019). By transforming input data into a higher-dimensional space, activation functions facilitate the extraction and representation of essential features for intrusion detection. Secondly, activation

functions are crucial in gradient propagation during training, ensuring stable and efficient learning (Hayou *et al.*, 2019). For instance, activation functions like ReLU help mitigate vanishing gradients, leading to faster convergence and better model performance. Thus, activation functions are instrumental in enhancing learning for the proposed model.

On the other hand, optimization refines a model's performance to achieve the best possible outcomes for a given task. It involves systematically adjusting model parameters and hyperparameters to minimize errors or loss functions, thereby maximizing predictive accuracy or effectiveness (Soydaner, 2020). Optimization ensures fast performance, high accuracy, and minimal memory utilization in the context of intrusion detection models within the Implantable Medical Devices ecosystem, offering a robust detection model for threats targeting the devices. Two objectives necessitate optimization of the detection model in the IMD ecosystem. First, there is a need for the highest possible accuracy in identifying security threats in the IMD environment. This phenomenon indicates that the developed model effectively identifies existing and novel attacks targeting diverse devices within the infrastructure. Optimization helps to achieve this objective by fine-tuning model parameters and algorithms (Soydaner, 2020). This fine-tuning improves classification and detection accuracy for the developed model, resulting in greater efficiency in identifying attacks.

Secondly, the need for efficient resource utilization drives model optimization. The IMD ecosystem adopts a minimalist design approach, reducing processing and storage requirements. This phenomenon reduces the speed, efficiency, and overall performance of deployment scenarios compared to conventional computing systems. However, optimization fine-tunes the developed model, enabling it to function effectively in this deployment environment (Soydaner, 2020). As a result, it reduces power consumption, resource requirements, and other factors that compromise the overall implementation efficiency. Lastly, optimization strengthens the overall accuracy and dependability of the developed model, enabling it to achieve the design goals (Soydaner, 2020). This feature is achieved by reducing the overall false-positive score in the model, thereby improving the detection accuracy and system reliability. Thus, optimization plays a

critical role in refining and strengthening the performance of the developed IMD detection model.

2.2.2 Activation Models

Various activation functions are used to improve the performance of machine learning models. These functions include the Rectified Linear Unit (ReLU), Sigmoid, Softmax, hyperbolic tangent, Exponential Linear Units (ELU), Scaled Exponential Linear Units (SELU), and leaky Rectified Linear Unit (Leaky ReLU). The ReLU introduces non-linearity by thresholding the input at zero: it outputs zero for negative values, and the input value for positive values (Dubey *et al.*, 2022). ReLU helps alleviate the diminishing gradient problem, enabling faster convergence during training. The leaky ReLU prevents neurons from dying by allowing a slight, non-zero gradient for the negative inputs. This feature is effective in convolutional neural networks, enabling the model to effectively extract feature maps from the input data (Dubey *et al.*, 2022). Further, it enables fast inference during model development, enabling the model to quickly predict based on the provided dataset. The tanh activation function maps input values to [-1, 1], making it well-suited for modeling data with both negative and positive values. Tanh provides strong non-linearity, enabling better gradient flow during backpropagation (Dubey *et al.*, 2022). This phenomenon makes it suitable for long-term short-term memory models, where it regulates information flow through the gates over time.

The SELU activation function maintains the mean and variance of its activations close to 0 and 1. This approach enables self-normalization in neural networks, stabilizing and accelerating training. The activation function operates on both positive and negative regions, effectively mitigating the vanishing gradient problem in ReLU (Kilicarslan *et al.*, 2021). The ELU uses negative values via an exponential function (Dubey *et al.*, 2022). This consideration aims to address the vanishing gradient problem encountered by ReLU. The outcome of this approach is the resolution of the model's self-normalization, resulting in improved accuracy. The sigmoid activation function maps input values to the range [0, 1], making it suitable for binary classification or tasks that require output probabilities. Sigmoid is commonly used in the output layer of binary

classification models to produce class probabilities (Dubey *et al.*, 2022). In the proposed model's autoencoder component, Sigmoid activation effectively improves decoder performance, providing a robust environment for input reconstruction. This phenomenon ensures that these inputs are within the expected range. The Softmax activation function is used in the output layer of multi-class classification models to produce class probabilities that sum up to one. Softmax normalizes the output scores across multiple classes, making it suitable for tasks involving multiple mutually exclusive classes (Dubey *et al.*, 2022). Softmax activation is effective at predicting probability distributions over multiple intrusion classes, enabling the identification of different types of security threats in IMD ecosystems.

2.2.3 Optimization Models

There are diverse optimization models designed for machine learning systems. Some of these algorithms are Adaptive Moment Estimation (Adam), Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSProp), Nesterov-accelerated Adaptive Moment Estimation (Nadam), and Adaptive Gradient Algorithm (Adagrad). Adam is an optimization algorithm that combines adaptive learning rates with momentum to accelerate convergence and improve the efficiency of gradient-based optimization (Soydaner, 2020). By adapting the learning rates for each model parameter based on the first and second moments of gradients, Adam is particularly well-suited for training deep neural networks. The Nadam optimization algorithm integrates Nesterov momentum with Adam's adaptive learning rates (Soydaner, 2020). This combination harnesses the advantages of both momentum optimization and adaptive learning rates, making it particularly suitable for optimizing the training process of deep neural networks.

The Adaptive Gradient Algorithm adjusts the learning rates of model parameters based on each parameter's historical gradients. It allocates more significant updates to infrequently occurring parameters and minor updates to frequently occurring ones (Soydaner, 2020). Adagrad is effective in sparse data settings and requires minimal hyperparameter tuning. The Stochastic Gradient Descent iteratively updates model parameters based on gradients computed on mini-batches of training data. Despite its

simplicity, SGD offers a robust training environment for neural networks, especially in scenarios with limited computational resources (Soydaner, 2020). The Root Mean Square Propagation adjusts the learning rates of model parameters based on the moving average of squared gradients. This approach mitigates the diminishing learning rates observed in Adagrad by using a decaying average of past gradients (Soydaner, 2020). RMSProp is robust to noisy gradients and well-suited for non-stationary objectives. Thus, these optimization algorithms are instrumental in strengthening the performance of deep learning models.

2.3 Comparative Analysis of Deep Autoencoders and Related Models

Understanding the efficiency of different machine learning and deep learning strategies is essential for developing a robust intrusion detection framework for Implantable Medical Devices. Such an intrusion detection system leverages the strengths of constituent models and mitigates the limitations of standalone models by incorporating hybrid functionality. This section provides a comparative assessment of conventional machine learning and deep learning strategies, demonstrating how these models differ in their learning capabilities and illustrating their suitability for intrusion detection in the IMD ecosystem.

2.3.1 Machine Learning Approaches

Inefficiencies in conventional controls have driven research into machine learning strategies to address the zero-day threats targeting the Implantable Medical Devices infrastructure. These attacks lack existing defined controls and prevention measures, enabling hackers to exploit previously unknown vulnerabilities in the system (Kirimi, 2023). Healthcare organizations expose sensitive information resources without elaborate preventive measures to unauthorized access or tampering. As a result, machine learning strategies have been proposed to address these threats, providing an improved threat-detection model that mitigates zero-day incidents. Machine learning approaches require training detection models to enable them to identify novel attacks without updating their detection database (Kirimi, 2023). This approach provides a dynamic threat-management environment that enhances the detection engine's relevance as new threats emerge. For instance, leveraging machine learning approaches

enables the detection engine to identify and classify zero-day attacks and their variants. This detection supersedes conventional detection engines that require updating the signature database.

Various machine learning approaches have been proposed to detect and respond to malicious attacks targeting Implantable Medical Devices and the Medical Internet of Things. For instance, models such as random forests, gradient boosting, and multilayer perceptrons have been proposed as viable methods for threat detection (Kirimi, 2023). Random forests and gradient boosting use ensemble learning, where weak machine learning models are used to optimize the final detection model. This approach enables models to overcome the limitations of individual models, thereby providing enhanced detection capability (Kirimi, 2023). The multilayer perceptron is a classification neural network model that offers threat detection through interconnected nodes. This detection model provides a robust training environment, enhancing effective feature extraction for optimal threat identification (Kirimi, 2023). While these models improve performance over conventional threat-detection models, they have limitations when handling security incidents. For example, the random forests have an accuracy of 93.75%, gradient boosting had 94.26%, and the multilayer perceptron had 92.55% (Kirimi, 2023). These relatively low accuracy results highlight the need for better machine learning models to improve accuracy in the IMD ecosystems.

2.3.2 Overview of Deep Learning Models

The inefficiencies of conventional machine learning strategies drive research into deep learning models, aiming to optimize their performance in detecting novel threats. Deep learning is an advanced machine learning strategy that leverages artificial neural networks to solve complex problems (Lampe & Meng, 2023). These computational frameworks simulate the functioning of the human brain, aiming to replicate human decision-making when handling business problems. At the core of these algorithms are neurons that mimic the brain's neurological behavior. This approach simulates the brain's decision-making approaches, offering an efficient problem-solving model for complex tasks. A neuron comprises the inputs, outputs, weights, transfer, and activation functions, and bias (Y. Wang *et al.*, 2023). The inputs are the parameters fed into the

neuron, while the outputs are the values computed by the neuron. The weights are the values integrated into the inputs for computational purposes, while the bias is the shift value for the neuron. The transfer function aggregates the multiple inputs based on the neuron's configuration, while the activation function incorporates non-linearity to the aggregated inputs. These components are shown in Figure 2.4.

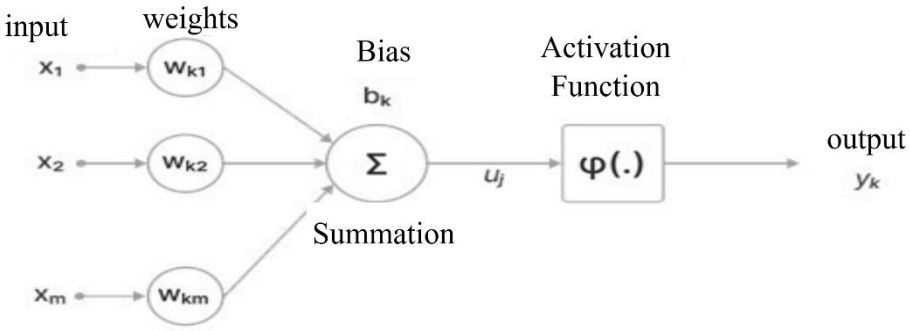


Figure 2.4: Architecture of a neuron (Y. Wang *et al.*, 2023)

The term 'deep' in deep learning refers to the presence of multiple hidden layers in the neural network architecture, which play a critical role in input transformation and feature extraction (Y. Wang *et al.*, 2023). Deep learning strategies aim to automatically learn hierarchical data representations, enabling them to extract features and patterns across multiple abstraction levels. This feature makes deep learning particularly powerful in intrusion detection and threat management tasks. The neural networks formed through this approach comprise three main layers: input, hidden, and output. The input layer captures raw inputs in the network (Y. Wang *et al.*, 2023). The layer initiates a forward pass for the network, effectively handling the provided dataset. The hidden layers are intermediate links between the input and output layers. These layers vary depending on the network's configuration and process the input using weights and activation functions. This processing enables the model to extract features from the dataset and learn from the target environment (Y. Wang *et al.*, 2023). The output layer provides the results, predicting or classifying the target dataset based on the deep learning problem. These layers are illustrated in Figure 2.5.

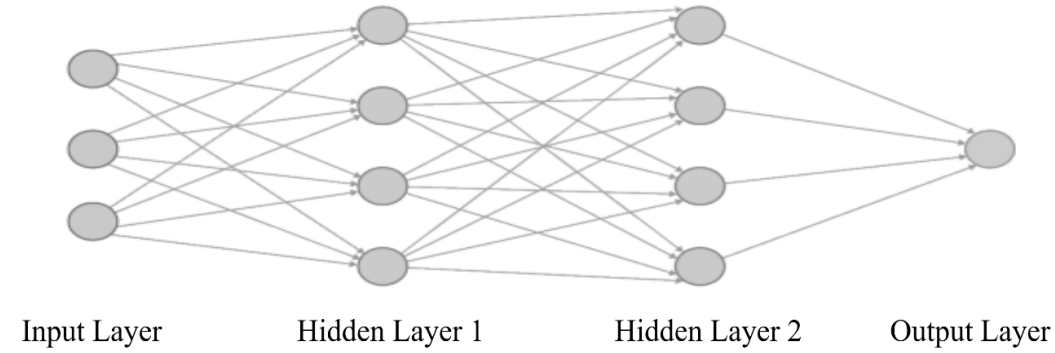


Figure 2.5: Architecture of a neural network (Y. Wang *et al.*, 2023)

Various neural networks are used in deep learning, offering diverse learning approaches. One type of neural network is the recurrent neural network (RNN), which is designed to handle sequential data. These neural networks efficiently handle time series analysis, natural language processing, and speech recognition by incorporating a feedback loop that captures temporal dependencies (Ferrag *et al.*, 2020). This loop allows neural networks to retain memory of previous inputs, providing an iterative training environment that addresses the limitations of conventional neural networks. Additionally, RNNs are flexible like their inputs, enabling them to handle varying input lengths. This phenomenon makes them adaptable to diverse applications (Ferrag *et al.*, 2020). A typical RNN adopts the conventional neural network architecture. However, it integrates hidden states that serve as memory, retaining information from the previous time steps. Weights in these hidden states are optimized based on the entire sequence and temporal data features, providing a varying performance environment (Ferrag *et al.*, 2020). This training aspect, known as backpropagation through time, enhances the overall performance of RNNs on sequential datasets. This feedback mechanism in the hidden states is shown in Figure 2.6.

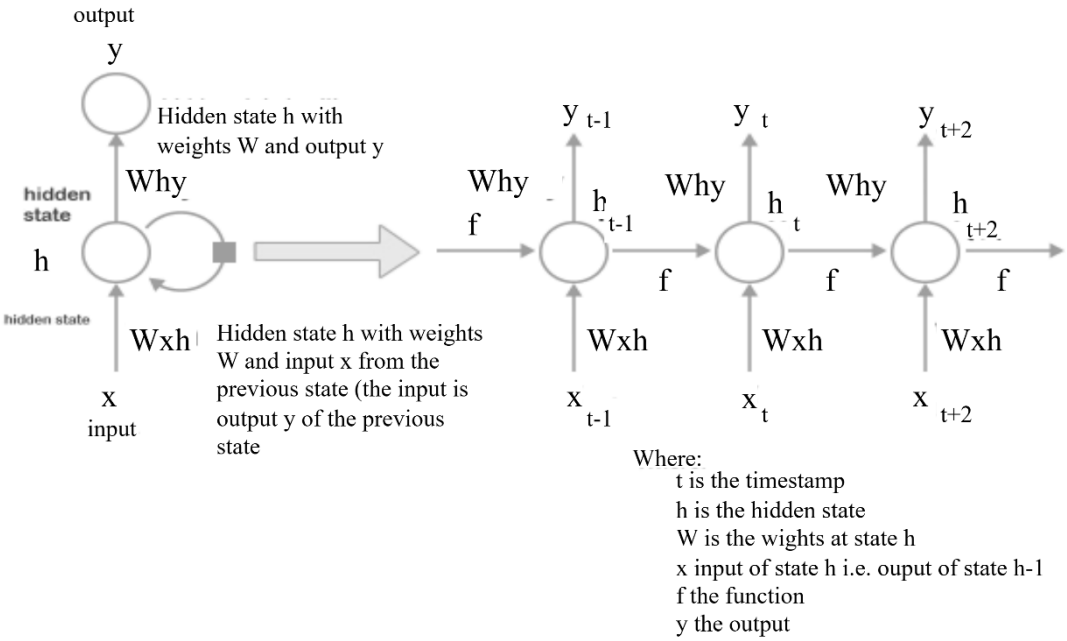


Figure 2.6: Feedback mechanism in RNN's hidden states (Ferrag *et al.*, 2020)

Another type is long short-term memory, an improved recurrent neural network that resolves memory limitations. This neural network incorporates memory cells into the feedback mechanism, resolving the short-term memory bottlenecks of conventional RNNs (Jiang *et al.*, 2020). The memory cells store temporal information throughout processing, enabling the model to optimize its performance based on prior information. Additionally, it integrates gating mechanisms that offer a robust decision-making model for selecting information to retain during the transformation. This decision is based on the binary analysis of the input provided by the preceding hidden states (Jiang *et al.*, 2020). The LSTM's architecture incorporates four additional components in the hidden states. The first component is the cell state, which provides the network with long-term memory functionality. The second component is the forget gate, which evaluates the current timestamp input and the previous cell state to determine which information to forget from the network (Jiang *et al.*, 2020). The weighted input matrices and the bias function influence this decision-making process. The third feature is the input gate, which determines which data to incorporate into the cell state. The last component is the output fate, which extracts meaningful information from the cell state to provide network output (Jiang *et al.*, 2020). The architecture of these additional components is illustrated in Figure 2.7.

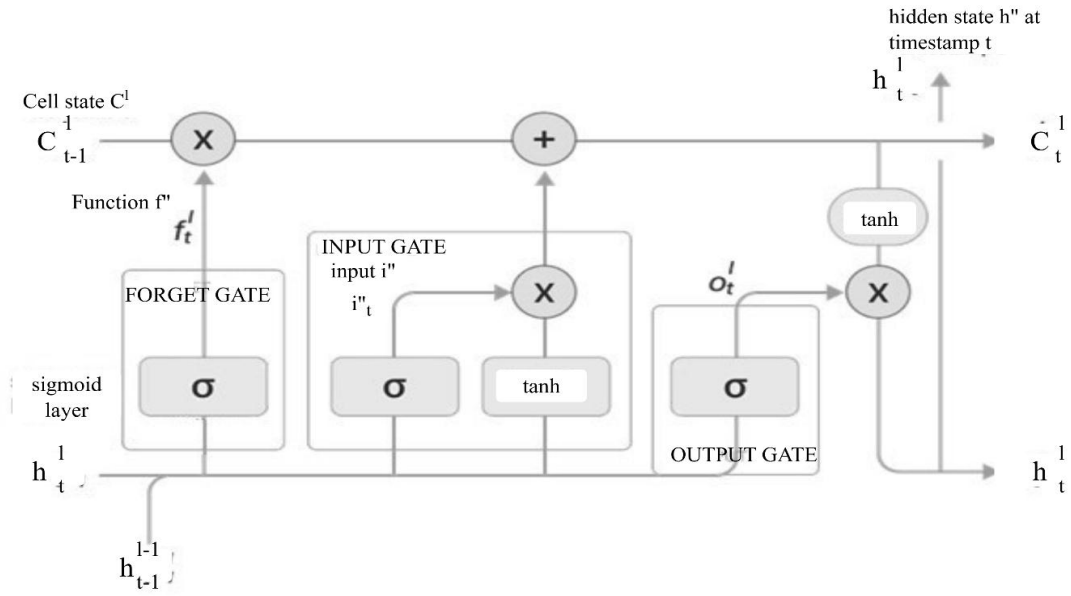


Figure 2.7: Memory architecture of LSTM (Jiang *et al.*, 2020)

Further, there are convolutional neural networks. Convolutional neural networks (CNNs) are neural networks designed to process and analyze structured grid-like data (Krichen, 2023). This phenomenon implies that these neural networks effectively handle two-dimensional datasets, enabling robust image analysis. Convolutional neural networks incorporate local receptive fields and shared weights that capture spatial feature hierarchies (Krichen, 2023). This feature allows the networks to learn hierarchical features automatically within the input grid. These neural networks differ from RNNs in that they include pooling, convolutional, and fully connected layers. The pooling layers are intended to reduce the spatial dimensions within the dataset, preserving essential features and improving computational efficiency for the network (Z. Li *et al.*, 2022). The convolutional layer filters the input dataset, extracting features to enable the network to recognize patterns on different scales. Nonlinear activation functions are incorporated into these networks to provide linearity, allowing them to model complex relationships in datasets. The fully connected layer leverages the convolutional layers' output to predict the image's class based on the extracted features (Z. Li *et al.*, 2022). The efficiency of this layer depends on the neurons' interconnectivity, which enables them to aggregate the extracted features. Effectively coordinating these layers empowers the network to transform two-dimensional data into

robust linear classifications. Figure 2.8 illustrates the architecture of classical convolutional neural networks.

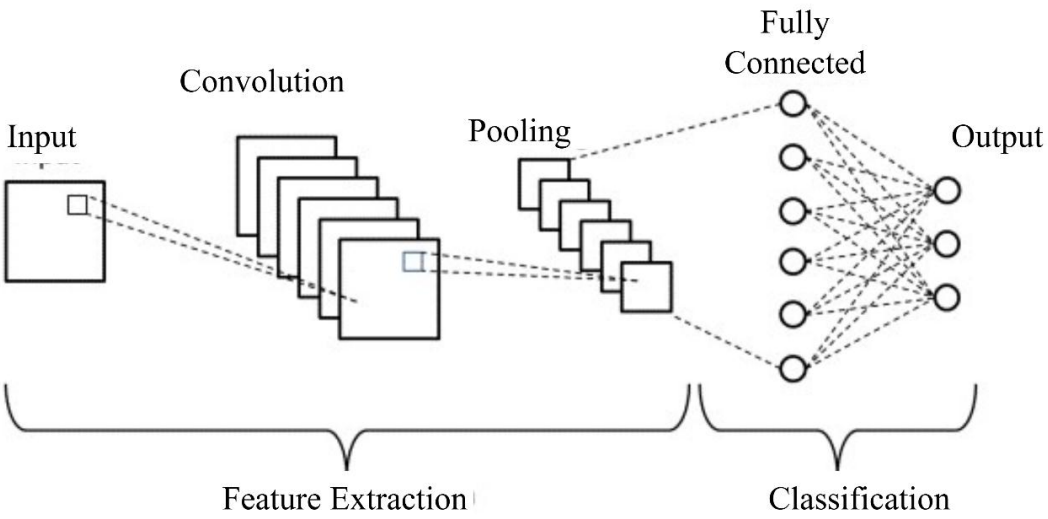


Figure 2.8: Architecture of convolutional neural network (Balaji, 2020)

Another deep learning model is the autoencoder, which reduces input into a latent-space representation (Nehra *et al.*, 2021). The output is then reconstructed using this representation scheme, providing an efficient input-reduction model while optimizing the algorithm's output. Autoencoders leverage supervised learning, a machine-learning strategy where the dataset is labeled to enable the model to learn from data (Nehra *et al.*, 2021). This deep learning model incorporates an encoder and a decoder, thereby enhancing its performance. The encoder maps the input to a lower-dimensional representation, thereby reducing dimensionality (Basati & Faghih, 2023). This layer captures the key features from the input, providing a robust analysis environment for the network. The decoder reconstructs input data from the compressed representation, generating output that closely resembles the original output (Basati & Faghih, 2023). A bottleneck layer separates the encoder and decoder. This layer further reduces the dimensionality of the input data and extracts essential patterns. Additionally, it regularizes the model, preventing overfitting and improving its overall performance. This architectural approach offers an efficient training environment for the model. Figure 2.9 illustrates this architecture of the autoencoders.

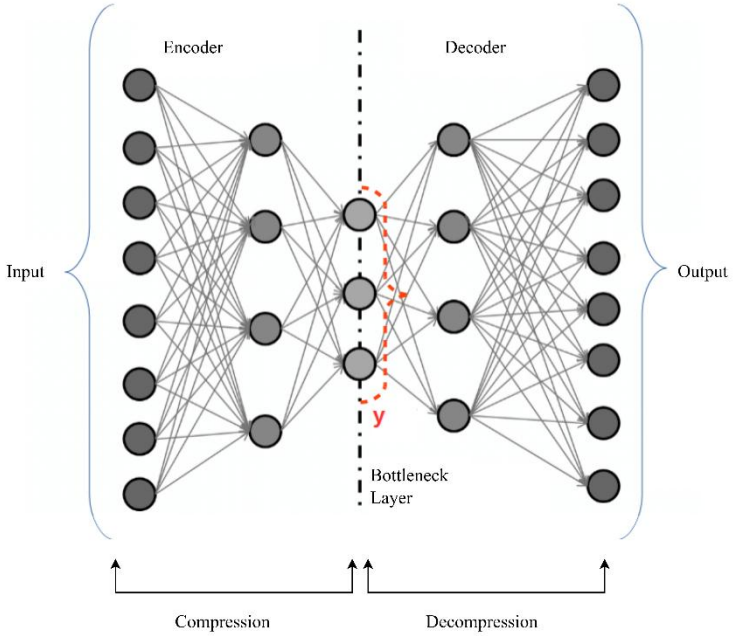


Figure 2.9: Architecture of the Autoencoders (Basati & Faghih, 2023)

Similarly, deep autoencoders have been extensively reviewed as deep learning models. Deep autoencoders extend the capabilities of conventional autoencoders by introducing multiple hidden layers in both the encoder and decoder. The hidden layers introduced in this design approach are the deep belief networks with shallow layers (Cunningham *et al.*, 2020). The layers resulting from this design approach are more complex than those in conventional autoencoders. The increased depth allows these models to capture more complex hierarchical representations from the input dataset (Cunningham *et al.*, 2020). The outcome is an increased ability of the model to learn abstract representations, optimizing its performance in handling hierarchical data. Deep autoencoders integrate activation functions within the neural networks implemented in the encoder and decoder segments. A loss function is also incorporated into deep autoencoders to quantify the difference between the input and reconstructed output (Cunningham *et al.*, 2020). Figure 2.10 illustrates the typical positioning of hidden layers in a deep autoencoder.

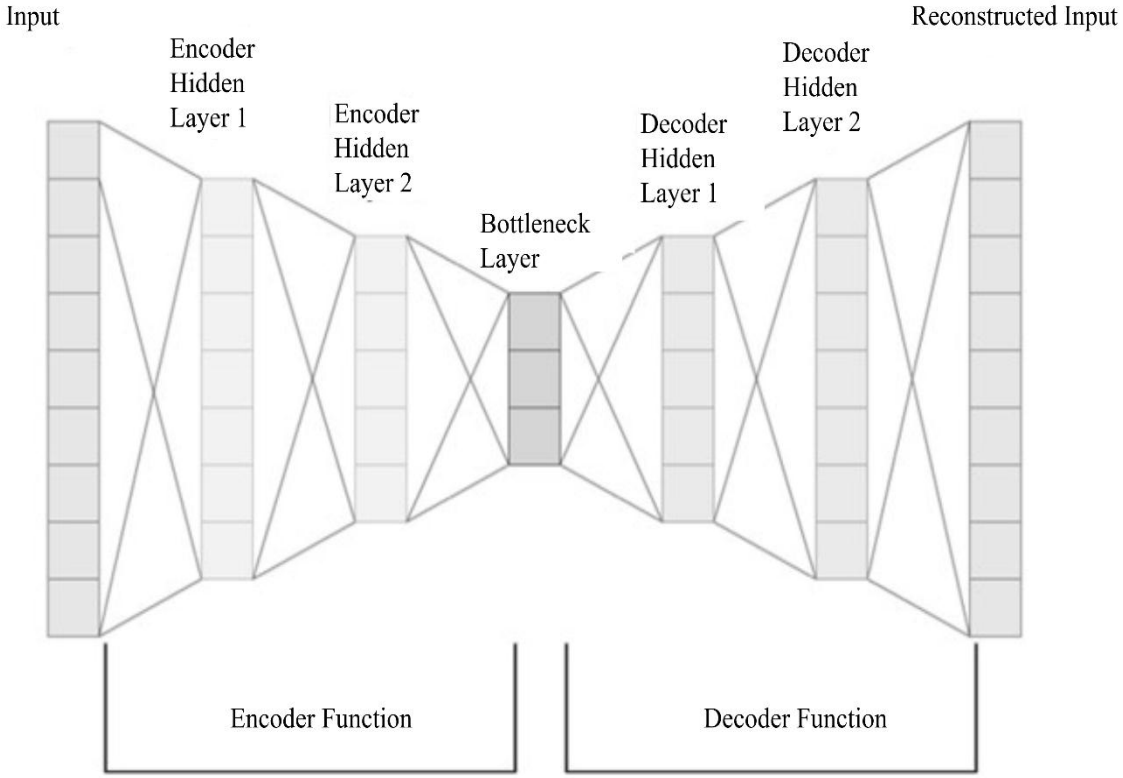


Figure 2.10: Architecture of a typical deep autoencoder (Cunningham *et al.*, 2020)

2.2.3 Deep Learning Models and Implantable Medical Devices Security

There is substantial research on diverse deep learning models for the cybersecurity of implantable medical devices. One such model is the autoencoder, which has been examined as a detection model for cybersecurity solutions. For instance, Basati & Faghih (2023) discuss the adoption of autoencoders as anomaly detection in network-based intrusion detection systems (NIDS). This review identifies the model's adequate performance in handling novel threats (Basati & Faghih, 2023). In other studies, autoencoders are leveraged for dimensionality reduction in intrusion detection systems (IDS), enabling detection models to reduce the dimensionality of input data while retaining features (Basati & Faghih, 2023). A separate study by Ortega-Fernandez *et al.* (2023) reviews the adoption of autoencoders as a detection model for NIDS. The researchers note that autoencoders provide a better performance environment than conventional supervised learning models (Ortega-Fernandez *et al.*, 2023). Siddiqui and Boukerche (2021) also examine the adoption of autoencoders in intrusion detection

systems. They note that leveraging autoencoders yields a robust threat-detection model. However, these neural network models require extensive computational resources, which are infeasible in IoT ecosystems (Siddiqui & Boukerche, 2021). They propose a strategy to reduce ensemble complexity while maintaining the detection model's efficiency. While such an approach seems promising for the IMD ecosystem, the researchers fail to provide a performance assessment based on an IMD-specific dataset. Such an assessment would enable healthcare service providers to determine the viability of autoencoders as intrusion detection frameworks for the IMD ecosystem.

Long short-term memory and convolutional neural networks have also been examined as potential solutions for intrusion detection systems in Implantable Medical Device ecosystems. For instance, Lampe & Meng (2023) discuss the adoption of Long Short-Term Memory and Convolutional Neural Networks in intrusion detection systems for IoT ecosystems. They note that CNN architectures comprise multiple layers of neural networks to solve complex problems (Lampe & Meng, 2023). Roy and Cheung (2018) have examined the adoption of LSs for detecting backdoors, denial-of-service attacks, worms, and reconnaissance attacks in IoT ecosystems. The researchers note that this neural network model provides 95% accuracy based on the dataset (Roy & Cheung, 2018). A separate study by Azizjon *et al.* (2020) examines the use of CNNs in an IDS environment. The researchers note that this architecture outperforms other machine learning models, such as random forests and support vector machines, achieving a detection accuracy of 90.91% (Azizjon *et al.*, 2020). These promising results from the CNN and LSTM provide a robust environment for extracting meaningful features, thereby improving the performance of the proposed autoencoder model. However, the studies conducted by these researchers fail to leverage the IMD-specific datasets, posing significant challenges in implementing the proposed solutions. For instance, using IMD-specific datasets for training and evaluation may yield different results, leading to distinct implementation considerations for the LSTM and CNN models.

Researchers have also examined hybrid deep autoencoders as potential intrusion detection solutions, aiming to integrate features from multiple models. One such hybrid model is proposed by Isa and Mhamdi (2022), which incorporates random forests into

a deep autoencoder. This proposed model is intended for software-defined networking and has 98% accuracy and precision (Isa & Mhamdi, 2022). A separate study by Moussa and Alazzawi (2021) integrates long short-term memory with an autoencoder to create a hybrid detection model for connected and autonomous vehicles. This hybrid intrusion detection model had an accuracy of 98.7% (Moussa & Alazzawi, 2021). While the findings from these hybrid models provide promising results for intrusion detection systems by addressing the limitations of the respective models, they are intended for non-healthcare infrastructure. This phenomenon reduces their adoption in the IMD ecosystem, as they fail to provide functionalities expected in the healthcare information infrastructure, such as meeting speed and accuracy targets of over 99.50%. As a result, there is a need for an IMD-oriented intrusion detection solution that addresses the unique requirements of the healthcare information infrastructure, such as high accuracy and low latency, to ensure patient safety and optimal service delivery.

2.3.4 Research Gaps

While studies have effectively identified alternative security solutions for the Implantable Medical Devices ecosystem, the proposed solutions have fundamental flaws. First, the deep learning approaches proposed in the studies exhibit overfitting. For instance, the Long Short-Term Memory and autoencoder models recommended in the reviewed studies suffer from overfitting issues. This training setback compromises these solutions' ability to handle novel datasets effectively. Secondly, these solutions fail to comprehensively detect and respond to the environment. Cybersecurity operations in the healthcare sector require timely responses to security incidents. This approach reduces the impact of attacks and averts their devastating effects on healthcare systems. The inability of the proposed security models to provide these functionalities compromises their efficiency in achieving these security goals. Thirdly, these security solutions do not leverage IMD and M-IoT datasets. Instead, they use general IoT datasets, which compromise their ability to address healthcare-specific security needs adequately. This phenomenon compromises the efficiency of existing hybrid intrusion detection models, as they fail to address emerging healthcare security needs. Lastly, different detection models have performance limitations. This phenomenon is attributed to weaknesses in individual models recommended in the studies.

There is a need for a hybrid solution that draws on features from the proposed models while addressing the limitations of the individual systems. Such a solution enhances the detection model's accuracy by leveraging the strengths of individual models (de Souza *et al.*, 2020). This feature optimizes the performance of the developed solution by resolving accuracy limitations for the individual models. For instance, one algorithm may perform best on one dataset and fail on another. Integrating multiple models ensures that the best-performing model across environments is selected for the intrusion detection system, thereby improving overall accuracy. Further, the hybrid approach enhances the solution's flexibility and adaptability, enabling it to handle novel scenarios (Naseer, 2020) effectively. This phenomenon allows for the developed solution to handle evolving IMD security needs, thereby improving performance over time. Thus, a hybrid solution offers the best approach to handling IMD security challenges by integrating features from multiple detection algorithms.

2.3.5 Proposed Solution

The proposed solution consisted of three main components that, collectively, provide the detection capabilities of the Intrusion Detection System. These components were the encoder, decoder, and the bottleneck layer. The encoder was responsible for transforming the input data into a compressed representation suitable for further analysis. It consisted of 8 layers of one-dimensional Convolutional Neural Networks, with the Leaky Rectified Linear Unit (Leaky ReLU) as the activation function. Leaky ReLU activation is commonly chosen for its simplicity and effectiveness in capturing nonlinearities within the data. Further, it was considered for its fast inference time, speeding up the detection capabilities of the developed model. The Encoder's activation function was sigmoid, facilitating the mapping of input data to a compressed representation in the range [0, 1]. This compressed representation retained essential information while reducing data dimensionality, enabling efficient processing and anomaly detection. Additionally, the optimizer used to train the Encoder was Nadam. Nadam combines the benefits of Nesterov accelerated gradient descent and Adam optimization, offering adaptive learning rates and momentum. This optimization function improved convergence and efficiency during training, enabling the autoencoder model to learn effectively.

The decoder was responsible for reconstructing the original input data from the compressed representation generated by the Encoder. Like the Encoder, the Decoder comprised 8 layers of 1D Convolutional Neural Networks with Rectified Linear Unit activation functions, enabling it to learn complex mappings between the compressed representation and the original input data. ReLU activation functions ensure that the Decoder captured intricate patterns and features during reconstruction. Like the Encoder, the decoder's activation function was sigmoid, facilitating the generation of output data in the [0, 1] range. The bottleneck layer comprises the long short-term memory and is pivotal for capturing temporal dependencies and sequential patterns in the data. Unlike the encoder and decoder that leveraged CNNs, this layer used recurrent neural networks (RNNs). The component consisted of 1 Neural Network (NN) layer and used the ELU activation function to introduce nonlinearity and capture complex temporal patterns. ELU's ability to handle vanishing gradients and accelerate convergence made it well-suited to the LSTM architecture. Further, it resolved self-normalization problems in LSTM, offering a fast and accurate prediction environment. LSTM layers excel in modeling sequences by maintaining long-term dependencies, making them highly effective for time-series data such as network traffic in an IDS scenario.

The encoder, bottleneck layer, and decoder were integrated during the training, providing a comprehensive IDS model for the cloud-based Implantable Medical Devices ecosystem. This integrated training approach leveraged the Nesterov-accelerated Adaptive Moment Estimation (Adam) optimization algorithm with mean squared error as the loss function. The approach ensured that all components of the deep autoencoder effectively captured and reconstructed the essential features of the network traffic data, thereby improving anomaly detection performance. The Nadam optimizer was used during training to update the autoencoder model's parameters. Additionally, mean squared error (MSE) was used as the loss function during training. MSE quantified the reconstruction error by measuring the discrepancy between the original input data and its reconstructed output. Minimizing the MSE loss enabled the autoencoder to generate reconstructions that closely match the original input data, thereby enhancing the accuracy and effectiveness of the IDS model in the cloud-based

IDM ecosystem. The SELU is used throughout the model due to its robust overall performance.

Integrating convolutional neural networks, long short-term memory, and autoencoders offered a comprehensive deep learning environment for intrusion detection in the Implantable Medical Devices infrastructure. The extensive use of CNNs in the hidden encoder and decoder layers provided a robust compression and decompression framework that effectively captures spatial information from the datasets. This feature enhanced the accuracy of the developed model, enabling it to identify threats in deployment scenarios efficiently. Further, this hybrid security solution sought to address the limitations of individual models by leveraging ensemble learning techniques in deep learning. The trained model was then tested against the conventional CNN, LSTM, and LSTM-based autoencoder to assess its performance. This testing approach provided an expanded analysis environment for the developed threat detection model, ensuring it meets the design goals and demonstrates substantial performance against the conventional models. The performance setbacks identified in the model were optimized through a series of deep learning strategies to develop the ideal detection environment for the IMD ecosystem. This solution's architecture is illustrated in Figure 2.11.

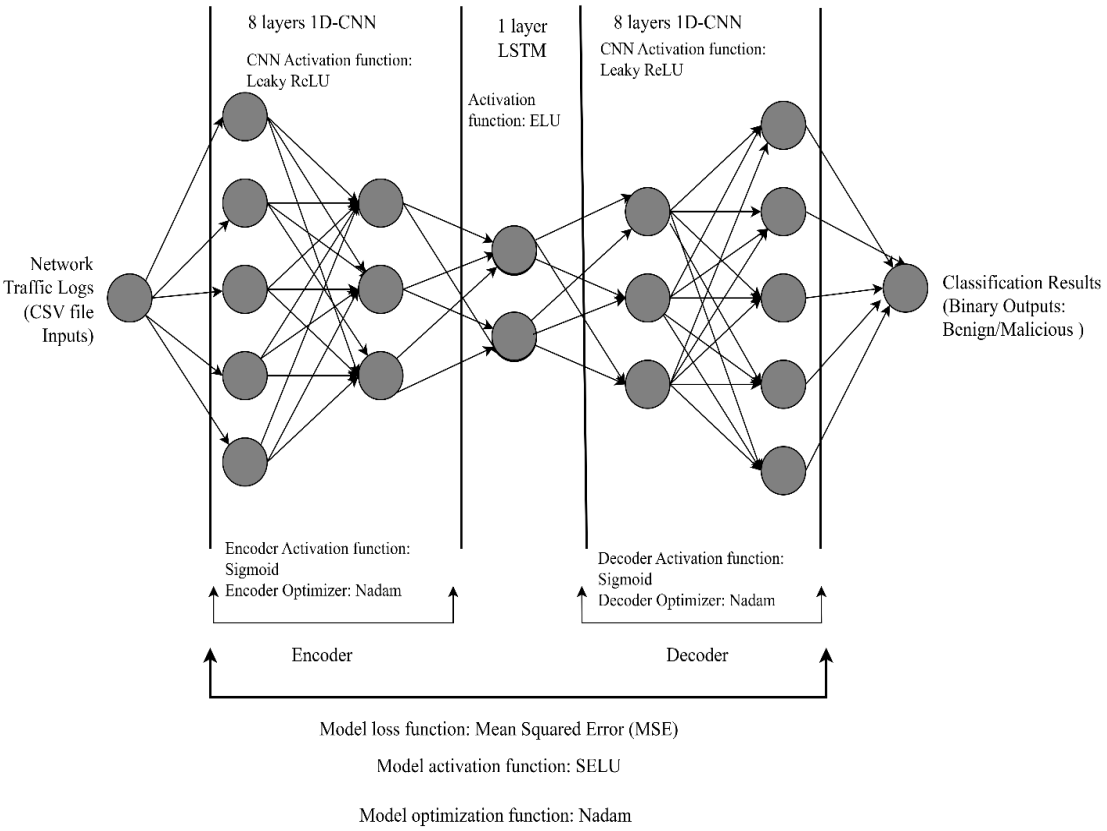


Figure 2.11: Proposed architecture for a deep autoencoder

2.4 Conceptual Architecture

The proposed Intrusion Detection System (IDS) architecture for the Implantable Medical Devices ecosystem was designed to provide a robust detection model for IMD threats. The core of this architecture consisted of the detection engine and the user interface. The user interface was a web-based environment that facilitated user interaction and visualization, allowing users to engage with the system effectively. The outputs for this component were graphics showing the reported threats, their risk scores, and potential mitigations. The intrusion detection engine comprised the inputs, outputs, mediating, and moderating variables. The model's inputs were network traffic logs, which provided a rich, dynamic source of captured communication in the Implantable Medical Devices ecosystem. Since IMD solutions increasingly use encryption frameworks such as Transport Layer Security (TLS) to safeguard patient safety and ensure the confidentiality of communication, it becomes challenging to analyze application-layer traffic due to data obfuscation (Zhou *et al.*, 2024). As a result, there is a need to capture raw traffic at the transport layer, providing raw IP-based packets

for analysis (El-Sherif *et al.*, 2024). The use of CNN and LSTM components in the model provided a robust framework for analyzing this traffic, effectively overcoming the encryption limitations that would have been encountered in application-layer data. Thus, the model's architecture, which used a deep autoencoder with CNN and LSTM features, was effective in handling encrypted traffic.

The optimization and activation functions served as moderating variables, influencing the detection model's performance. The optimization function was adjusted based on different optimization functions available for autoencoders. In contrast, the activation function is applied to the individual neural networks in the model's encoder, bottleneck, and decoder. For instance, selecting the Nadam optimization function and the ELU, SELU, and ReLU activation functions in the model enhances convergence speed, reducing inference time. Further, they improve the model's stability by enhancing its accuracy, precision, and recall, thereby significantly influencing its performance on these metrics. Selecting alternative activation functions in the model would substantially degrade its performance, resulting in poor results that fail to meet the deployment requirements for the developed IDS solution. The mediating variable was the number of classification problems the model handled. These variables shaped the model's performance by influencing its accuracy and efficiency. Further, it provided the training dataset that enabled the model to perform as expected. A significantly low or imbalanced number of classification samples compromised the model's efficiency, leading to skewed results, overfitting, or underfitting. For instance, using a dataset with a high number of benign cases would prevent the resulting model from effectively classifying malicious cases. Thus, these moderating and mediating variables played a critical role in enhancing model stability, accuracy, and suitability for real-time deployment. The relationship between these components is illustrated in the conceptual diagram shown in Figure 2.12.

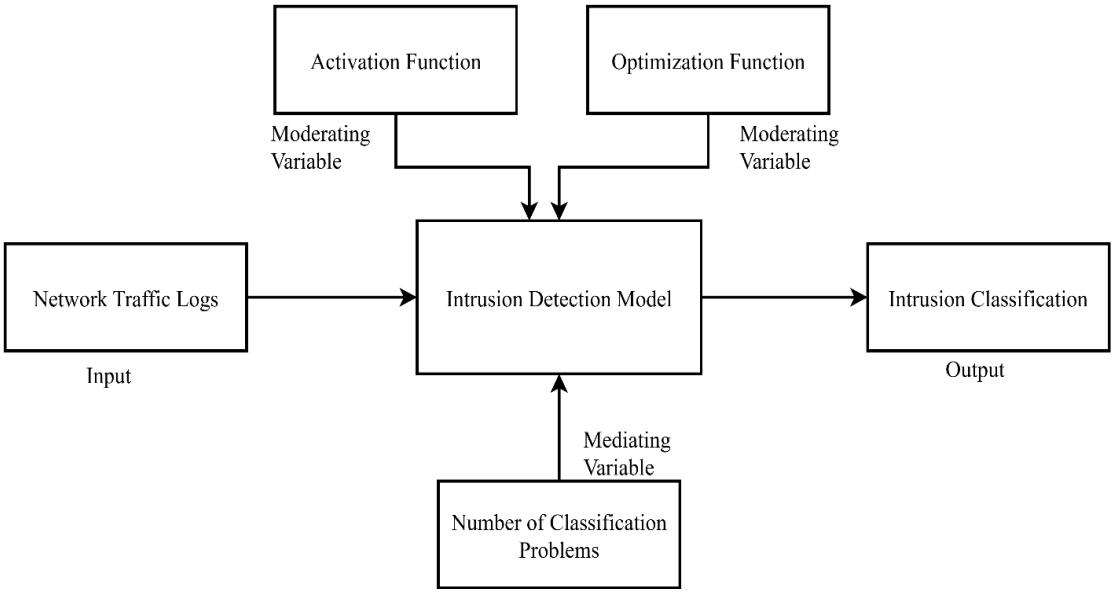


Figure 2.12: Conceptual architecture of intrusion detection engine

The output from the detection engine included classification results and risk scores for the identified threats. The classification results were used to determine whether the traffic log fed into the model was malicious or benign. The classification of these logs was evaluated based on precision, recall, accuracy, F1, and inference time as presented in section 2.2.6. The accuracy, recall, precision, and F1 scores examined the model's performance in handling classification requirements, indicating its ability to handle both existing and novel incidents. On the other hand, the inference time examined the model's ability to meet the near-real-time requirements for the clinical implementation. The model needed to effectively balance these scores to enhance adoption in the proposed environment. For instance, high performance in terms of accuracy, precision, recall, and F1 scores, coupled with high inference time, would prevent the adoption of such a model, as it would prove ineffective at handling threats in real time, despite the high accuracy scores. Low inference time with poor performance in accuracy, recall, precision, and F1 would prevent the model's adoption, as it would meet near-real-time requirements but fail to deliver the high accuracy expected for clinical systems. Thus, the effective balancing of these factors in the model provided a robust detection framework for clinical settings involving IMDs.

The risks identified by the detection engine were determined by their severity levels. These severity levels are based on Table 2.2 and Figure 2.4. This approach provided dynamic risk management, as the risk classification depends on the overall threats identified in a minute. For instance, an increase in the attacks targeting implantable medical devices necessitates a proportionate security response, while a reduction in the attacks reduces the resources allocated. However, this resource reduction does not advocate lax security management, as it ensures that appropriate measures are put in place to address the identified threats. The security personnel configured in the system were notified via short messaging service (SMS) about these risks for medium, high, and critical incidents. This notification ensured the security administrator promptly responded to the incident and efficiently orchestrated the remediation of the compromised assets. Administrators were notified via email about threats with a low or negligible risk score. This approach ensured that users were informed of the breaches, enabling relevant measures to be implemented.

These results were stored in the alert registry, providing a historical record of security incidents. The alert registry was integrated into healthcare workflows and alarm management, enabling effective deployment of the IDS for monitoring IMDs. For instance, the system administrators were notified of the threats detected by the IDS via email and Short Messaging Service. The emails were intended for non-critical incidents in which the threats did not reach the critical or high-risk level under the risk classification mechanism. When threats reached this level, system administrators were notified via email and SMS. This expanded notification mechanism ensured that device administrators were aware of the incident's criticality and enabled prompt responses to reported incidents. The storage of threats ensured that system users could access historical information relating to the attacks, assess the measures implemented, and recommend approaches to safeguard the infrastructure. The results were visualized using an interactive dashboard. This presentation approach enabled users to quickly locate preferred information regarding the cybersecurity landscape and appropriately respond to security incidents identified by the detection engine. For instance, the dashboards retrieved attack history, enabling users to monitor deployed devices effectively. This monitoring improved healthcare decision-making by integrating IDS

functionality into existing security management workflows. Further, summaries of major threats and the most targeted devices were provided, allowing management to implement appropriate risk management strategies. The provision of these summaries and their retention by the system enabled system administrators to monitor the IDS's progress toward meeting healthcare security goals. These components are illustrated in Figure 2.13.

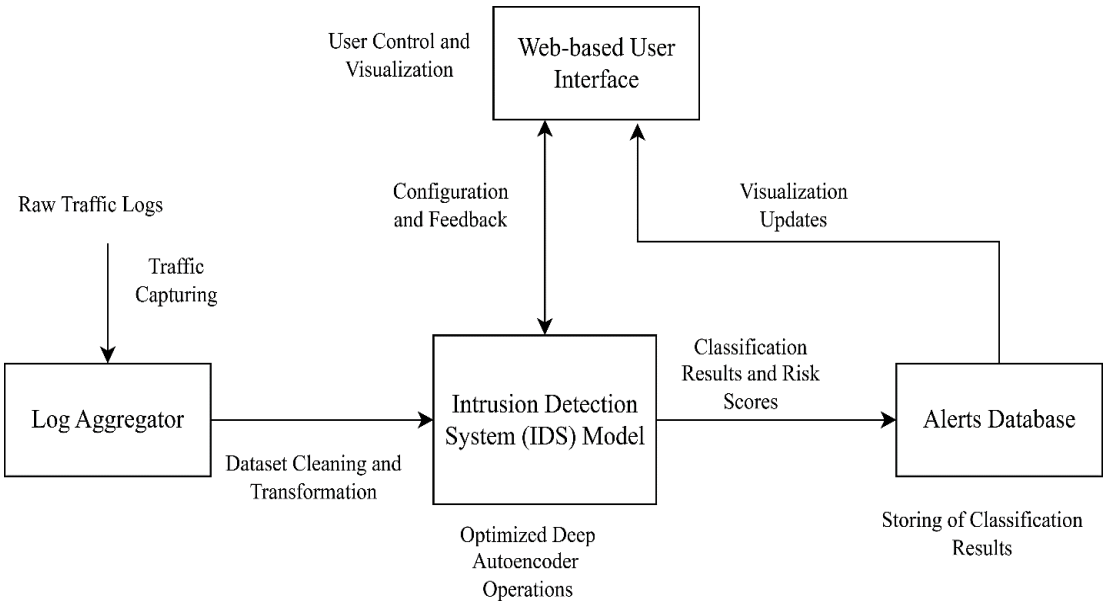


Figure 2.13: Proposed IDS architecture

2.5 Summary of Existing Implantable Medical Devices Cybersecurity Solutions

Table 2.4 summarizes the features of existing security solutions, highlighting their limitations and contributions to the current study.

Table 2.5: Summary of IMD Cybersecurity Solutions

Proposed Security Feature	Author and Year	Findings	Knowledge Gap	Focus of the Current Study	Research Outcomes
Hybrid security solutions	de Souza *et al.* (2020)	Integration of diverse security features enhances data protection and improves performance.	Limited emphasis on responsiveness to security breaches in commercial security strategies for IMD infrastructure.	Evaluate the efficiency of hybrid security solutions and propose strategies to enhance responsiveness to security breaches in IMD ecosystems.	Integrated MLP as a classification head into CNN-LSTM-CNN autoencoder, incorporated Nadam optimization, and leveraged regularization and batch normalization to improve responsiveness to security threats. It strengthened the IDS's ability to identify threats by achieving a low false-positive rate.
Cloud computing frameworks	Thamilarasu *et al.* (2020)	Cloud computing enables expanded security functions for IMDs, resolving resource limitations.	Emphasis on resilience rather than responsiveness to security breaches.	Evaluating the effectiveness of cloud computing frameworks in managing resources for elaborate security controls in IMD ecosystems.	The Google Colab environment, a cloud computing framework, was effectively used to train and evaluate the model, demonstrating the importance of such frameworks in strengthening IMD security. Storage for the cloud was obtained via APIs, showing the promise of integrating cloud infrastructure to handle security threats for the IMD devices.
Secure communication protocols	Bhuiyan *et al.* (2021)	Proposed protocols facilitate robust device authentication and validation, enhancing safety.	Limited focus on IMD and M-IoT datasets in evaluating security solutions.	Assessing the effectiveness of secure communication protocols in enhancing communication safety in Medical IoT ecosystems.	Used IMD (ICU dataset) and M-IoT (WUSTL) datasets to develop the IDS for the IMD ecosystem. The developed IDS engine enhanced communication security by identifying and flagging malicious incidents with 100% accuracy in the ICU dataset and 79.32% in the WUSTL dataset. The Edge IIoT dataset achieved an accuracy of 96.87%.

Machine learning approaches	Kirimi (2023)	Machine learning approaches offer dynamic threat management, improving threat detection.	Limited focus on IMD and M-IoT datasets in evaluating security solutions.	Comparing the performance of deep autoencoders with other machine learning models for intrusion detection in IMD and M-IoT environments.	Random Forests, Multilayer Perceptron, and Gradient Boosting showed better performance than standard and Nadam-optimized models. MLP was integrated into the model as a classification to provide similar classification accuracy.
Long Short-Term Memory and Convolutional Neural Networks	Lampe & Meng (2023)	CNN and LSTM architectures provide robust features for intrusion detection in IoT ecosystems.	Resource limitations of the IoT ecosystems compromise the implementation of robust CNN and LSTM solutions. Also, the lack of IMD-specific datasets for model training leads to overfitting.	Investigating the potential of CNN and LSTM models in enhancing threat detection for IMD and M-IoT security.	The CNNs were used to create the encoder and decoder segments of the model, while the LSTM was used to develop the bottleneck layer. The developed model achieved 100% accuracy, recall, precision, and specificity on the ICU dataset.
Autoencoders	Basati & Faghih (2023)	Autoencoders demonstrate effectiveness for anomaly detection and dimensionality reduction in NIDS.	Overfitting issues due to a lack of IMD-specific datasets.	Examining the efficiency of autoencoders in detecting anomalies and reducing dimensionality in intrusion detection systems for IMDs.	The hybrid model based on an autoencoder architecture surpassed the scores achieved in the individual CNN and LSTM models.
Hybrid Deep Autoencoders	Isa & Mhamdi (2022); Moussa & Alazzawi (2021)	Hybrid deep autoencoders achieve high accuracy and precision in deployment scenarios.	No focus on the IMD and M-IoT dataset for the IDS	Leveraging IMD and M-IoT datasets to train the hybrid deep autoencoder model	Used the ICU and WUSTL datasets, which were IMD and M-IoT datasets

CHAPTER THREE
METHODOLOGY

3.1 Study Site

The primary study site selected for the research was Chuka University. This situation implies that all research operations, including data collection and analysis, were coordinated through Chuka University. This site was selected in accordance with the program's guidelines. As a result, institutional resources such as the library, computer laboratory, and network connectivity were extensively leveraged in the study. While the designed solution was intended for the healthcare environment, a secondary site focusing on such settings was not considered. Instead, institutional resources were used to simulate and assess the performance of the deployed model. This approach provided a comprehensive assessment environment for the proposed solution within the primary study site.

3.2 Research Design

The research adopted a pragmatic paradigm to develop a hybrid intrusion detection system for Implantable Medical Devices (IMDs). The pragmatic research paradigm emphasizes practicality, utility, and real-world application, making it well-suited to the development of an intrusion detection system (Kankam, 2019). It emphasizes the importance of addressing practical problems and generating knowledge directly applicable to improving practice or decision-making. In the IMDs scenario, there is a need for practical solutions that enable healthcare institutions to identify and resolve intrusions targeting the deployed devices. The pragmatic paradigm provided the relevant research environment to develop a practical solution that effectively addresses the sector's security needs. The interdisciplinary nature of the pragmatic research paradigm enabled it to provide a comprehensive solution to research issues in the healthcare sector (Kelly & Cordeiro, 2020). Healthcare systems require diverse teams, including medical professionals, nurses, care providers, patients, and IT personnel. The pragmatic paradigm provided a robust framework for integrating such teams throughout the research process, leading to the development of a comprehensive cybersecurity solution to address intrusion threats in IMD ecosystems.

The research recognized the importance of empirical evidence from experimentation and theoretical insights from the literature review in advancing cybersecurity domain knowledge. Integrating findings from experimentation and literature review provided a holistic understanding of the cybersecurity threats faced by IMDs and enabled the development of effective solutions to mitigate these risks. Additionally, the study considered cybersecurity research a multifaceted and dynamic environment, acknowledging the evolving nature of threats and the contexts within IMD ecosystems. This consideration recognized the complexity of the IMD ecosystem and the interconnectedness of various actors and systems. For instance, the research acknowledged that intrusion threats targeting the IMD ecosystem change rapidly, raising the need for adaptive security solutions that enhance the service provider's ability to navigate this dynamic landscape. This consideration led to the adoption of deep learning strategies for developing an intrusion detection engine (Basati & Faghih, 2023). Thus, the pragmatic research paradigm provided a robust framework for assessing the security threats facing the IMD ecosystem and for developing the threat detection system.

The research used an experimental approach, where the deep learning models for the intrusion detection system were designed, developed, and tested. This strategy offered a hands-on assessment of the proposed solution, enhancing a review of its features and functionalities to ensure it meets the design goals. The inputs for the experiment were the network traffic logs obtained from the Implantable Medical Devices. These logs were formatted as comma-separated values (CSV) and fed into the intrusion detection model. This model was developed as a deep autoencoder that leverages features from long short-term memory (LSTM) and convolutional neural networks (CNNs). The LSTM network captured temporal dependencies in the traffic, while CNNs identified spatial patterns in the data. Collectively, these deep learning approaches identified evolving security incidents, including denial-of-service attacks, data exfiltration, and packet manipulation. The output from the developed model was the intrusion classifications. These classifications were binary: benign and malware. The benign category included harmless requests to the devices, while the malware category

consisted of harmful requests. The harmful requests were intended to compromise the devices' normal functioning, while the harmless ones represented typical functioning.

The risk scoring for the classification output was based on the qualitative framework presented. This approach involved classifying threats into five categories critical, high, medium, low, and negligible and assigning priority levels in ascending order (Chandra *et al.*, 2022). The high-priority incidents were the critical ones, while the lowest-priority incidents were negligible. The model notified the IDM administrator using emails for the negligible and low-risk levels. In contrast, emails and short messaging services (SMS) were used for medium-, high-, and critical-risk incidents. These notifications included details regarding the nature of the attack and possible remediation measures. This notification approach ensured effective resource utilization while enabling the administrators to understand the security posture. For instance, notifying users by email only for critical and high-risk incidents would prevent them from promptly addressing the issue, as they may not check their emails until later. On the other hand, notifying them via email and SMS for low-risk, negligible incidents would unnecessarily increase maintenance costs for the deployed model. Thus, this notification approach adequately balanced the costs and responsiveness for security incidents.

The datasets were split into benign and malware classes. The benign group comprised harmless requests for Implantable Medical Devices. These requests were from legitimate users seeking services from the devices. For instance, the healthcare facility may seek to access information on the device through legitimate means. Further, the IMD devices may transmit patient data to the controller, updating device status or providing information regarding the monitored organs. These legitimate requests were considered harmless if they complied with the confidentiality, integrity, and availability measures implemented in the IMD ecosystem. The malware group included illegitimate requests to the IMD controller. Such requests violated the confidentiality, integrity, or availability measures implemented in the IMD ecosystem. For example, a malicious user may tamper with the packets being transmitted to the controller from the healthcare facility, compromising data integrity. The samples for the model training were selected from these groups to provide a balanced representation of the security threats. This

selection involved shuffling the captured datasets to prevent bias in the initial sample. After shuffling, stratified random sampling was used to select the target entries for the training. Using stratified random sampling further prevented potential bias in the dataset, enhancing the efficiency of the developed model (Zhao *et al.*, 2019). Thus, this dataset preparation approach was instrumental in training the intrusion detection model.

The research leveraged three datasets, offering a comprehensive model training and testing environment. The first dataset considered was the Institute of Electrical and Electronics Engineers (IEEE) IoT Healthcare Security dataset (Hussain *et al.*, 2021). The dataset comprised internal patient monitoring devices deployed in an Intensive Care Unit environment with 80,126 entries. This dataset was selected because it is adequate due to its extensive size. Further, it was reliable and valid, providing both benign and malicious cases. Also, it was relevant as it covers Healthcare IoT devices, encompassing Implantable Medical Devices. This aspect enabled the dataset to simulate an IMD environment, addressing IMD security considerations. The dataset was obtained in 2021, demonstrating its recency. These features made this dataset suitable for developing an intrusion detection system for IMD devices. The dataset considered in this study contained 30,000 samples. These samples were randomly selected from the dataset to prevent bias and enhance the model's generalization. The dataset was used for training and testing the model, providing an expanded comparative environment for convolutional neural networks, long short-term memory, and autoencoders.

The second dataset was the Washington University in St. Louis' Electronic Health Management System (EHMS) 2020, managed by Washington University in St. Louis (WUSTL, 2020). It offered an expanded data environment for healthcare IoT, covering 16,318, demonstrating its adequacy. This dataset was reliable and valid, as it included both benign and malicious cases, providing an extensive training environment. Further, it was relevant because it covers diverse electronic healthcare systems, including Implantable Medical Devices. This feature provided the IMD with a broad training and testing dataset that helps refine the insights obtained from the IoT Healthcare dataset. Additionally, the dataset provided a general perception of the Medical IoT environment, enhancing the assessment of the developed model and enabling its performance

evaluation on the broader healthcare information infrastructure. The dataset was obtained in 2020, providing relatively recent data for the models. The samples considered for this dataset's model training and testing were 15,000. These samples were randomly selected from the dataset to prevent bias and enhance generalization. Further, the sample was sufficiently large to ensure optimal representation of the total population. This dataset was used for training and evaluation, strengthening the comparative assessment of the intrusion detection engine.

The last dataset was the Edge Industrial IoT dataset, managed by the Kaggle platform (Ferrag *et al.*, 2023). The dataset had 2,219,201 cases, providing an expanded list of Industrial IoT (IIoT) malware and benign traffic. This large dataset was instrumental in verifying and validating the performance of the developed IMD detection system. This feature demonstrated the dataset's adequacy. Further, it provided both benign and malicious cases, demonstrating its reliability and validity in meeting the needs of intrusion detection systems Ferrag *et al.*, 2023). The dataset covered industrial IoT devices alongside broader IoT devices. This phenomenon enhanced the generalizability of the developed model, enabling its implementation across different IoT scenarios. For instance, the model's training and test performance on this dataset informed its generalizability across IoT scenarios. Further, it assessed the model's replicability in other non-medical IoT scenarios, thereby enhancing the generalizability of the findings. The dataset was obtained in 2023, demonstrating its recency. These features ensured that the developed model reflects the evolving threat landscape and attack strategies. The sample considered for the model was 100,000. This sample size was substantially larger than needed to represent the total population, providing a reasonable dataset for training and testing. The dataset was used for training and evaluation, providing an expanded assessment environment for the model.

The three datasets had different features and formats, which may affect model training and evaluation. This issue was addressed by selecting specific fields from the datasets for model training and evaluation. The top 10 features in each dataset were selected for modeling, providing an expanded environment for developing the intrusion detection engine. Three considerations were made during this selection. First, the fields in each

dataset were numerical. These fields were selected as they provide statistically significant information vital for analysis and modeling (Long *et al.*, 2019). Secondly, the selected fields were standardized by converting them to floating-point values. This conversion was intended to standardize all entries across datasets, enhancing consistency in model development and assessment. Lastly, feature engineering was leveraged to determine the specific features considered in each dataset. This approach ensures that there are no predetermined features in the datasets, preventing potential model skewing. Each dataset presents these features, leading to the model's transparency and assessment.

3.3 Data Collection

The datasets used in this research were obtained from public repositories. The approach involved downloading the datasets from these repositories for use in the model. For instance, data was downloaded from the Universal Resource Locators (URLs) linked to the datasets and loaded into the Google Colab environment, where it was used alongside the model that was developed. This data collection approach provides three benefits for the study. Firstly, public repositories offered expansive open-access data, providing researchers with diverse samples and scenarios to analyze. This abundance of data ensured that the study captured a comprehensive range of potential security threats, vulnerabilities, and patterns relevant to implantable medical devices. Secondly, public repositories host datasets that have been carefully curated and validated by experts in the field, ensuring high quality and reliability. By tapping into these repositories, researchers accessed datasets that have undergone rigorous scrutiny and validation, saving time and resources that would otherwise be spent on data collection and verification. Lastly, using publicly available datasets promoted transparency and reproducibility in research, as other researchers can access and verify the same data to replicate or validate the study's findings. This adherence to open science principles enhanced the credibility and robustness of the research outcomes.

Three datasets were considered for the research, providing a varied training environment for the proposed model. The first dataset obtained from IEEE offers a global perspective of IMD security threats (Hussain *et al.*, 2021). The insights drawn

from this dataset were instrumental in developing an IMD-specific intrusion detection model. The findings from this dataset were essential for assessing the relevance, applicability, and suitability of the developed model for handling real-time security needs in the IMD ecosystem. The second dataset was obtained from WUSTL, offering an academically centered dataset for IMD security modeling (WUSTL, 2020). The insights drawn from this dataset were instrumental in enhancing the generalization of the intrusion detection model in other Medical IoT devices. For example, the findings from this dataset would inform the implementation of the developed model in healthcare information infrastructure, facilitate its integration into the existing IDS solution, and determine potential uses of the model to optimize existing solutions in non-IMD ecosystems. The last dataset on the Kaggle platform comprises Industrial IoT intrusion data (Ferrag *et al.*, 2023). This dataset enhanced the generalization of the developed model in other IoT implementation scenarios. Further, it provided insights into whether the developed model could be applied to general IoT scenarios to identify threats and enhance responsiveness to security incidents.

3.4 Data Analysis

Data analysis provided the foundation for validating the proposed intrusion detection framework by incorporating data preprocessing, model development, optimization, and evaluation. The phase ensured that the data was cleaned, organized, formatted, and normalized before it was used to train the model. Further, it ensured the model is designed, trained, and evaluated based on accuracy, recall, precision, inference time, and F1 scores. Similarly, it ensured that the model was optimized using diverse optimization approaches and re-evaluated to assess its performance. The phase also provided the hardware and software specifications for the proposed model, defining environmental attributes that facilitated the development of the intrusion detection system.

3.4.1 Data Preprocessing

The purpose of data preprocessing was to refine the publicly available datasets to ensure they were suitable for the modeling environment used to develop the intrusion detection engine. This dataset refinement comprised data cleaning, exploratory data analysis,

feature engineering, and dataset splitting. Each of these phases in the pipeline was crucial in ensuring the final dataset is suitable for training the model. Python data preprocessing tools, such as NumPy and Pandas, were used to load the dataset, clean it, and remove non-numeric columns not needed for the analysis. Additionally, the Pyplot library was used to visualize the results of preprocessing, providing graphs that enhanced understanding of the dataset's characteristics.

3.4.1.1 Dataset Cleaning

The objective of dataset cleaning was to prevent bias and ensure the reliability of the data used in training the model. Bias in the dataset compromises the reliability and accuracy of the analysis results, skewing the model's performance and applicability (G. Y. Lee *et al.*, 2021). The research incorporated three features to prevent this phenomenon: addressing missing data, normalizing features, and randomization. Entries with missing features in the dataset were deleted to ensure the remaining dataset was complete. This approach ensured that incomplete entries do not skew the analysis. The features were normalized, providing a statistical analysis environment (G. Y. Lee *et al.*, 2021). This phenomenon indicates that two classes were created during the cleaning process: benign and malware. Further, only numerical features in the dataset were considered for analysis. Randomization involved shuffling the dataset to prevent skewing of the analysis to a designated behavior (P. Li *et al.*, 2021). The ethical considerations in dataset cleaning enhanced the integrity of the cleaning process. For instance, data entries were anonymized to prevent unique traceability to specific devices without compromising their accuracy and relevance. Also, the analysis results were shared in the research to enhance transparency. Additionally, the dataset was balanced between the benign and malicious classes to provide a reasonable learning environment for the model. These ethical considerations offered a comprehensive framework for safeguarding data integrity and ensuring research compliance.

3.4.1.2 Exploratory Data Analysis

The exploratory data analysis (EDA) sought to provide insights into the characteristics and patterns of the datasets. The EDA systematically examines the three datasets to uncover key trends, distributions, and anomalies relating to cybersecurity threats

(Sahoo *et al.*, 2019). The process incorporated statistical and visual techniques to analyze the datasets. For instance, descriptive statistics such as the mean, median, standard deviation, and frequency distributions were computed to summarize the dataset's central tendencies and dispersion. These statistics were instrumental in understanding the dataset's behavior and attributes. Further, they helped identify potential imbalances in the datasets. Such imbalances may influence the model's performance or generalization (Dhany & Izhari, 2023). Visualization techniques were used to offer additional insights regarding the datasets. For example, histograms, box plots, scatter plots, and heat maps describe the datasets. These visualization approaches summarize the datasets while offering key relationships and pattern mappings. Thus, the insights obtained from these techniques were instrumental in modeling the intrusion detection system.

3.4.1.3 Feature Engineering

Feature Engineering was intended to derive key attributes from the dataset. This phase was critical in shaping the performance of the trained model as it involves selecting, transforming, and creating the input variables from the raw dataset (Long *et al.*, 2019). This approach was intended to improve the predictive model's performance and effectiveness. Feature engineering involved identifying relevant features from the dataset obtained from public repositories and transforming them into a format suitable for model training (Long *et al.*, 2019). For instance, the data were encoded as categorical variables. These categorical variables were grouped into two main classes: benign and malware. Since the selected environment was Python, the two classes were identified as "0" for benign and "1" for malware. This class mapping enabled Python's numerical libraries used for machine learning models to handle the datasets effectively.

Ten key attributes were selected for modeling by identifying the significant characteristics in the datasets through principal component analysis (PCA). The PCA approach transformed the dataset of correlated features into a set of uncorrelated variables (Ferrag *et al.*, 2020). These components were ordered by highest variance, with the most significant component first. The PCA identified the dataset's dominant features without relying on predetermined attributes, further reducing potential bias in

the model (Ferrag *et al.*, 2020). These key attributes were used to encode the categories. This phenomenon implies that all attributes were considered in determining whether an entry in the dataset is malicious or benign. This strategy enabled the deep learning model to effectively capture and leverage attributes in classifying requests targeting Implantable Medical Devices. Further, this approach enhanced the effectiveness of the developed model by leveraging modeling libraries to handle the encoded data effectively. Thus, feature engineering improved the efficiency of model development by adequately capturing all the features required by the intrusion detection engine.

3.4.1.4 Dataset Splitting

The datasets used in the research were split into the training and testing subsets. The training dataset was used to develop the model by learning key attributes and patterns from the training subset (Fashoto *et al.*, 2021). The trained model was then evaluated on the test subset, with predicted values compared against actual values in the dataset. The comparisons obtained from this assessment were critical in assessing the model's performance. Each dataset was split into training and test subsets using three ratios. This approach provided an expanded comparative training and testing environment, enabling evaluation of the developed model across different datasets and environments. The first ratio was 60:40, allocating 60% of the data to training and 40% to testing. The smaller training set limited the model's ability to learn complex patterns in the data, potentially leading to lower overall performance. The second ratio was 70:30, allocating 70% to training and 30% to testing. This split offered a relatively balanced compromise between the two extremes, providing a reasonable amount of data for both training and testing (Fashoto *et al.*, 2021). The ratio struck a balance between maximizing the model's learning performance and ensuring reliable evaluation metrics.

The last ratio was 80:20, where 80% of the data was used for training and 20% for testing. This ratio yielded a larger training set, resulting in a more reliable model (Fashoto *et al.*, 2021). However, the smaller test set might have led to higher variance in the evaluation results, as it did not adequately reflect the overall data distribution. These split subsets train and evaluate the model in the three iterations for each dataset. This phenomenon implies that the training and testing were performed for each splitting

ratio. The ratios' overall performance was compared to assess the model's performance across different scenarios. This comprehensive training and testing strategy enhanced the model's selection of the best splitting ratio, refining the comparative assessment of the developed intrusion detection system. The outcome of this initiative was selecting the optimal model across the investigated settings and datasets, resulting in robust intrusion detection in the IMD ecosystem.

3.4.2 Model Development and Optimization

The intrusion detection model was developed through six phases that address specific issues in the design problem. The first phase was dataset preparation, where the dataset was obtained and cleaned. This stage was intended to provide the relevant data environment for the model design. Accurate, properly formatted data ensured an unbiased training environment, preventing overfitting, underfitting, and skewed model performance. Each dataset was standardized using Min-Max scaling. This approach ensured the dataset was normalized to the range [0, 1]. The standardized data was then split into three training and testing ratios. These ratios are the 60% training and 40% testing, 70% training and 30% testing, and 80% training and 20% testing. Each of these training/testing ratios was used in the subsequent stages to provide a comprehensive training and evaluation environment for the model.

The second phase was the model design, which aimed to develop the proposed intrusion detection model. This stage provided the initial model for the IMD ecosystem, serving as the foundation for intrusion detection operations. The initial model was designed with three main features: an encoder, a bottleneck, and a decoder. The encoder and decoder segments comprised one-dimensional Convolutional Neural Networks (1D-CNNs) that were dynamically configured as the model's layer count increased. The dynamic modification of these layers enhanced the model's scalability as the number of layers increased, addressing different design considerations. These CNN layers used the sigmoid and Leaky Rectified Linear Units (Leaky RELU) as activation functions. These activation functions provided a fast-inference environment, effectively captured class probabilities for intrusion detection, and extensively extracted feature maps from the input data (Dubey *et al.*, 2022). The number of hidden layers in the encoder and

decoder sections was equal to provide a balanced reconstruction environment for the model through mapping of the corresponding encoder and decoder layers (Mienye & Swart, 2025). The generic design of these hidden layers enabled fast, efficient scaling of the model without redesigning its components.

The bottleneck layer consisted of one long short-term memory (LSTM). The LSTM component was pivotal in capturing temporal dependencies within the sequential data (Fenanir *et al.*, 2020). This feature enabled it to extend the encoder's capabilities, but it limited its performance in learning key features for effective threat identification. Further, the LSTM used Exponential Linear Units (ELU) and Scaled Exponential Linear Units (SELU) as the activation functions. These activation functions reduced the vanishing gradient problem in the encoder, accelerated learning, and enhanced the model's self-normalization (Kilicarslan *et al.*, 2021; Dubey *et al.*, 2022). The learning rate was optimized to balance convergence speed and model stability during the training (Fenanir *et al.*, 2020). Also, the loss function was tailored to the unique characteristics of the dataset, providing an efficient intrusion-detection environment. Lastly, the number of training epochs was carefully selected to improve the efficiency of the training model. This selection involved integrating early stopping to prevent overfitting and improve generalizability (Andresini *et al.*, 2019).

The third phase was model training, during which the deep autoencoder was trained for threat detection in the Medical Internet of Things. This phase sought to provide the actual model and train it with the dataset cleaned in the previous stages. This training used the three datasets and the designed model prepared in the preceding phase. The three training/testing ratios were then used to train the developed model, with the number of hidden layers in the encoder and decoder increased from 1 to 8. This scaling up of the hidden layers was observed in each training/testing session, with the results being recorded for comparative assessment with convolutional neural networks, long short-term memory, and autoencoder models. Hidden layers in the deep autoencoder model were varied from 1 to 8, with each configuration trained, evaluated, and results recorded for comparative assessment. This approach provided a comprehensive model

training and evaluation environment to identify the optimal configuration for the proposed intrusion detection system in the implantable medical devices ecosystem.

The fourth phase was optimization, where the model was fine-tuned by incorporating additional features derived from the performance evaluation. The goal of this stage was to increase the model's accuracy, recall, precision, and F1 scores, and to reduce the false-positive rate and inference time. Optimization for the hybrid model involved integrating the Nesterov-accelerated Adaptive Moment Estimation (Nadam) optimization into the proposed model. This approach sought to improve the model's performance by reducing its constraints. The fifth stage was model evaluation, where the model was assessed to determine its performance. This evaluation checked the performance metrics such as recall, accuracy, precision, F1, and inference time. Since the model did not achieve the desired 99.95% accuracy, recall, precision, and F1 across all datasets, further improvement was needed.

The last phase was the integration of the machine learning models into the intrusion detection engine. The objective of this phase was to improve further the model's accuracy, recall, precision, and F1 score, and to reduce inference time and false-positive rates. This improvement resolved the limitations of these scores in the preceding phase, offering a better version of the model. The integration involved incorporating a multilayer perceptron into the model, further enhancing the hybrid nature of the IDS. The multilayer perceptron integrated as the classification head expanded the model's architecture into four segments. Three hidden layers were selected as this number provided the minimal environment for the model. The first layer of the MLP served as the input, scaling the encoder's output; the second layer extracted features from the dataset; and the third layer classified traffic as either malicious or benign. This final layer enhanced the encoder's classification, further refining the accuracy, recall, precision, and F1 scores.

The initial encoder, bottleneck, and decoder segments remained consistent with the optimized hybrid model design developed in the second phase. The retention of these segments enhanced the code's reusability and the ease of advancing the model. For

instance, it was easy to leverage the existing hybrid deep autoencoder model with a few adjustments to handle the new design requirements for the MLP-enhanced model. The encoder and decoder layers were scalable, providing an adjustable environment for achieving the desired model configuration. For instance, the layers could be increased from 1 to 7, with the model evaluated on accuracy, recall, precision, F1, and inference time at each increase. This adjustment led to the identification of the four hidden layers in the encoder and decoder segments as the ideal model configuration on the ICU dataset. The model selected for the IDS is shown in Figure 3.1, and the development process is shown in Figure 3.2.

All graphs in the model were generated using Python's Pyplot library. This approach enabled the training and evaluation pipelines to run sequentially, reducing the need for interruption. Further, the training and evaluation metrics were stored in a CSV file, providing a reference point for future review of the model performance under different considerations. For instance, these stored scores were used to generate comparative graphs for the selected models across various model configurations. The scores were copied into Microsoft Excel, and simple graphing techniques were used to plot the graphs. The reason for this approach was to extend the graphs generated by Python with simple graphs to compare the performance of the selected models.

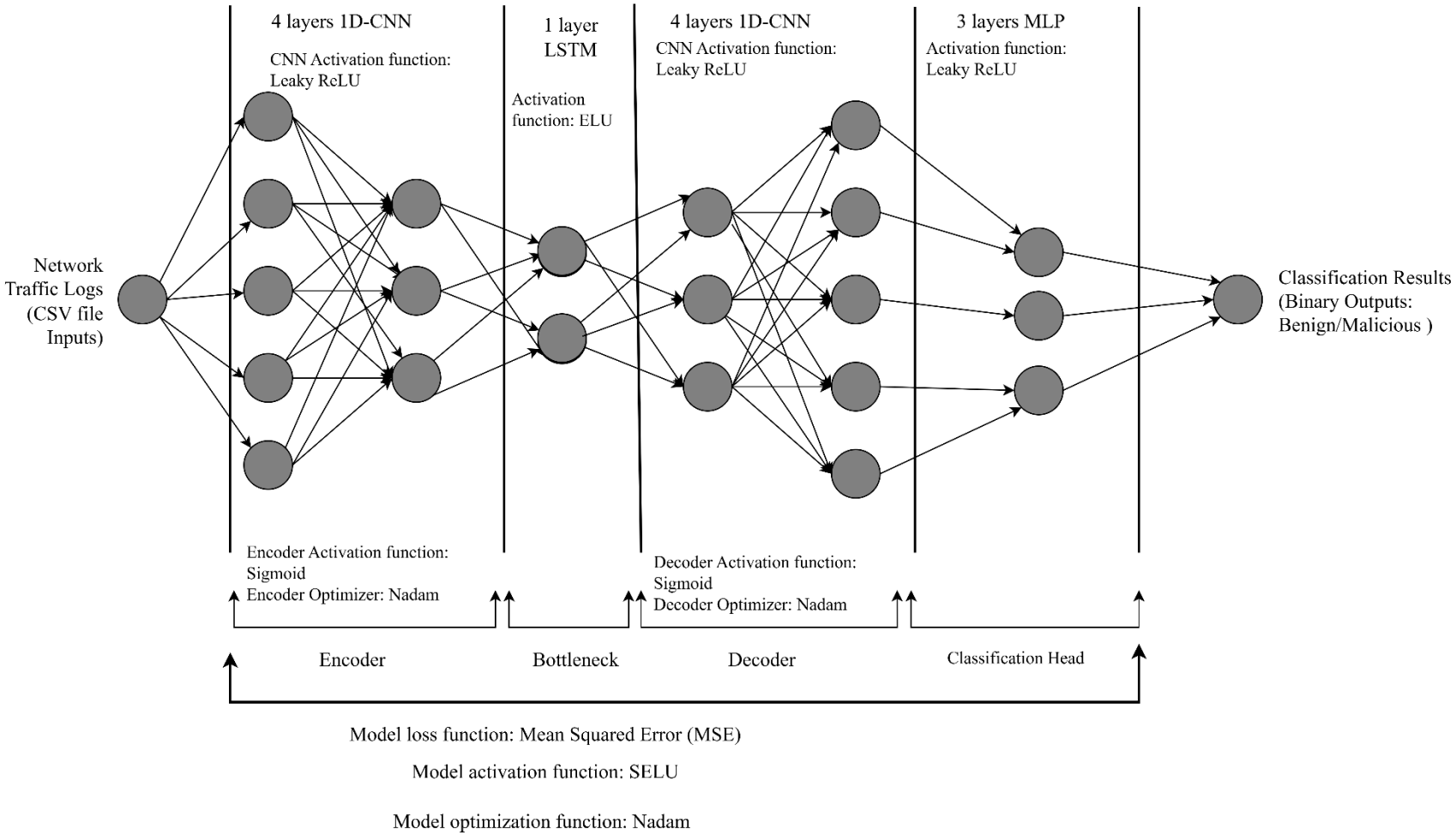


Figure 3.1: Proposed hybrid model architecture

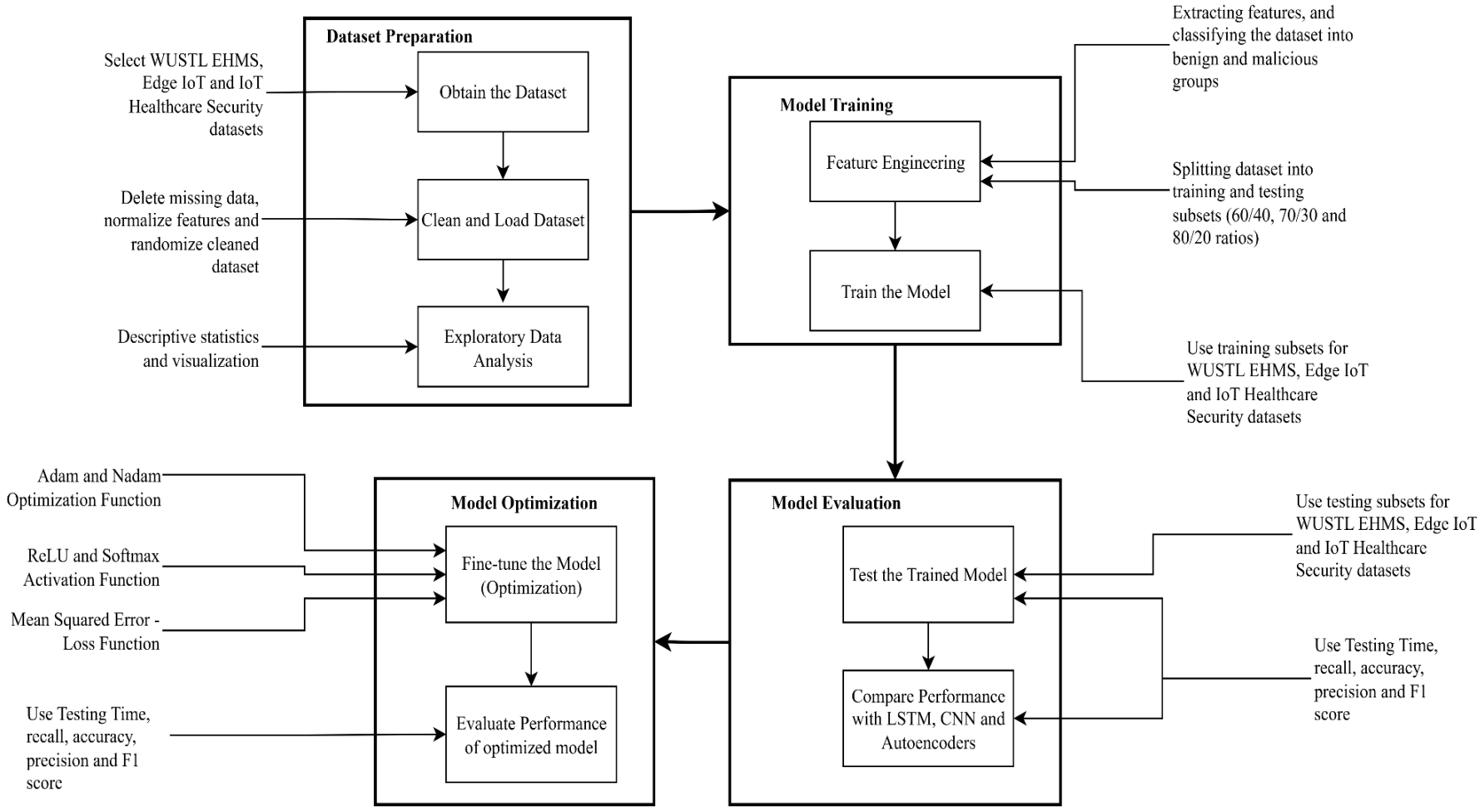


Figure 3.2: Model development process

3.4.3 Model Evaluation

The proposed model was evaluated based on five primary metrics. The first metric was inference time; the total time the model took to classify threats. This metric was considered as it provided a robust environment for assessing the model's responsiveness in identifying threats. For instance, a slow model would not be effective in providing real-time or near-real-time detection capabilities for the IMD ecosystem. The second metric was accuracy, which measured the model's ability to correctly identify threats. This evaluation was selected because it offered an elaborate framework for assessing the model's ability to distinguish between malicious and non-malicious traffic using the provided dataset. A model that misclassifies malicious traffic as benign enables hackers to bypass intrusion detection systems, compromising the underlying information infrastructure.

The third metric considered was precision, which provided the proportion of false alarms. This metric provided insights into the model's disruption of regular traffic due to erroneous classification. Such classification errors would disrupt normal communication operations, as the system administrator would be notified of the potential threat when there was none. The fourth metric was recall, which measured the model's ability to classify malicious traffic accurately. Failure of the model to accurately classify malicious traffic enables hackers to maintain a presence within the healthcare infrastructure undetected, leading to its compromise. The last metric was the F1 score, which balances the trade-off between false positives and negatives. This metric assessed the model's ability to handle false positives and negatives in the IMD ecosystem. Collectively, the five metrics provided a comprehensive assessment framework for evaluating the model's suitability in the IMD ecosystem, based on real-time detection capabilities, accuracy, and error rates. Thus, these metrics enhanced the understanding of the model's performance in the deployment settings.

The ideal scores for these metrics were arbitrarily set before implementing the model to provide an informed decision-making framework. For instance, the ideal inference time for the developed model was arbitrarily set to 1.5 seconds to simulate near-real-time behavior. If the developed model had a score below this, it would be considered suitable for deployment. The acceptable accuracy, precision, recall, and F1 scores for

the model were set at 99.95%. This situation implies that the model needed to achieve at least 99.95% in each of these metrics before being considered reasonable. Nevertheless, there was a need to address the potential trade-off between these scores and the inference time. If such a situation arose, consideration would be given to accuracy, precision, recall, and F1 before selecting the inference time. This approach would ensure that model accuracy is prioritized over latency when evaluating its suitability for the clinical environment.

3.4.4 Considerations for Lightweight Model Architecture

The primary objective of the developed model was to provide a lightweight IDS solution while ensuring optimal accuracy, recall, precision, and F1 scores. This objective was achieved through three considerations in the design process. First, the encoder and decoder segments of the model were designed to use small, efficient convolutional layers that extracted flow patterns without increasing computational cost. The integration of separable CNN layers with depth control and small kernels has achieved significant reductions in memory and computational costs (Dantas *et al.*, 2024). The integration of this feature into the model significantly enhanced the IDS's lightweight nature, enabling its deployment in the IMD ecosystem. Secondly, the use of a narrow bottleneck layer in the model reduces the memory requirements for the resulting hybrid autoencoder. The narrow bottleneck layer minimizes the model's memory requirements by storing only minimal temporal features (Rassam, 2024). This feature further reduces the model's memory requirements, aligning with the lightweight objectives.

Lastly, there is a complex balance between detection accuracy and inference time through a tiered deployment strategy. The IDS model is deployed in the controller, which has reasonable processing and storage resources to manage the connected IMD devices. This controller is connected to the healthcare provider’s cloud platform, which provides regular training and fine-tuning. The short-range communication mechanisms used between the IMDs and controller reduce delays in identifying and responding to threats, providing near-real-time detection capabilities. The resulting implementation mechanism is a trade-off between inference time and functionality. For instance, communication between the controller and the IMD device is near real-time, as these

devices use short-range signals that reduce latency. Although implementing the model in an IMD device would have provided real-time detection capabilities, such an approach would be impractical given these devices' resource constraints and their inability to run the deployed model due to platform limitations. Thus, there is a subtle trade-off between performance and latency in the developed model.

3.4.5 Hardware and Software Specifications

The machine learning modeling environment was implemented in Google Colab. This platform was selected as it is a cloud-based computing environment with pre-installed machine learning libraries. This phenomenon fast-tracked the training and testing operations for the models. The Google Colab environment had 15GB of Graphics Processing Unit (GPU), 12GB of Random Access Memory, and 112GB of disk storage. The GPU had a single-precision (FP32) floating-point performance of 8.1 tera-floating-point operations per second (TFLOPS). This situation implies that the GPU could perform 8.1 trillion floating-point operations per second, providing a high computation environment for the model. The high GPU and RAM were selected to provide the ideal computational environment for developing the model, as deep neural networks require intensive computational resources (K. J. Lee, 2021). A relatively high storage environment was chosen for storing the model files. These files would be used to compare memory utilization (storage space) across the different models. This configuration is shown in Figure 3.3.

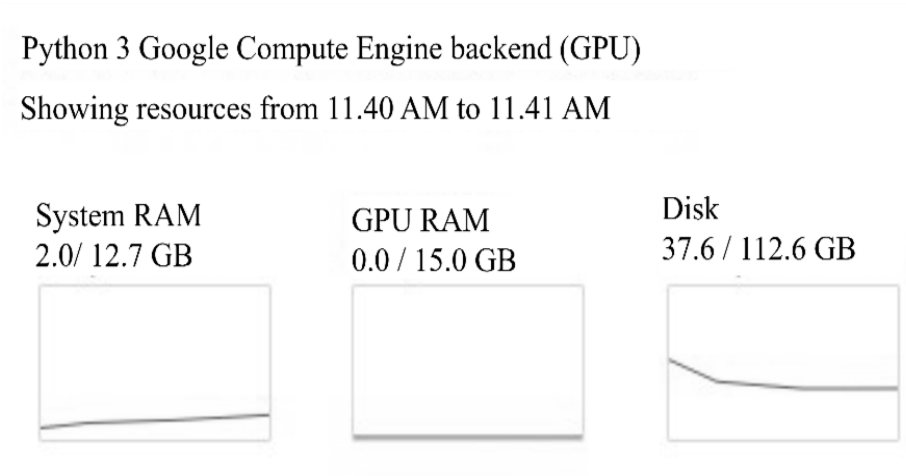


Figure 3.3: Google Colab environment

Additionally, selecting a cloud environment for training and evaluating the model reduced reliance on local computing resources, aligning with the research's objective of leveraging cloud infrastructure to implement the proposed Intrusion Detection System. The Python environment used in Google Colab was the Jupyter Notebook, which provided an interactive Python coding environment. This feature enhanced error identification and resolution early in the modeling process, ensuring the developed model was practical and addressed the design goals.

3.5 Ethical Considerations

There were ethical considerations in the research process, which enhanced the model's credibility and reliability. Firstly, ethical approval was sought from the Ethics Committee at Chuka University (Appendix 5) to ensure that the research adheres to ethical guidelines and safeguards the rights and well-being of participants involved in the study. This approval process was essential to ensure that the research is conducted in an ethically responsible manner, with due consideration given to potential risks and benefits. Secondly, the researcher applied for a research permit from Kenya's National Commission for Science, Technology, and Innovation (NACOSTI) (Appendix 6). This permit was necessary to conduct research involving data collection, analysis, and experimentation within the country. By obtaining the required research permit, the study demonstrated compliance with national regulations and ensured that the research activities were conducted legally and ethically. Ethical issues that emerged during the research were resolved in accordance with the guidelines issued by Chuka University's Ethics Committee and NACOSTI. This approach offered a comprehensive ethical management framework that enhances compliance with the research expectations. Thirdly, this study upheld principles of academic integrity by adequately citing and acknowledging all content sourced from other authors. By recognizing the work of others, the research project ensured transparency, credibility, and respect for intellectual property rights. Fourthly, there was a responsible disclosure of vulnerabilities identified during the research process. This situation implies that the findings obtained from the research were shared in line with the research disclosure guidelines. Further, harm to the target systems was minimized. This situation implies that the researcher did not engage in activities that intentionally exposed the computing resources. Instead, the researcher worked with the involved parties to minimize potential risks and harms

across the systems studied throughout the research process. Also, data collected and analyzed in the research were protected at all costs in compliance with privacy laws and other regulations. This phenomenon enhanced the confidentiality of the sensitive information resources collected and stored during the study. Lastly, there was respect for Intellectual Property Rights throughout the research process. This situation implies that the researcher did not participate in activities that contravene Intellectual Property Rights and strove to enhance compliance with established guidelines.

CHAPTER FOUR
RESULTS

4.1 Introduction

The designed model was implemented in the Google Colab environment, trained, and tested across three datasets and three training/testing ratios. The baseline models were developed, trained, and tested, alongside the hybrid deep autoencoder. The standard hybrid deep autoencoder was first deployed and then optimized by integrating the Nadam optimizer. After training the Gradient Boosting, Random Forests, and Multilayer Perceptron, the MLP was incorporated into the Nadam-optimized hybrid deep autoencoder and trained. This chapter presents the results and analysis from this experiment, assessing the model's ability to meet the design objectives.

4.2 Model Design and Training

The architectures for the standard, Nadam-optimized, and MLP-enhanced models with one hidden layer are presented in Figures 4.1, 4.2, and 4.3, respectively.

Deep Autoencoder Model without Optimizer
Model: "sequential"

Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 25, 10)	0
conv1d (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu (LeakyReLU)	(None, 25, 64)	0
conv1d_1 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_1 (LeakyReLU)	(None, 25, 64)	0
max_pooling1d (MaxPooling1D)	(None, 12, 64)	0
flatten (Flatten)	(None, 768)	0
dense (Dense)	(None, 250)	192,250
reshape_1 (Reshape)	(None, 250, 1)	0
lstm (LSTM)	(None, 250, 64)	16,896
elu (ELU)	(None, 250, 64)	0
flatten_1 (Flatten)	(None, 16000)	0
dense_1 (Dense)	(None, 768)	12,288,768
reshape_2 (Reshape)	(None, 12, 64)	0
up_sampling1d (UpSampling1D)	(None, 24, 64)	0
conv1d_2 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_2 (LeakyReLU)	(None, 24, 64)	0
conv1d_3 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_3 (LeakyReLU)	(None, 24, 64)	0
flatten_2 (Flatten)	(None, 1536)	0
dense_2 (Dense)	(None, 250)	384,250
reshape_3 (Reshape)	(None, 25, 10)	0

Total params: 12,921,204 (49.29 MB)
Trainable params: 12,921,204 (49.29 MB)
Non-trainable params: 0 (0.00 B)

Figure 4.1: CNN-LSTM-CNN hybrid model architecture for one hidden layer

Deep Autoencoder Model with Nadam Optimizer
Model: "sequential_2"

Layer (type)	Output Shape	Param #
reshape_8 (Reshape)	(None, 25, 10)	0
conv1d_8 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_8 (LeakyReLU)	(None, 25, 64)	0
conv1d_9 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_9 (LeakyReLU)	(None, 25, 64)	0
max_pooling1d_2 (MaxPooling1D)	(None, 12, 64)	0
flatten_6 (Flatten)	(None, 768)	0
dense_6 (Dense)	(None, 250)	192,250
reshape_9 (Reshape)	(None, 250, 1)	0
lstm_2 (LSTM)	(None, 250, 64)	16,896
elu_2 (ELU)	(None, 250, 64)	0
flatten_7 (Flatten)	(None, 16000)	0
dense_7 (Dense)	(None, 768)	12,288,768
reshape_10 (Reshape)	(None, 12, 64)	0
up_sampling1d_2 (UpSampling1D)	(None, 24, 64)	0
conv1d_10 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_10 (LeakyReLU)	(None, 24, 64)	0
conv1d_11 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_11 (LeakyReLU)	(None, 24, 64)	0
flatten_8 (Flatten)	(None, 1536)	0
dense_8 (Dense)	(None, 250)	384,250
reshape_11 (Reshape)	(None, 25, 10)	0

Total params: 12,921,204 (49.29 MB)
Trainable params: 12,921,204 (49.29 MB)
Non-trainable params: 0 (0.00 B)

Figure 4.2: CNN-LSTM-CNN Nadam-optimized hybrid model architecture for one hidden layer

Deep Autoencoder Model with Nadam Optimizer, Regularization and Batch Optimization
Model: "functional_2"

Layer (type)	Output Shape	Param #	Connected to
input_layer_2 (InputLayer)	(None, 25, 10)	0	-
conv1d_6 (Conv1D)	(None, 25, 64)	1,984	input_layer_2[0]…
batch_normalizatio… (BatchNormalizatio…	(None, 25, 64)	256	conv1d_6[0][0]
activation_6 (Activation)	(None, 25, 64)	0	batch_normalizat…
max_pooling1d_2 (MaxPooling1D)	(None, 12, 64)	0	activation_6[0][…
dropout_4 (Dropout)	(None, 12, 64)	0	max_pooling1d_2[…
bidirectional_4 (Bidirectional)	(None, 256)	197,632	dropout_4[0][0]
repeat_vector_2 (RepeatVector)	(None, 1, 256)	0	bidirectional_4[…
bidirectional_5 (Bidirectional)	(None, 1, 256)	394,240	repeat_vector_2[…
conv1d_7 (Conv1D)	(None, 1, 128)	98,432	bidirectional_5[…
activation_7 (Activation)	(None, 1, 128)	0	conv1d_7[0][0]
up_sampling1d_2 (UpSampling1D)	(None, 2, 128)	0	activation_7[0][…
conv1d_8 (Conv1D)	(None, 2, 64)	24,640	up_sampling1d_2[…
activation_8 (Activation)	(None, 2, 64)	0	conv1d_8[0][0]
flatten_2 (Flatten)	(None, 128)	0	activation_8[0][…
concatenate_2 (Concatenate)	(None, 384)	0	bidirectional_4[… flatten_2[0][0]
dense_6 (Dense)	(None, 128)	49,280	concatenate_2[0]…
batch_normalizatio… (BatchNormalizatio…	(None, 128)	512	dense_6[0][0]
dropout_5 (Dropout)	(None, 128)	0	batch_normalizat…
dense_7 (Dense)	(None, 64)	8,256	dropout_5[0][0]
dense_8 (Dense)	(None, 1)	65	dense_7[0][0]

Total params: 775,297 (2.96 MB)
Trainable params: 774,913 (2.96 MB)
Non-trainable params: 384 (1.50 KB)

Figure 4.3: CNN-LSTM-CNN Nadam-optimized and MLP-enhanced hybrid model architecture for one hidden layer

4.3 Hybrid Model Development

The initial phase of model development involved integrating convolutional neural networks into the encoder and decoder and using long short-term memory in the

bottleneck layer to construct a hybrid deep autoencoder. The resulting model was trained 8 times by varying the number of hidden layers in the encoder and decoder, with the bottleneck layers set to 1 in each case. This training was conducted for the three datasets and three training/testing ratios in each dataset. The results from these sessions were evaluated based on precision, accuracy, recall, F1 score, and inference time.

4.3.1 Hybrid Model on the ICU Dataset

A review of recall results from the Intensive Care Unit (ICU) dataset across baseline models and the eight configurations of the proposed hybrid model revealed the models' sensitivities. For instance, the CNN's 70/30 model achieved the highest score of 71.45%, outperforming both baseline and hybrid models. This model was followed by the CNN's 80/20 and LSTM's 80/20 variants, which achieved scores of 50.95% and 50.72%, respectively. The eight hidden layers of the hybrid model's 80/20 variant achieved a recall of 50.72%, while the three hidden layers' 80/20 variant achieved a recall of 50.47%. The seven-hidden-layer 60/40 model achieved a recall score of 50.36%, matching those of the LSTM and CNN 60/40 variants. This phenomenon indicates that few hybrid models, such as the 80/20 variants of the eight- and three-hidden-layer models, performed similarly to the baseline models. These results are shown in Figure 4.4.

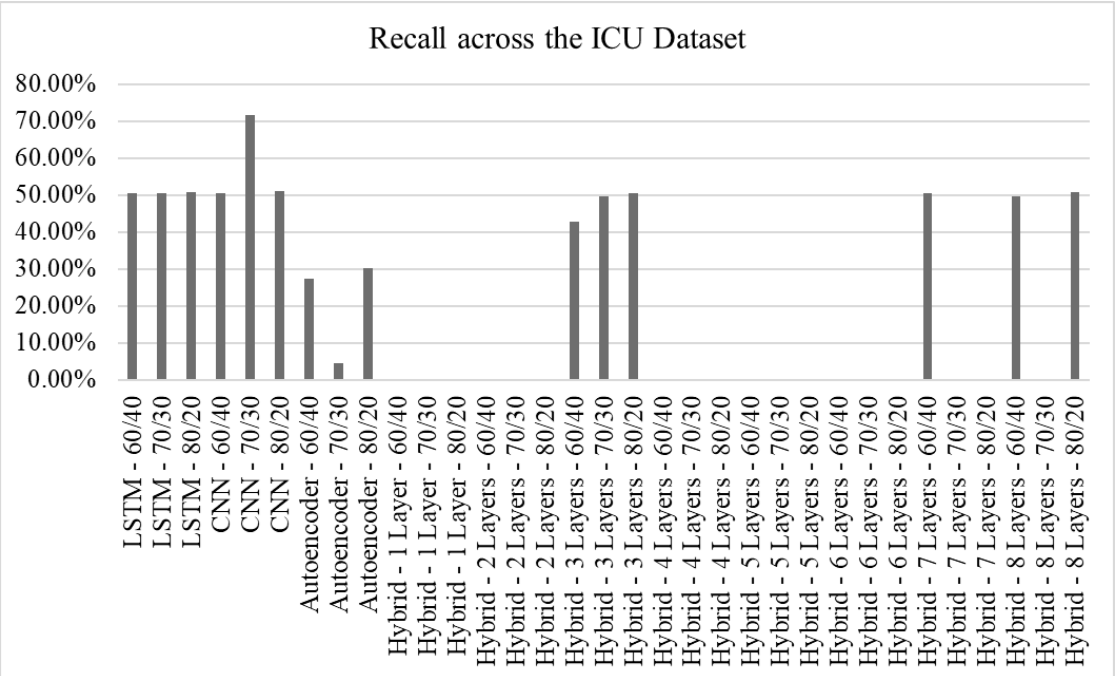


Figure 4.4: Recall scores for the ICU dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

Analysis of precision scores across the dataset revealed the models' positive predictive performance. The CNN's 70/30 ratio was the highest at 79.43%, indicating strong confidence in traffic classification. This model was followed by the CNN's 80/20 variant at 75.07%. The 60/40 and 80/20 variants of the autoencoder model achieved scores of 64.48% and 61.68%, respectively. The three-hidden-layer variant of the hybrid model (60/40) scored 50.36%, closely followed by the 70/30 variant with one hidden layer and the 70/30 variant with two hidden layers. The 70/30 variant of the three hidden layers achieved a precision score of 49.67%. While several other variants of the hybrid model had zero precision scores, 70/30 variants with 1 to 3 hidden layers demonstrated fair performance compared to the baseline models. These scores are shown in Figure 4.5.

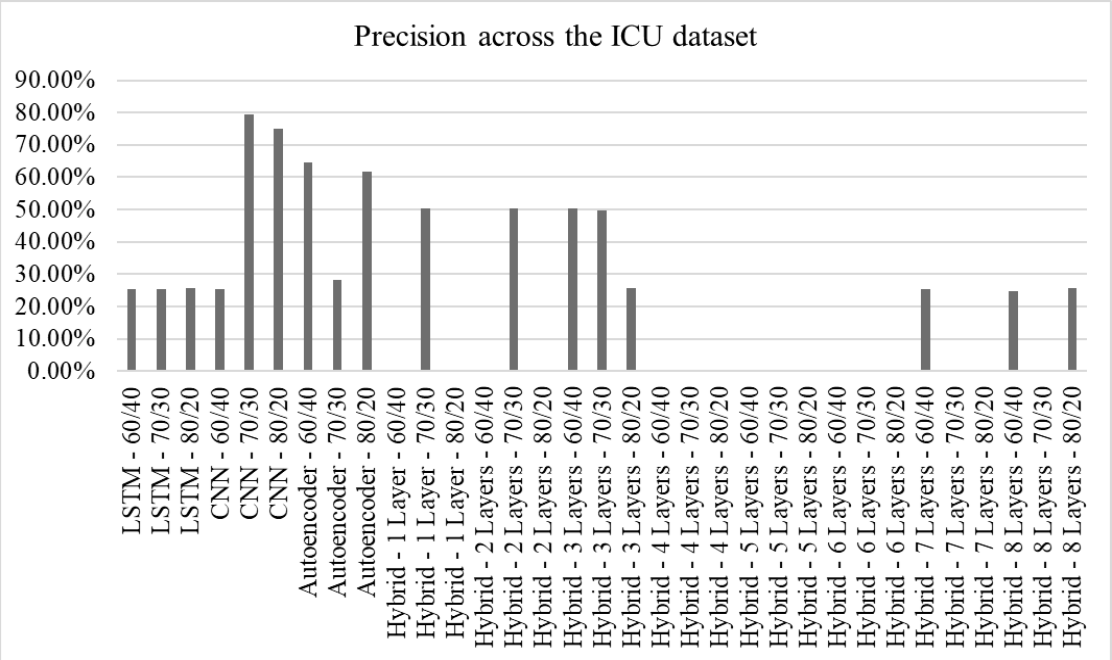


Figure 4.5: Precision scores for the ICU dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

The assessment of specificity scores across the baseline and hybrid models shows the true negative rates for the models. The autoencoder models provide the highest scores, with the 60/40 variant leading at 91.39%. The 80/20 and 70/30 variants of the autoencoder have 91.21% and 89.90% respectively. Different variants of the hybrid models follow these baseline models. For example, the three-hidden-layers' 60/40 had a score of 88.54%, while the two-hidden-layers' 80/20 and 60/40 variants had scores of 87.50% and 85.71%, respectively. The 70/30 and 80/20 variants of the three-and four-hidden-layer models had scores of 83.22% and 80.00%, respectively. These scores were

higher than those from the CNN and LSTM variants, though lower than those from the autoencoders. These results are shown in Figure 4.6.

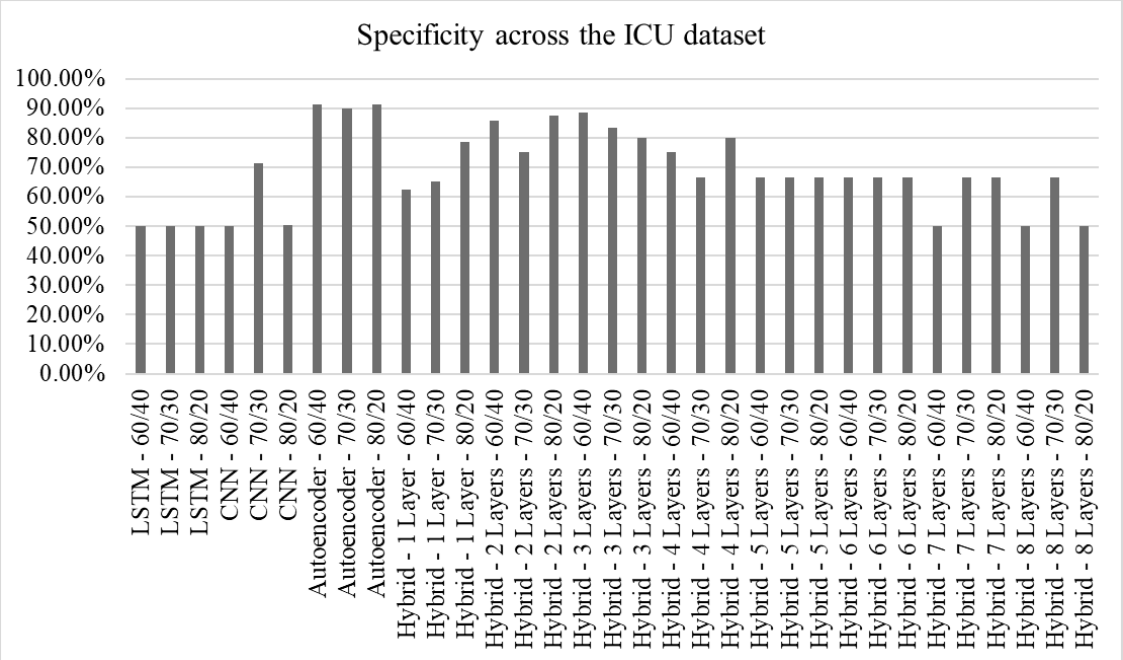


Figure 4.6: Precision scores for the ICU dataset across the LSTM, CNN, Autoencoder, and hybrid deep autoencoder model variants

The analysis of accuracy scores across the dataset provides insights regarding the models' general correctness. The CNN's 70/30 variant had the highest accuracy score at 71.45%, followed distantly by the CNN's 80/20 variant at 50.95%. The 80/20 variants for LSTM and the hybrid model's eight hidden layers followed with a score of 50.72%. The three hidden-layer 80/20 variant scores 50.47%, followed by the 60/40 variants for LSTM, CNN, and seven hidden layers at 50.36%. The LSTM's 70/30 variant scores 50.33%, which is similar to that of some hybrid models. This phenomenon demonstrates that the hybrid model offers reasonable accuracy compared to the baseline models. These results are shown in Figure 4.25.

The review of F1 scores across the baseline and hybrid models demonstrates the balance between precision and recall. CNN's 70/30 had the highest F1 score at 69.30%. The three hidden layers' 70/30 and 60/40 variants yielded 49.67% and 46.22%, respectively. The autoencoder's 80/20 and 60/40 variants achieved scores of 40.38% and 37.56%, respectively. Although the three hidden layers' 7030 and 60/40 variants had scores below 50%, their relatively high performance compared to the baseline models

demonstrates the promise of hybrid models for intrusion detection. These results are shown in Figure 4.7.

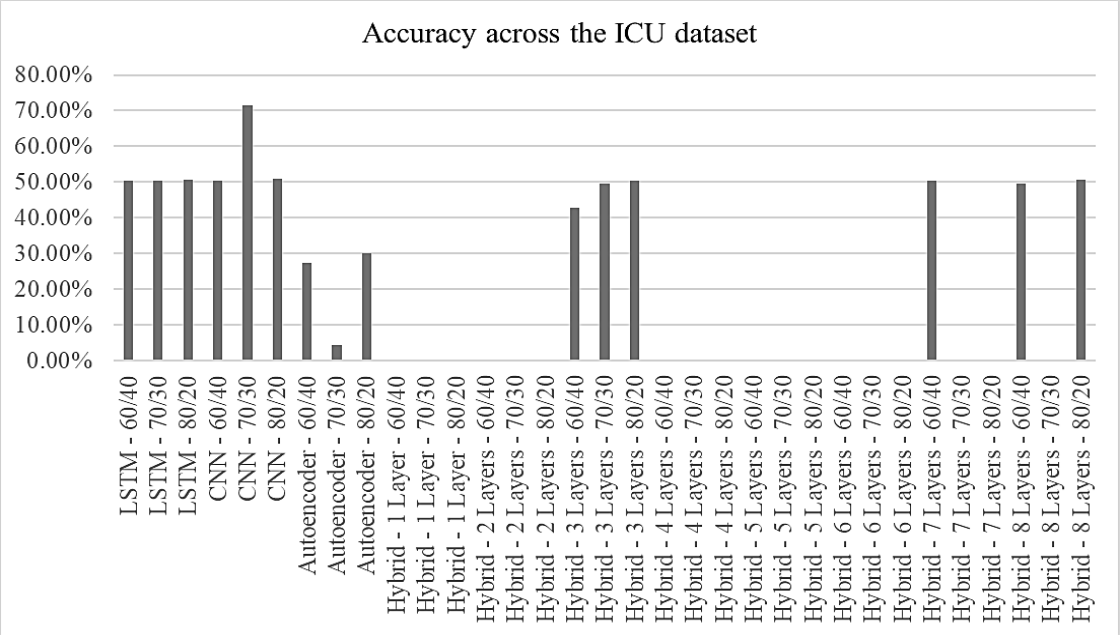


Figure 4.7: Accuracy scores for the ICU dataset across the LSTM, CNN, Autoencoder, and hybrid deep autoencoder model variants

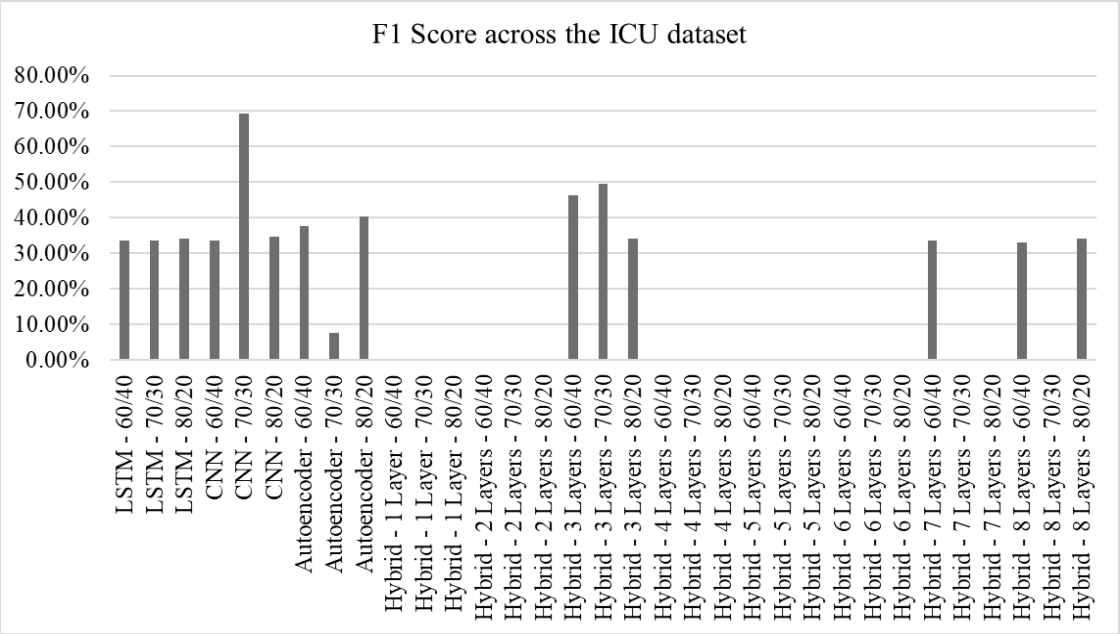


Figure 4.8: F1 scores for the ICU dataset across the LSTM, CNN, Autoencoder, and hybrid deep autoencoder model variants

The review of false positive rates across the baseline and hybrid models shows the models' performance in classifying threats. The autoencoder models emerge as the best, with the lowest FPR rates of 8.61% for 60/40, 8.79% for 80/20, and 10.10% for 70/30.

These models are then followed by hybrid models, with the three hidden layers' 60/40 taking the lead at 11.46%. The two-hidden-layer 80/20 and 60/40 variants score 12.50% and 14.29%, respectively. The three hidden layers' 70/30 and the four hidden layers' 80/20 close the 20% mark at 16.78% and 20.00%, respectively. This situation shows that hybrid models achieve promising false-positive performance, outperforming non-encoder-based models. These results are shown in Figure 4.9.

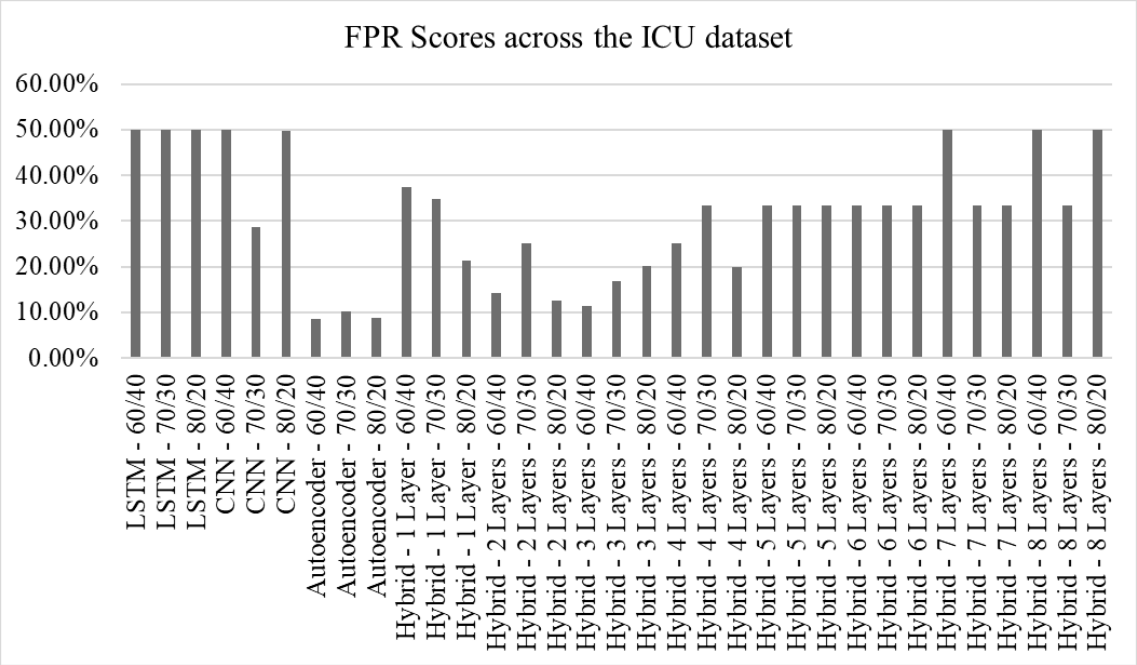


Figure 4.9: FPR scores for the ICU dataset across the LSTM, CNN, Autoencoder, and hybrid deep autoencoder model variants

The analysis of inference time for the baseline and hybrid models across the dataset shows that the LSTM's 70/30 variant is the fastest, at 0.64 seconds. This model is followed by the CNN's 80/20 variant at 0.69 seconds and the Autoencoder's 60/40 variant at 0.82 seconds. The best-performing hybrid model in terms of inference time is the 80/20 variant with three hidden layers at 1.35 seconds. This phenomenon demonstrates that the hybrid models perform poorly in terms of inference speed. These results are shown in Figure 4.10.

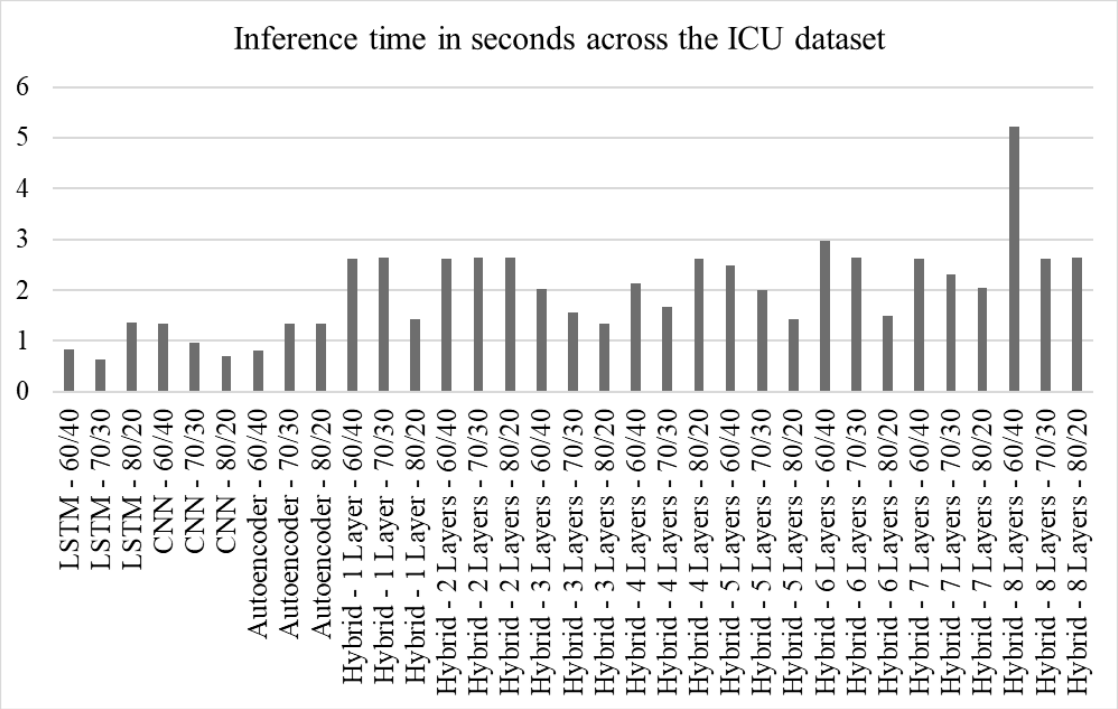


Figure 4.10: Inference time in seconds for the ICU dataset across the LSTM, CNN, Autoencoder, and hybrid deep autoencoder model variants

The analysis of selected hybrid models provides insights into their architecture, training process, and performance. The models considered for analysis in the ICU dataset must achieve at least 50% accuracy. This criterion led to the selection of three models: the 80/20 variants with eight hidden layers, three hidden layers, and seven hidden layers with 60/40 splits. The analysis of the Receiver Operating Characteristics (ROC) graph for the 80/20 variant with eight hidden layers enhances understanding of the model's performance. The model achieved an Area Under the ROC (AUC) of 0.5, indicating that it randomly guessed the classification. This phenomenon is evident in the confusion matrix, where the model identified 2,957 false negatives.

The review of the testing and validation loss curve offers insights into what might have led to this situation. The model's testing loss started at zero, then increased slightly to about eight before falling back to zero in the third epoch. It then rose to 70 in the fourth epoch, dropped to 30 in the fifth epoch, and slightly rose to about 45 in the sixth epoch. It then fell to ten in the seventh epoch, and reached zero in the eighth epoch, before rising to 30 in the ninth epoch. This phenomenon implies that the model did not stabilize during testing and may not stabilize even with additional training. Further, the high

testing loss indicates the model's unreliability, as evidenced by the low AUC value. These results are shown in Figures 4.11, 4.12, and 4.13.

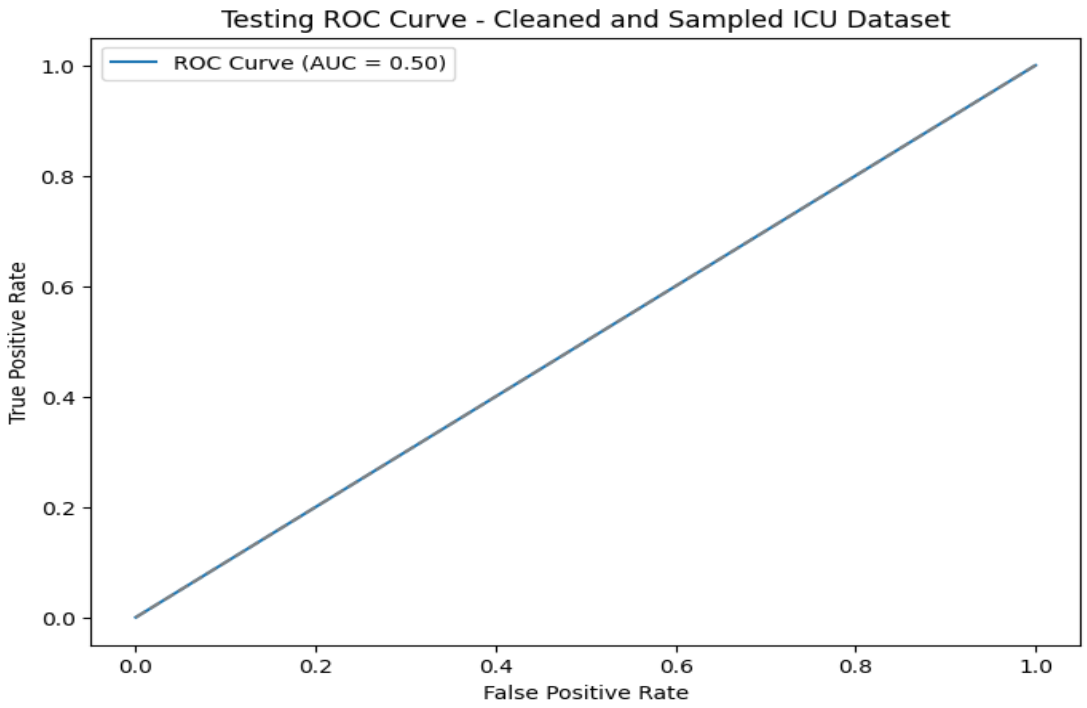


Figure 4.11: Receiver Operating Characteristics curve for the eight-hidden-layers' 80/20 hybrid model on the ICU dataset

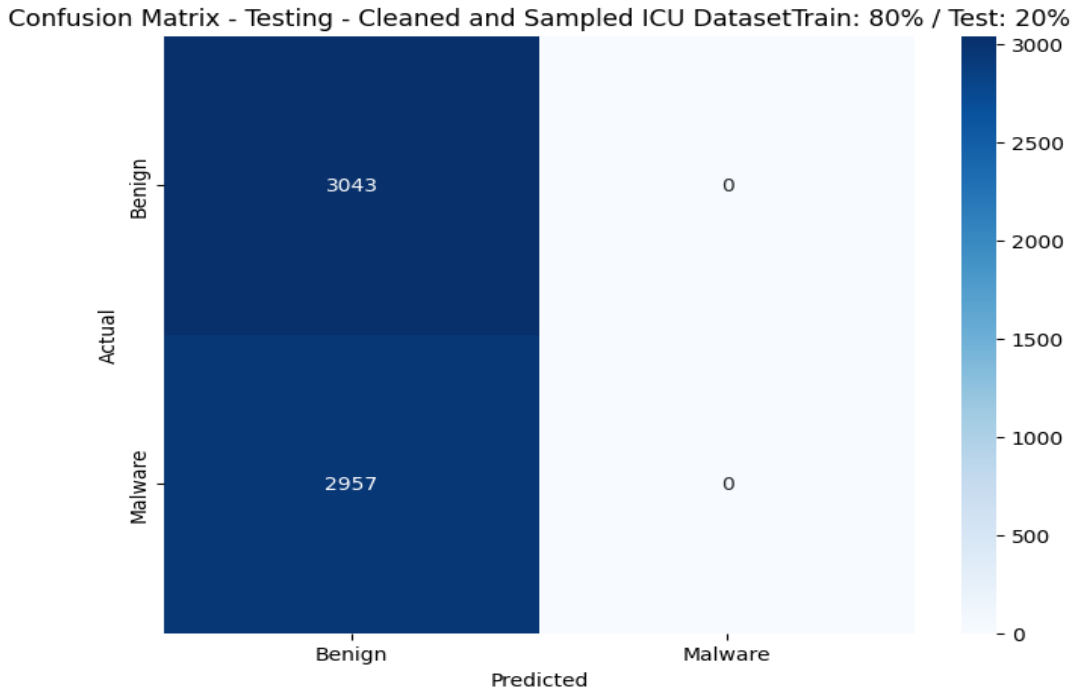


Figure 4.12: Confusion matrix for eight-hidden-layers 80/20 hybrid model on the ICU dataset

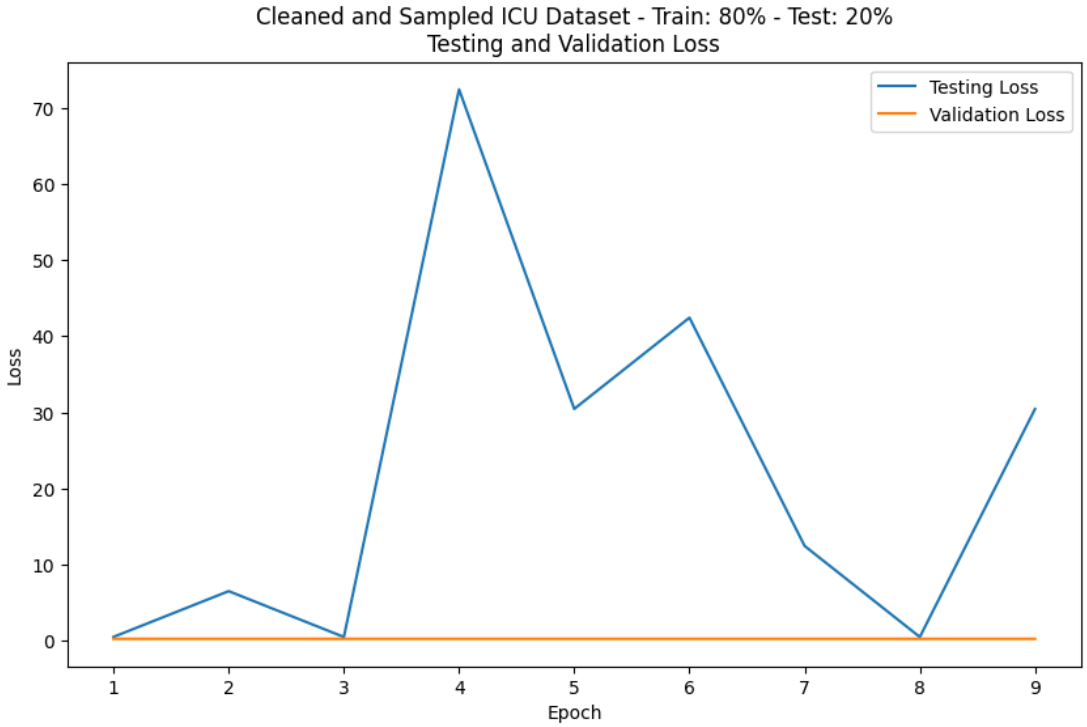


Figure 4.13: Testing and validation loss curves for eight-hidden-layers' 80/20 hybrid model on the ICU dataset

The analysis of ROC, confusion matrices, and loss curves for the 80/20 variant with three hidden layers provides insights into the model's training and testing performance. The ROC curve shows non-discriminatory behavior, with an AUC of 0.5. This poor classification behavior is evident in the model's relatively high number of false negatives, which totaled 2,953. While the model demonstrated a slight improvement over the eight-hidden-layers 80/20 variant, it recorded 15 false positives. This value was higher than that presented in the eight-hidden-layers' 80/20 variant, affecting the overall model performance.

Analysis of the model's testing and validation loss curves provides insights into the factors that drive its performance. The testing loss curve started at about 5, increased to 40 in the second epoch, then fell to 0 in the third epoch. This loss value increased to about 15 in the fourth epoch, then reached 20 in the fifth, before falling to 10 in the sixth. It rose to 40 in the seventh epoch due to the adaptive learning rate, before falling to 30 in the eighth epoch. It then rose to 40 in the ninth epoch and maintained that value until the tenth. Additional training for the model would have enhanced its stability, but

would not have reduced the loss value based on the recorded performance. These results are shown in Figures 4.14, 4.15, and 4.16.

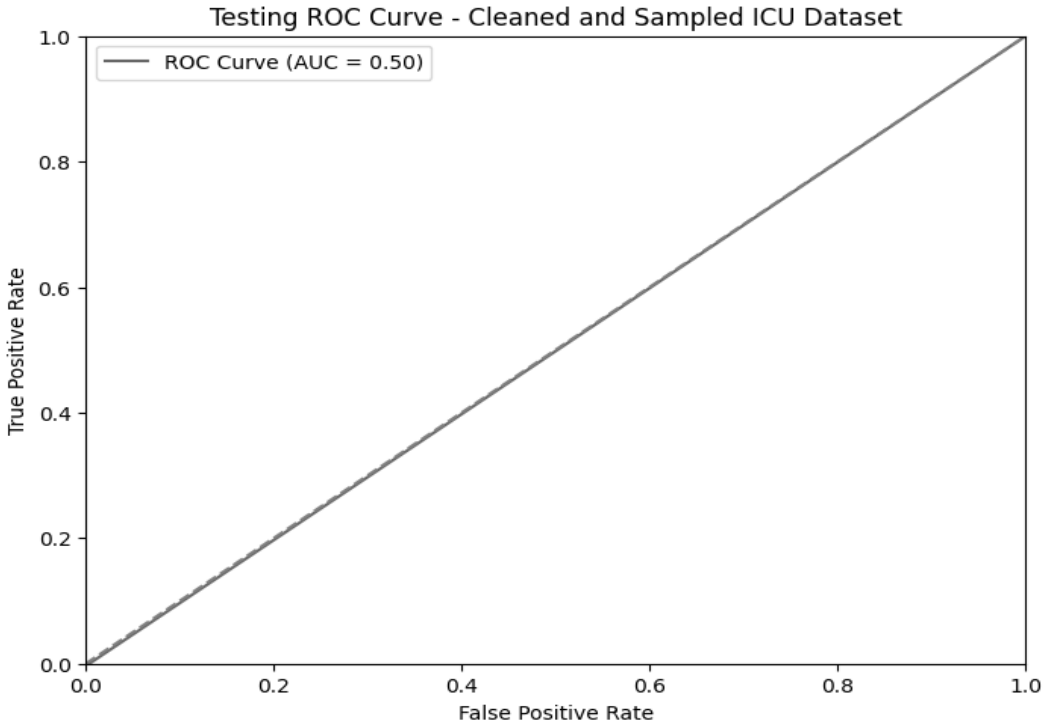


Figure 4.14: Receiver Operating Characteristics curve for three hidden layers' 80/20 hybrid model on the ICU dataset

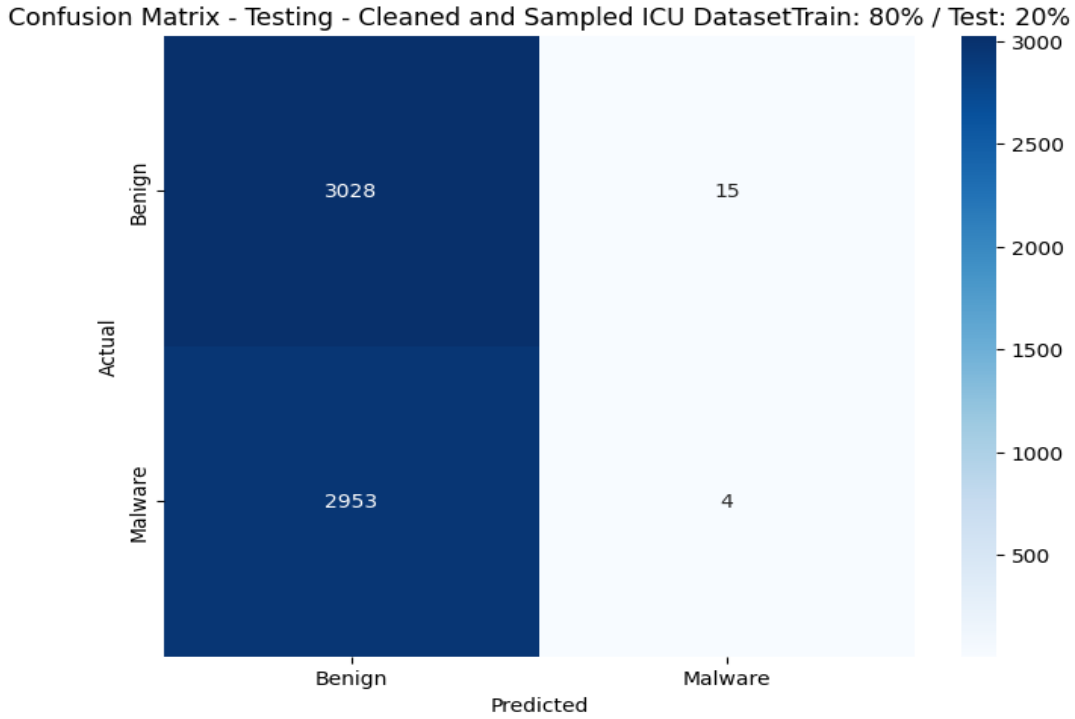


Figure 4.15: Confusion matrix for three hidden layers' 80/20 hybrid model on the ICU dataset

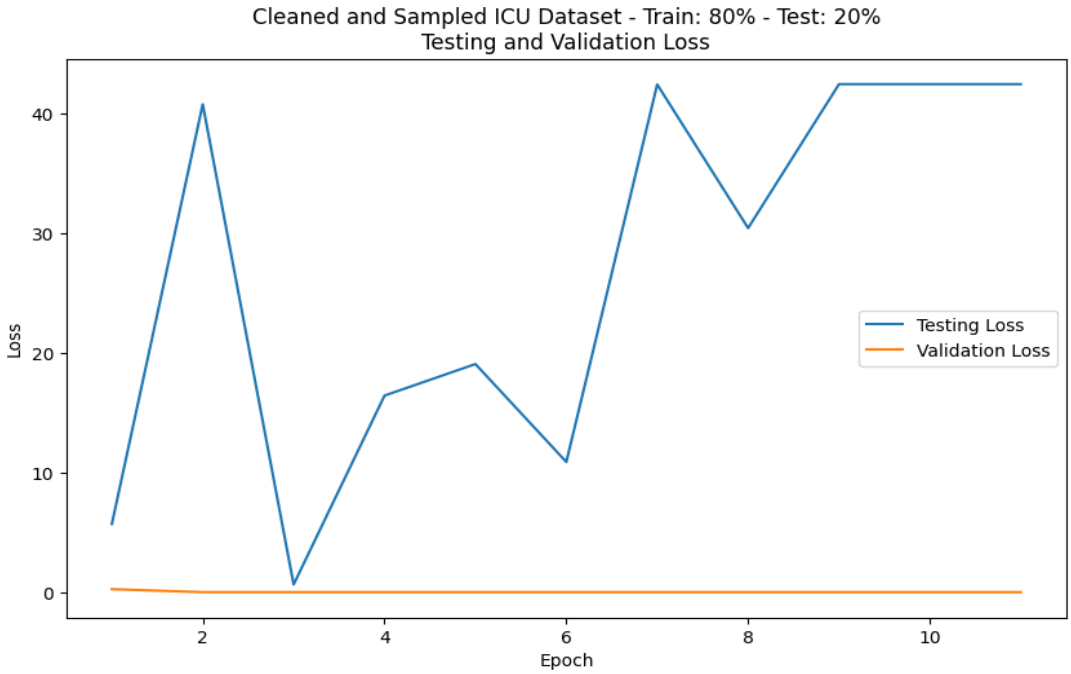


Figure 4.16: Testing and validation loss curves for three hidden layers' 80/20 hybrid model on the ICU dataset

The analysis of ROC, confusion matrices, and loss curves for the 60/40 model with seven hidden layers provides insights into its performance. For instance, the model had an AUC score of 0.5, implying that it was non-discriminant in classifying threats. This low score is reflected in the model's confusion matrix, which shows 2,957 false negatives. The testing loss curve for the model started at zero and rose to 20 in the second epoch, then slightly stabilized at 30 in the third and fourth epochs, before falling to 0 in the sixth epoch. This behavior demonstrated the model's unstable performance across the testing epochs. The model's validation loss remained at zero until the end of the test session. These results are shown in Figures 4.17, 4.18, and 4.19.

Analysis of training loss curves for these selected models offers insights into their performance behavior. For instance, the training loss for the eight-hidden-layer' 80/20 variant started at five and dropped to one in the second epoch. This value wobbled between one and three until the end of the training session in the ninth epoch. The training validation loss started at nine and dropped to about 0.5 in the third epoch, then increased to 4 in the eighth epoch. This situation indicates that the model's training did not stabilize. The training loss curve for the 80/20 variant with three hidden layers started at 0.25 and declined to 0 in the third epoch, stabilizing at this value until the end

of training. The model's training-validation loss showed similar behavior, suggesting that additional training epochs would not have improved its performance. On the other hand, the training loss curve for the 60/40 variant with seven hidden layers started at eight and fell to 2 in the second epoch, while the training-validation loss began at zero and increased to 1 in the second epoch. The training loss and validation curves wobbled in the remaining epochs, suggesting that additional training might have improved the model's performance. These results are shown in Figures 4.20, 4.21, and 4.22.

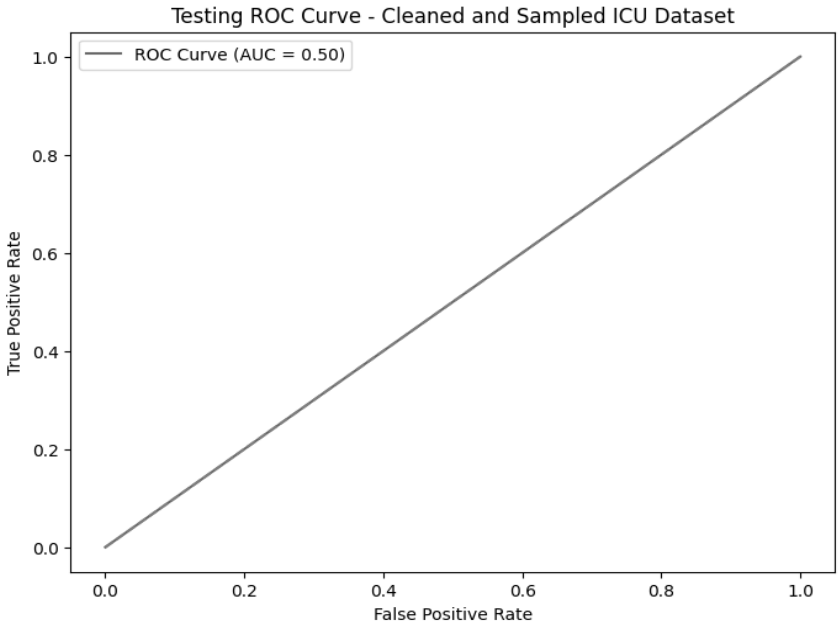


Figure 4.17: Receiver Operating Characteristics curve for the seven hidden layers' 60/40 hybrid model on the ICU dataset

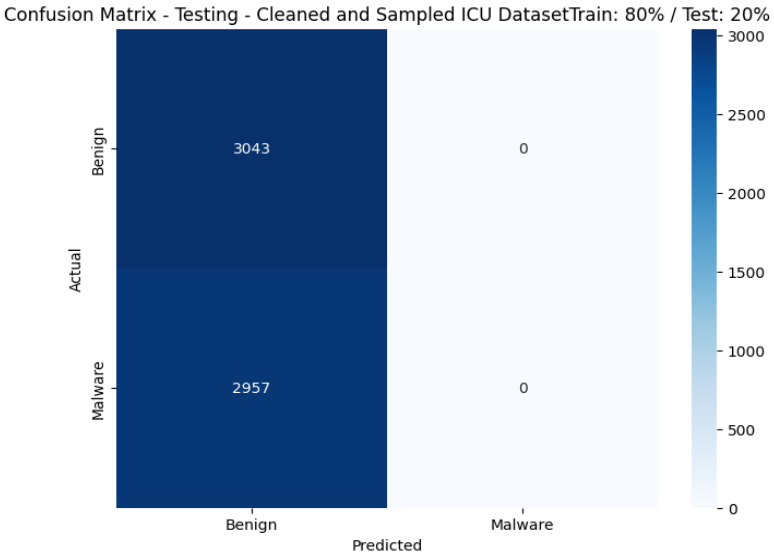


Figure 4.18: Confusion matrix for seven hidden layers' 60/40 hybrid model on the ICU dataset

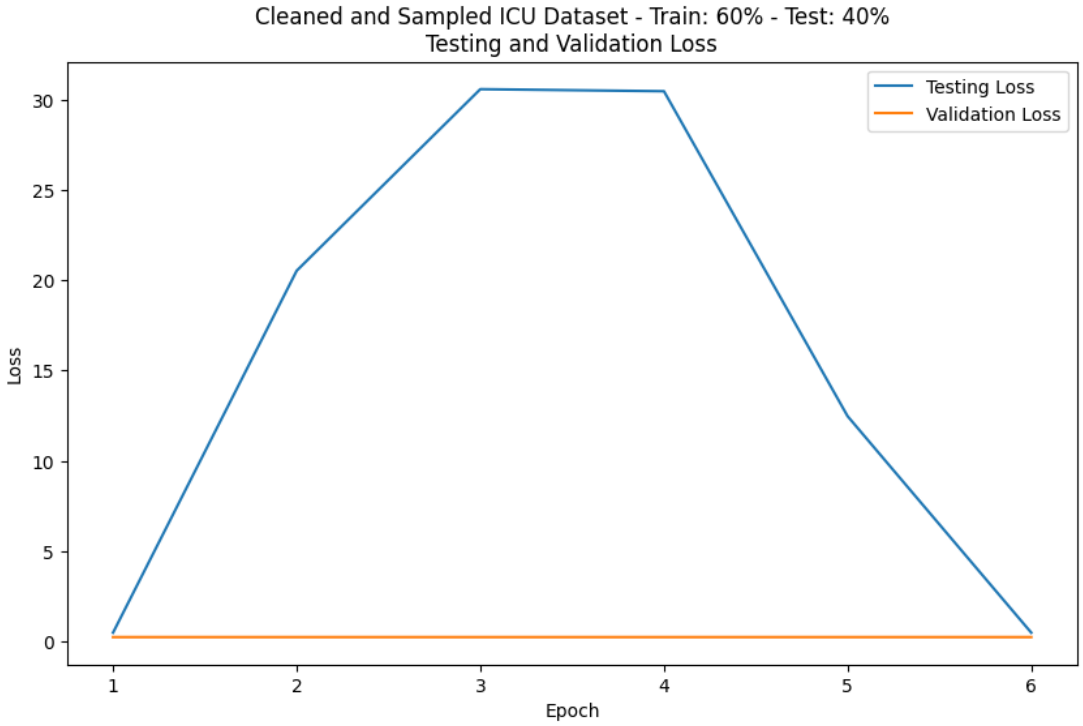


Figure 4.19: Testing and validation loss curves for the seven hidden layers' 60/40 hybrid model on the ICU dataset

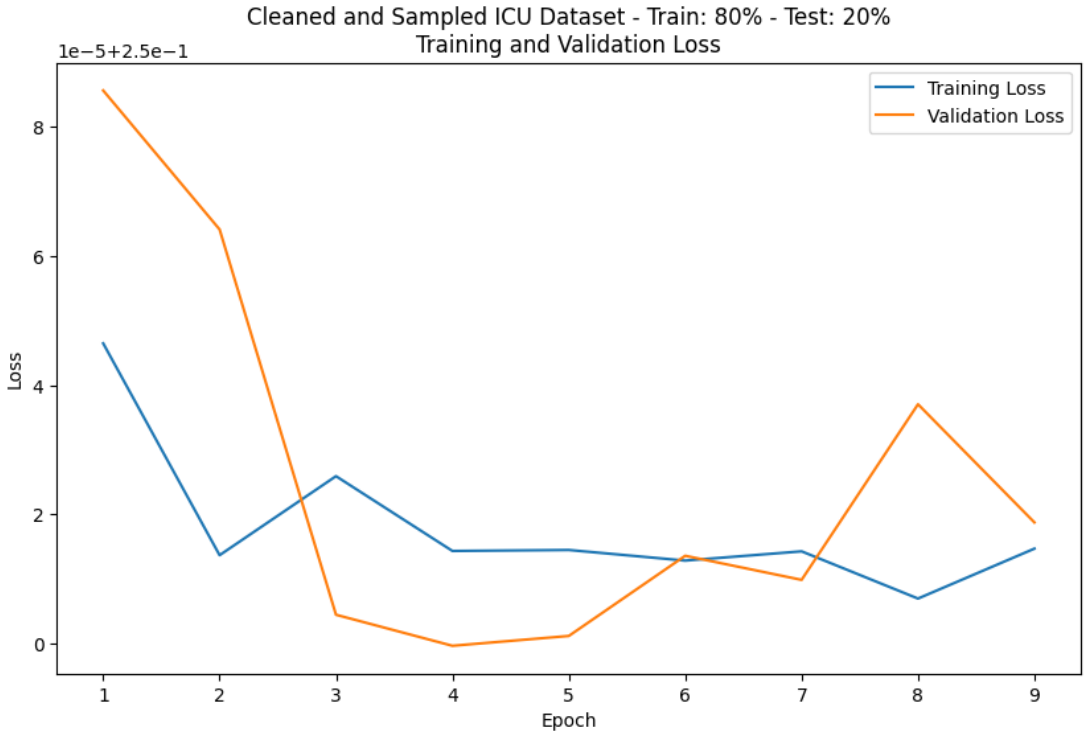


Figure 4.20: Training and validation loss curves for eight-hidden-layers' 80/20 hybrid model on the ICU dataset

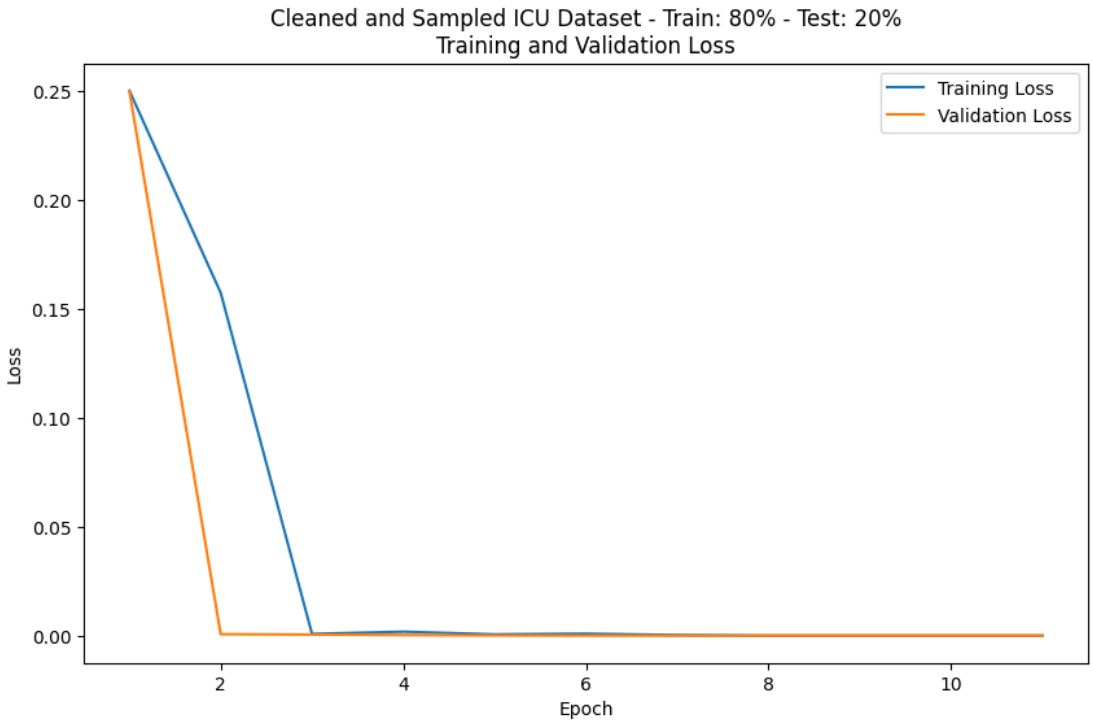


Figure 4.21: Training and validation loss curves for three-hidden-layer 80/20 hybrid model on the ICU dataset

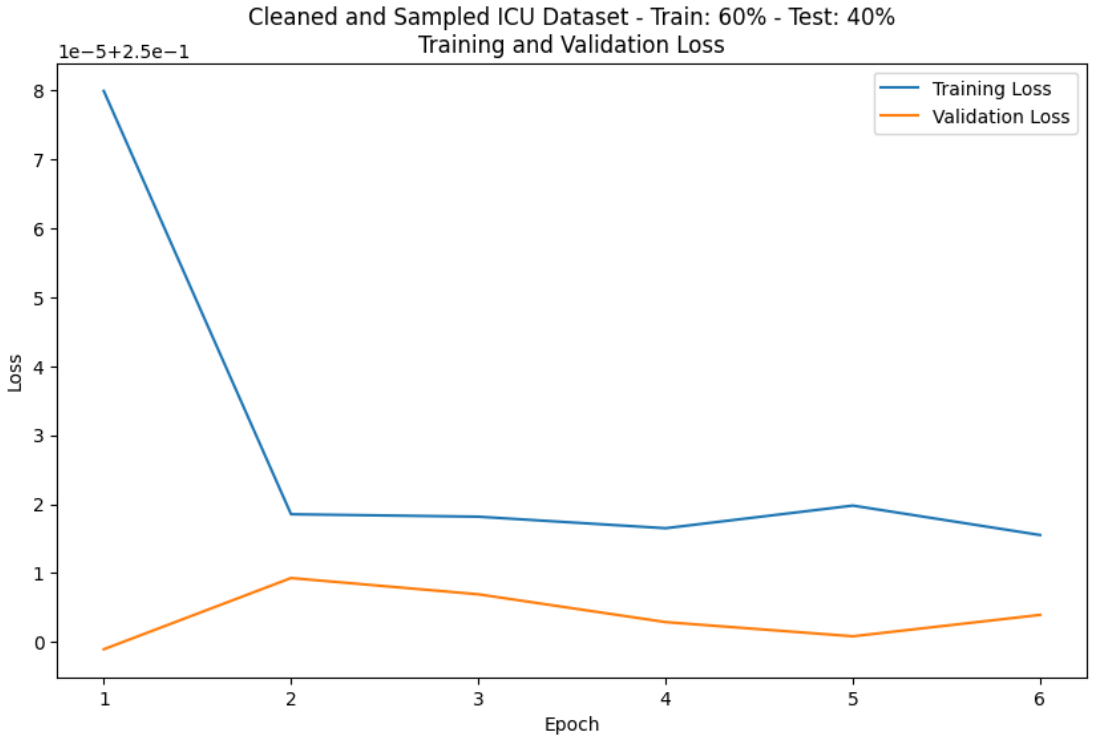


Figure 4.22: Training and validation loss curves for seven hidden layers' 60/40 hybrid model on the ICU dataset

4.3.2 Hybrid Model on the WUSTL Dataset

The analysis of recall scores for the WUSTL dataset across the baseline and hybrid models shows insights into the models' sensitivity. The eight-hidden-layers 70/30 variant emerged as the best model, with a score of 50.73%, followed by the seven-hidden-layers 60/40 variant for the hybrid model. The 80/20 variants for CNN and LSTM scored 49.82% and 49.57%, respectively, while the 80/20 variant of the hybrid model with five hidden layers scored 49.57%. These scores demonstrate the hybrid model's ability to outperform baseline models in terms of true positive rate on the medical IoT dataset. These scores are shown in Figure 4.23.

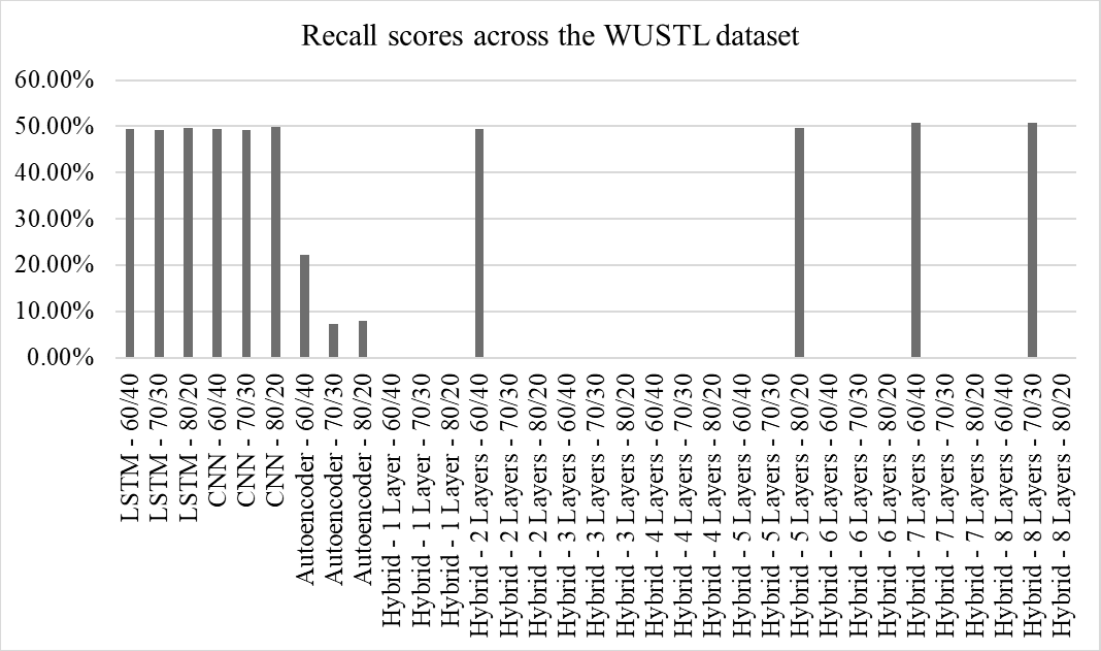


Figure 4.23: Recall scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and hybrid deep autoencoder model variants

The assessment of precision scores for the WUSTL dataset provides insights into the positive predictive value for the baseline and hybrid models. The baseline models rank in the top five for this metric. For instance, the autoencoder's 80/20 variant leads at 69.16%, followed by the CNN's 80/20 and 70/30 variants at 58.25% and 44.54%, respectively. The autoencoder's 70/30 and 60/40 variants follow at 43.24% and 39.99%. The best hybrid model in this metric is the 70/30 variant with eight hidden layers, which scores 25.74%. This score is substantially lower compared to those obtained from the baseline models. These results are shown in Figure 4.24.

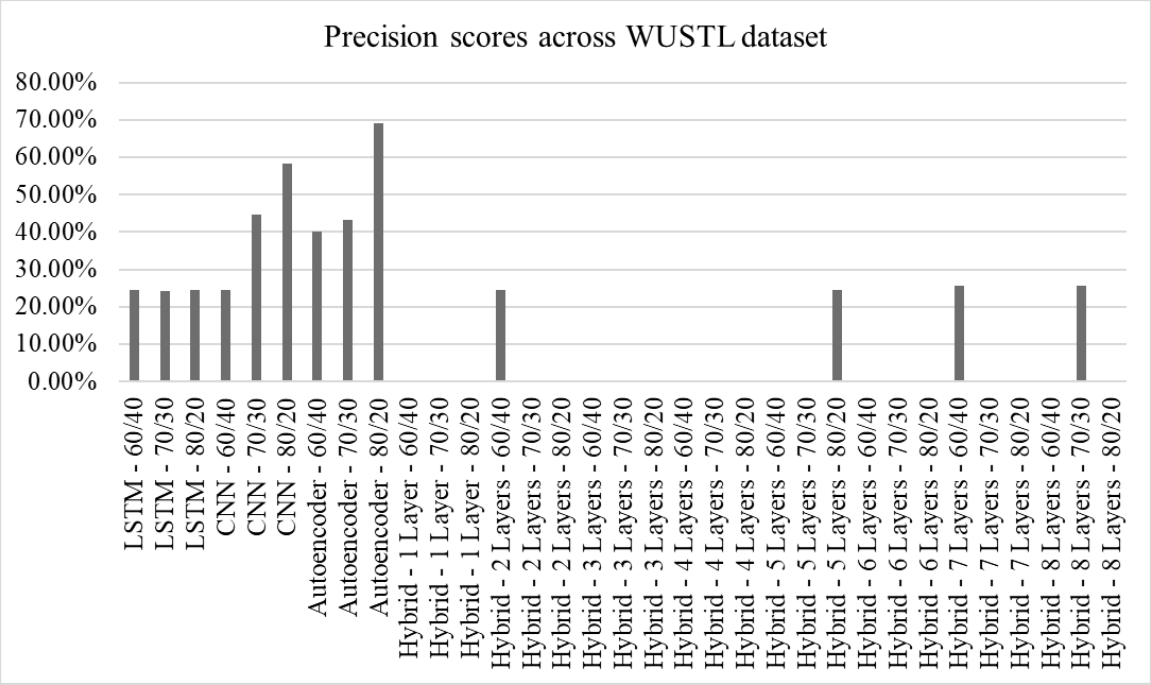


Figure 4.24: Precision scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and hybrid deep autoencoder model variants

The review of specificity scores for the baseline and hybrid models on the WUSTL dataset offers insights into their true negative rate. The autoencoder model emerges as the best, with the 80/20 variant scoring 90.47%, the 70/30 variant scoring 89.78%, and the 60/40 variant scoring 88.55%. The hybrid models closely follow these models, with one hidden layer's 60/40 and two-hidden-layers' 70/30 variants scoring 75.00%. Numerous other hybrid models score 66.67%, while the remaining ones score 50.00%. Only the CNN's 70/30 variant scores below 50%. These scores are shown in Figure 4.25.

The review of accuracy scores for the WUSTL dataset across the baseline and hybrid models demonstrates the models' overall correctness. The 70/30 variant with eight hidden layers emerges as the best, scoring 50.73%. This model is followed by the seven hidden layers of the 60/40 hybrid variant, scoring 50.64%. The baseline and other hybrid models fail to reach 50% in their scores. This situation demonstrates that some configurations of the hybrid models can outperform the baseline models in terms of accuracy. These results are shown in Figure 4.26.

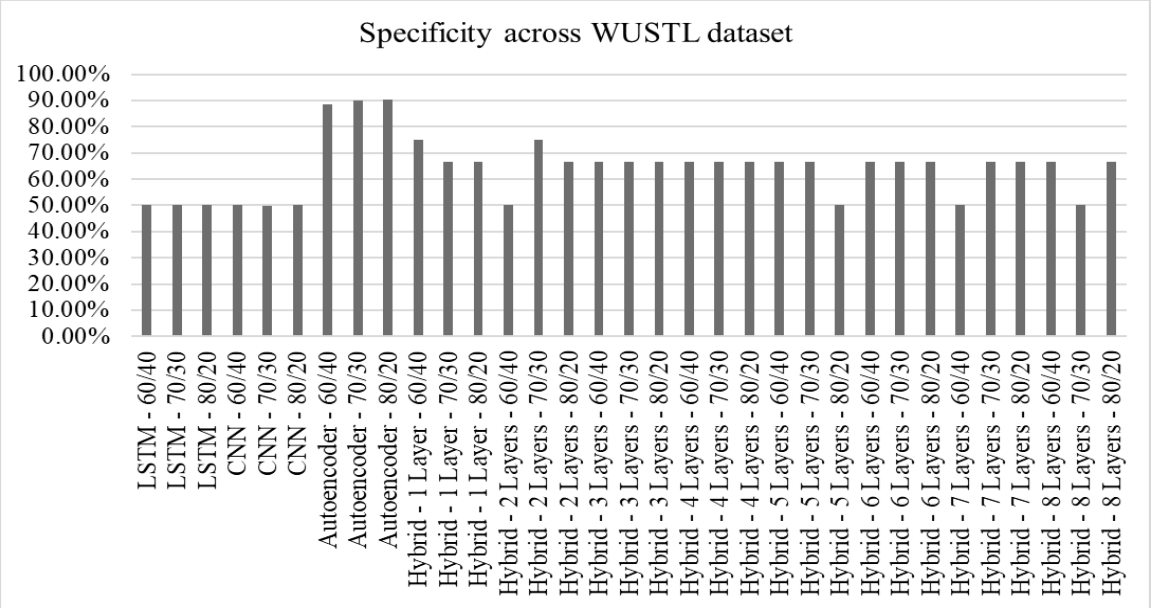


Figure 4.25: Specificity scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

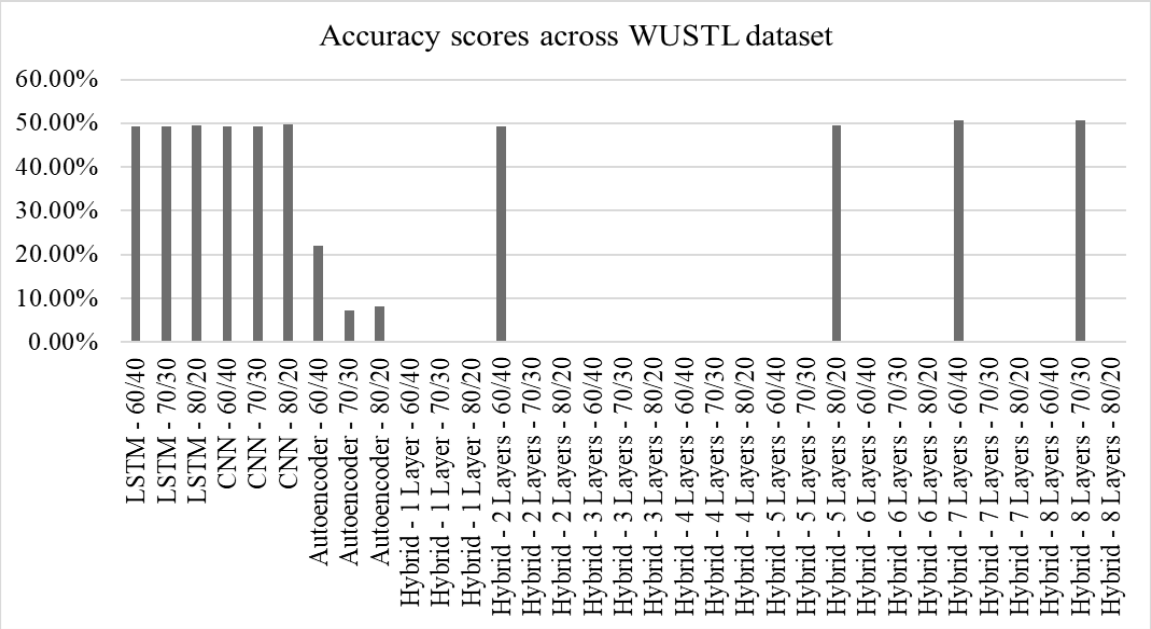


Figure 4.26: Accuracy scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

The assessment of F1 scores for the WUSTL dataset across the baseline and hybrid models shows how these models balance recall and precision. The 70/30 8-hidden-layer variant and the 6040 7-hidden-layer variant of the hybrid model emerged as the best, scoring 34.15% and 34.05%, respectively. These hybrid models were followed by the CNN's 80/20 variant at 33.82%, LSTM's 80/20 variant at 32.86% and the five-hidden-layers' 80/20 variant of the hybrid model at 32.86%. While most of the remaining

hybrid models scored zero, the emergence of a few in the leading positions demonstrates the promise of this approach for addressing threats to the IMD ecosystem. These results are shown in Figure 4.27.

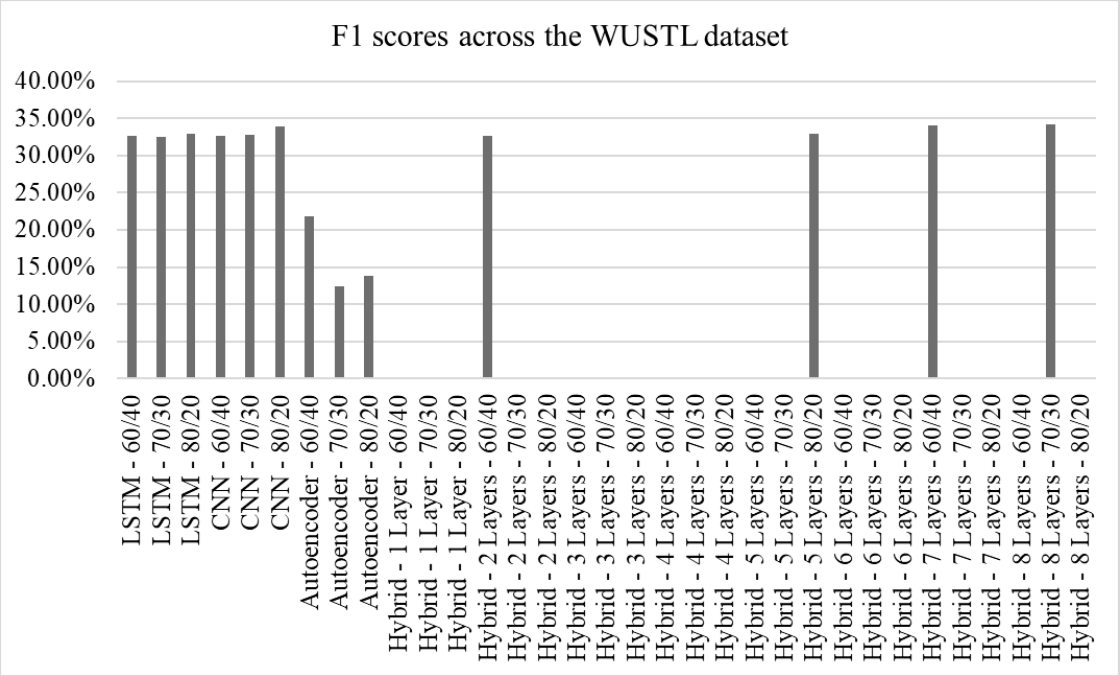


Figure 4.27: F1 scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

The analysis of the false-positive rate for the WUSTL dataset across the baseline and hybrid models shows the relative number of threats erroneously classified as benign. The autoencoder models achieve the best false-positive rates, with the 80/20 variant scoring 9.53%, the 70/30 variant scoring 10.22%, and the 60/40 variant scoring 11.45%. The 60/40 variant of the hybrid model's one-hidden-layer model closes at 25%, with 25.00%. This phenomenon demonstrates the promises of hybrid models in achieving high accuracy scores with additional model improvements. These scores are shown in Figure 4.28.

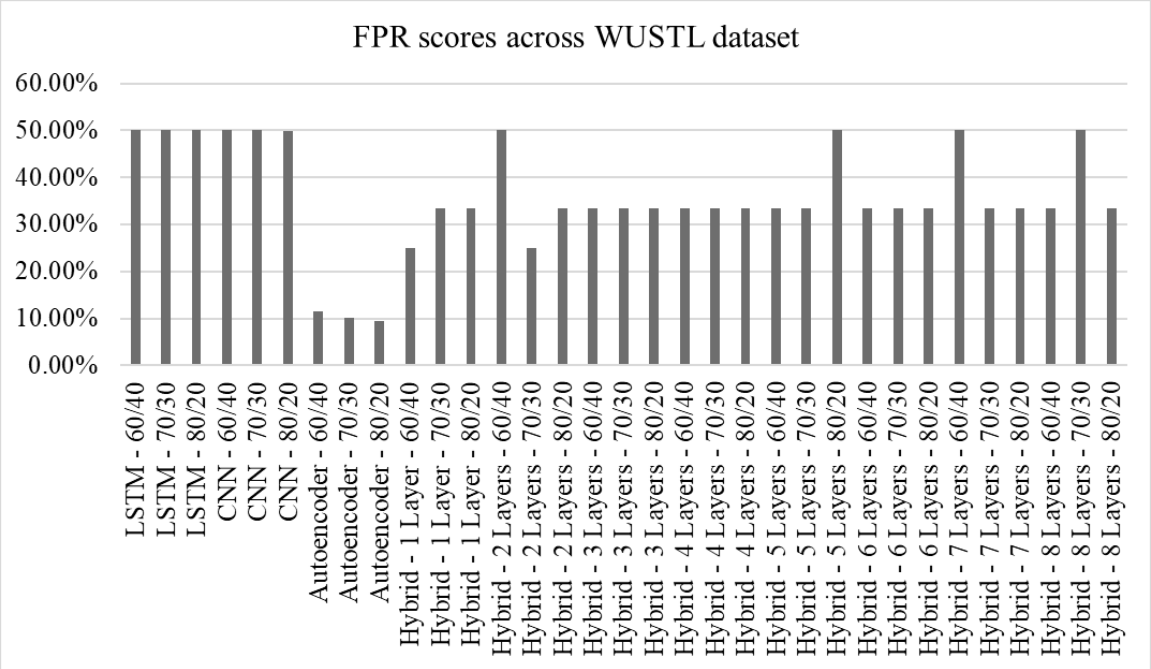


Figure 4.28: False positive rates for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

The review of inference time for the WUSTL dataset across the baseline and hybrid models shows the detection speeds for various models. The LSTM variants emerge as the fastest at detecting threats, scoring 0.19 seconds for 70/30, 0.21 seconds for 80/20, and 0.21 seconds for 60/40. The 80/20 variant of the hybrid model's one-hidden-layer follows next, with a speed of 0.25 seconds. This phenomenon demonstrates the promise of hybrid models for achieving reasonable speeds despite the high number of layers in their architectures. These results are shown in Figure 4.29.

The analysis of confusion matrices, Receiver Operating Characteristics, testing, and validation loss curves for the selected models in the WUSTL dataset provides insight into their performance, enhancing understanding. These models provide a general overview of the hybrid deep autoencoder's performance on the Medical IoT dataset. The selected models are eight-hidden-layers' 70/30, seven hidden layers' 60/40, five-hidden-layers' 80/20, and two-hidden-layers' 60/40 variants. The eight-hidden-layers' 70/30 variant of the model had an Area under the ROC curve of 0.5, demonstrating the model's non-discriminatory nature. This behavior is confirmed by the model's confusion matrix, which shows 605 false positives. This relatively high number of false positives relative to true positives demonstrates weaknesses in the model's ability to identify threats correctly. The review of the model's loss curves highlights what might

have led to this situation. Although the loss value for the model started at zero, increased to 5 in the second epoch, then fell to 0 in the third, it rose steeply to 70 in the fourth. The loss score dropped to 30 in the fifth epoch, rose slightly in the sixth, and then reached zero in the eighth epoch. While the model achieved a relatively stable loss score of 30 in the ninth and tenth epochs, additional training would have provided a better understanding of the model. These results are shown in Figures 4.30, 4.31, and 4.32.

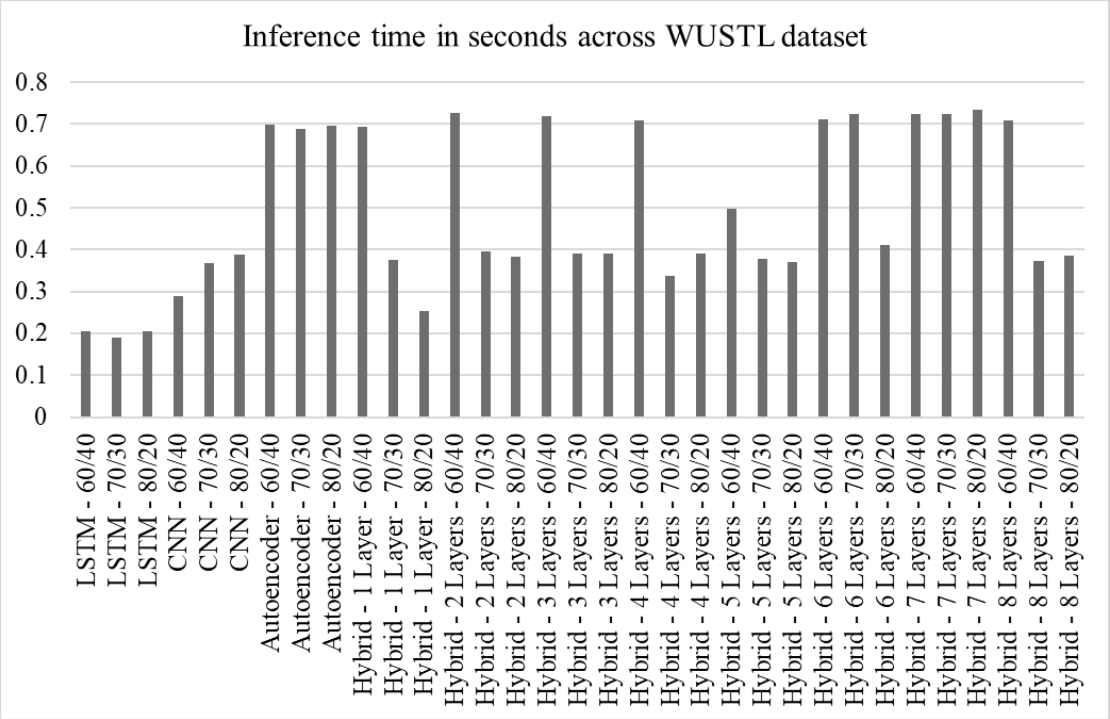


Figure 4.29: Inference time in seconds for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

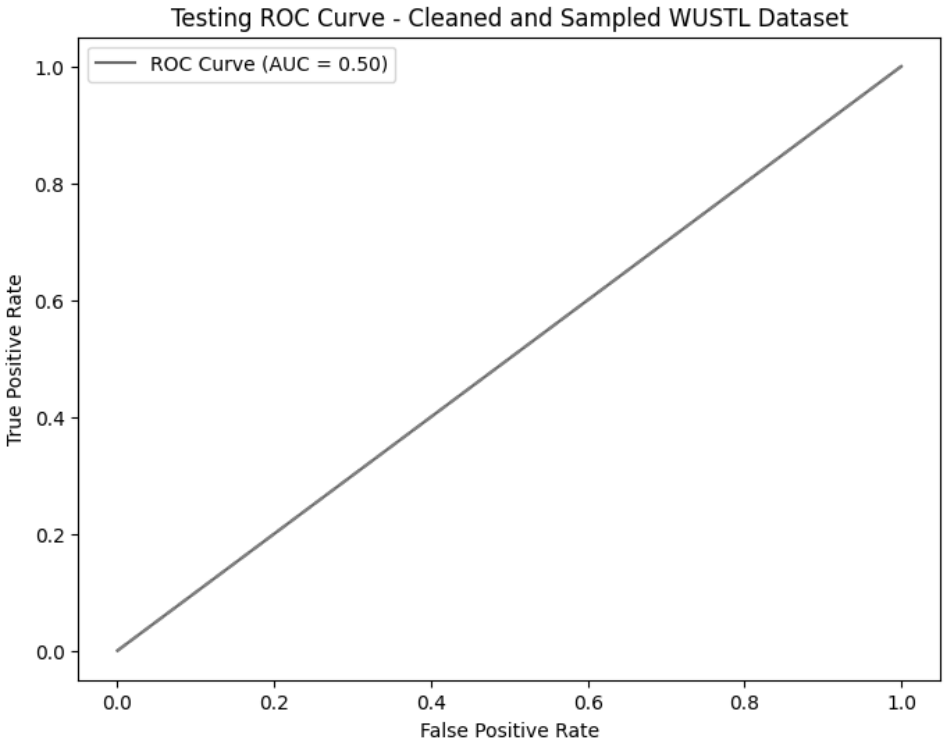


Figure 4.30: Receiver Operating Characteristics curve for an eight-hidden-layer' 70/30 hybrid model on the WUSTL dataset

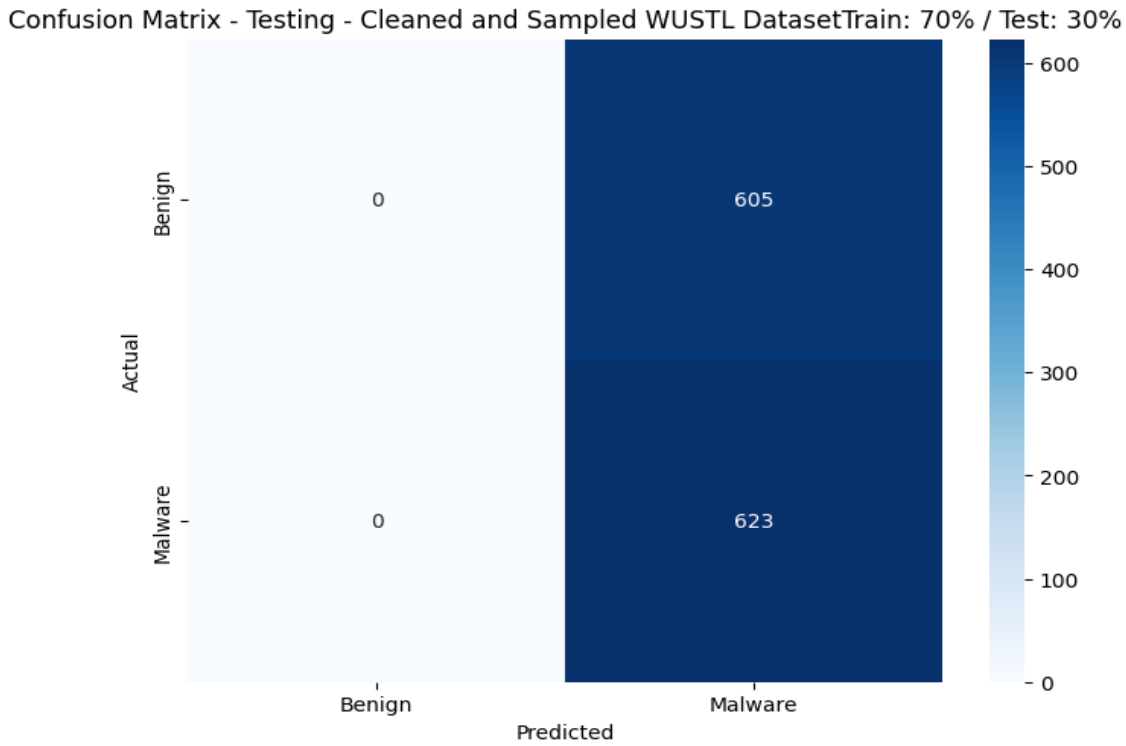


Figure 4.31: Confusion matrix for the eight-hidden-layers' 70/30 hybrid model on the WUSTL dataset

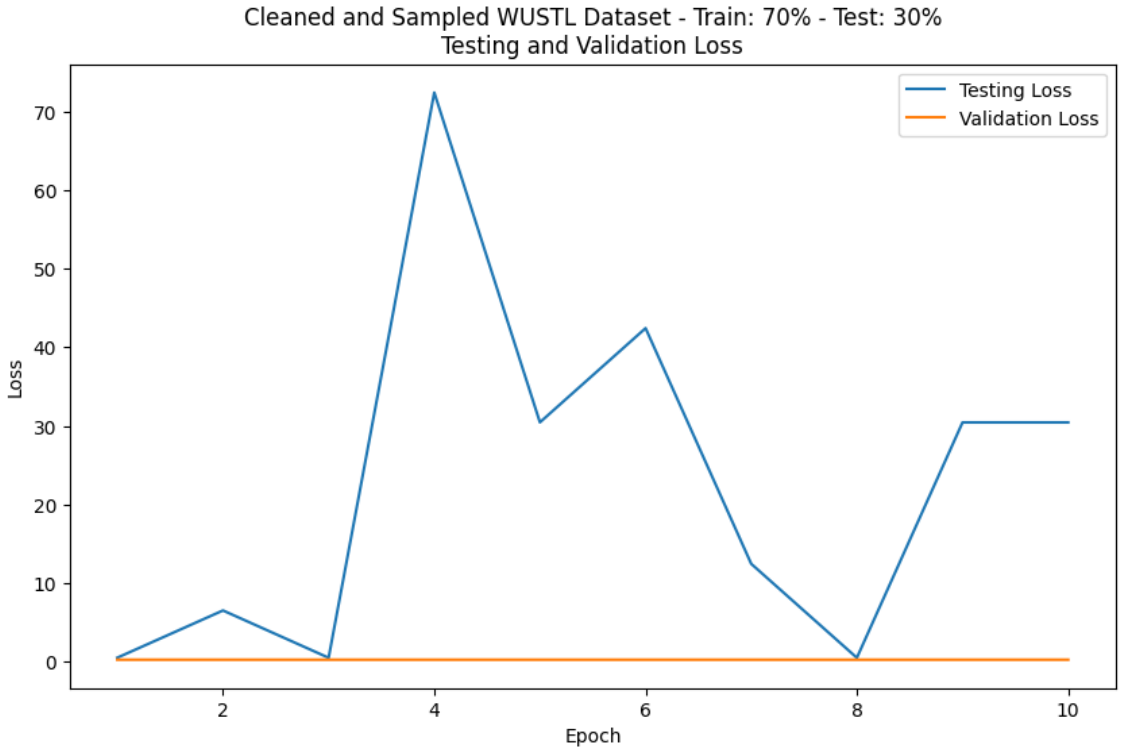


Figure 4.32: Testing and validation loss curves for the eight-hidden-layers' 70/30 hybrid model on the WUSTL dataset

The assessment of the 60/40 model across the seven hidden layers shows its weak performance. For instance, the model achieved an AUC score of 0.5, indicating its non-discriminatory behavior when classifying threats. The model had a false-positive score of 808, which was relatively high given the true positive rate of 829. The model's testing and validation loss curves offer insights into the reason behind this weak performance. The testing loss started at zero but quickly rose to 20 in the second epoch and to 30 in the third. The model briefly stabilized before falling to 10 in the fifth epoch, and finally reaching zero in the sixth epoch. Additional training might have yielded different results for the model. These results are shown in Figures 4.33, 4.34 and 4.35.

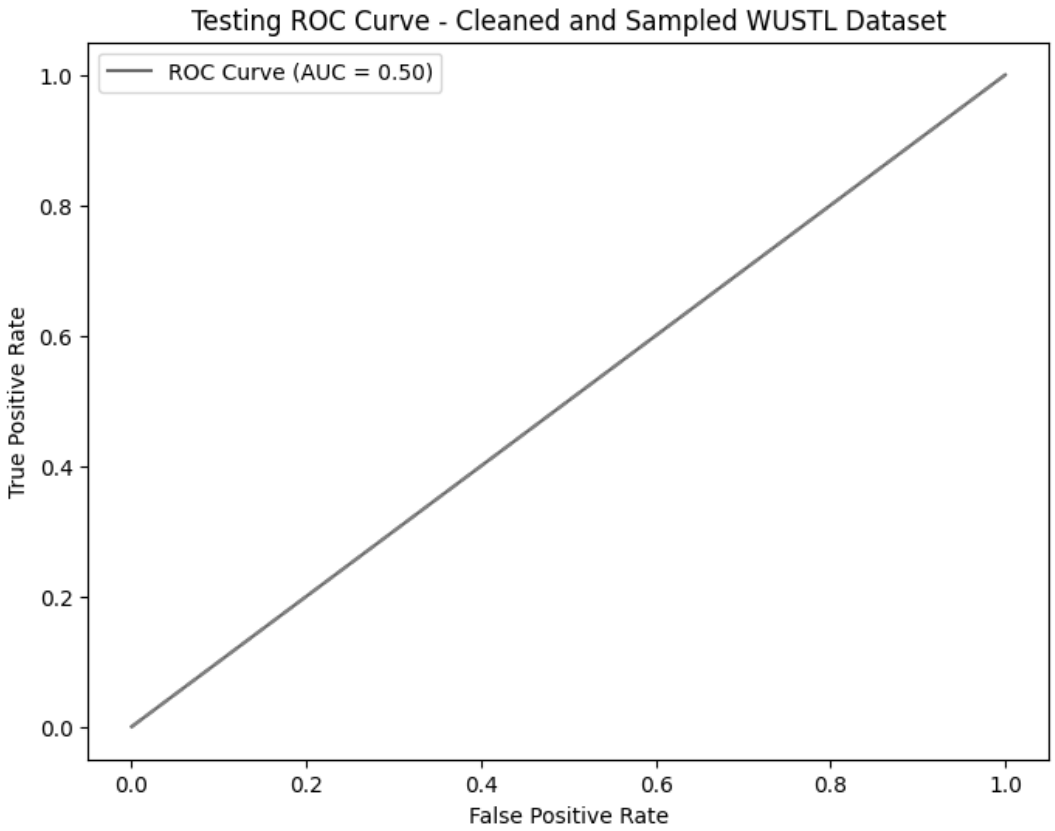


Figure 4.33: Receiver Operating Characteristics curve for the eight-hidden-layers' 60/40 hybrid model on the WUSTL dataset

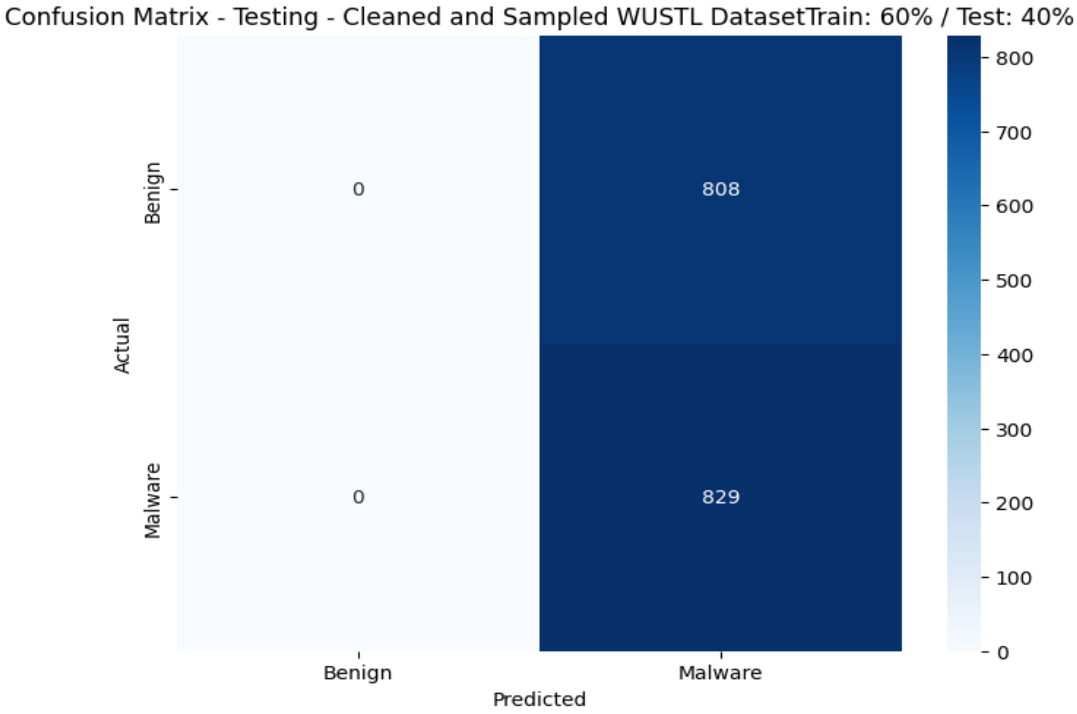


Figure 4.34: Confusion matrix for the eight-hidden-layers' 60/40 hybrid model on the WUSTL dataset

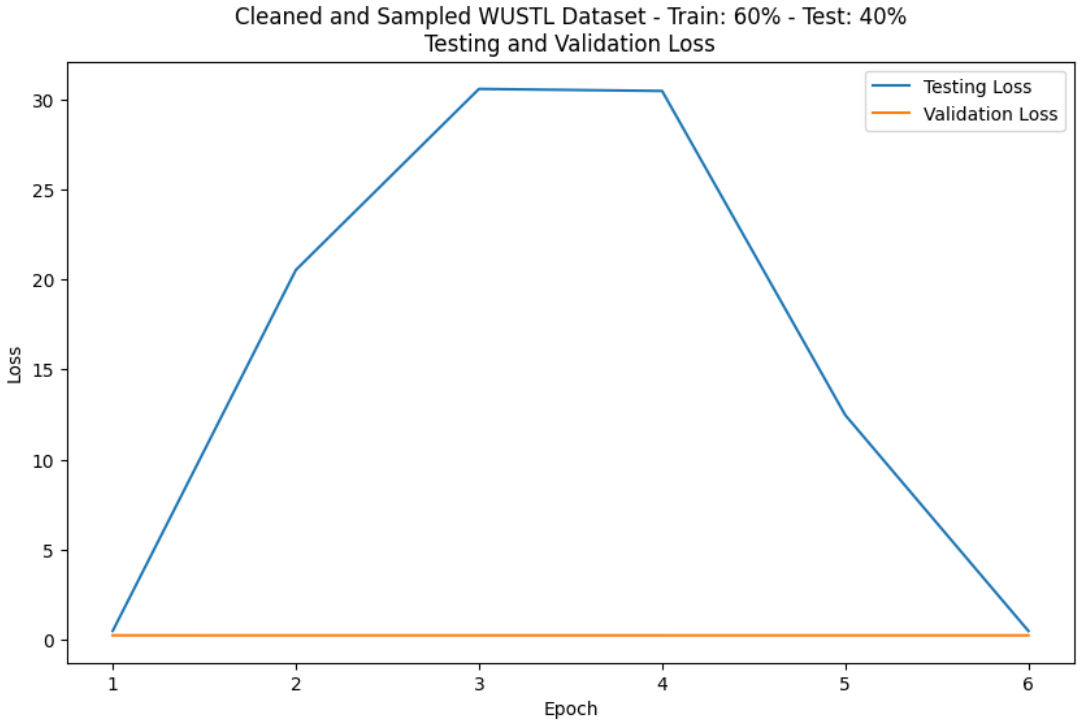


Figure 4.35: Testing and validation loss curves for the eight-hidden-layers' 60/40 hybrid model on the WUSTL dataset

The analysis of the five-hidden-layer' 80/20 variant offers insights into the hybrid model for the 80/20 training and testing ratio in the WUSTL dataset. The model's ROC curve yielded an AUC of 0.5, indicating non-discriminatory performance against the threats. The review of the model's confusion matrix shows the model had a false negative score of 413, which was higher than the true negative cases at 406. The assessment of the model's testing behavior indicates that the loss started at about 45, before falling to 20. This loss value rose to 30, before falling to 5, wobbling around this value until the end of the training. The validation loss curve stabilized at zero. These results are shown in Figures 4.36, 4.37, and 4.38.

The analysis of a 60/40 two-hidden-layer model for the WUSTL dataset offers insights into the model's performance with the reduction of hidden layers from seven to two. The model's ROC curve had an AUC of 0.5, indicating non-discriminatory performance. Further, the model had 829 false negatives against 808 true negatives. This high number of false negatives implies that the model failed to prevent threats, as they were effectively classified as benign. The review of loss curves for the model demonstrates inefficient learning and performance. For instance, the model's testing loss started at

35, increased to 70 in the third epoch, then fell to 15 in the fourth epoch. It then rose to 35 in the fifth epoch, dropped to zero in the seventh epoch, and rose again to 70 in the ninth epoch. Although the validation loss curve stabilized at zero throughout the session, the test loss did not. These results are shown in Figures 4.39, 4.40, and 4.41.

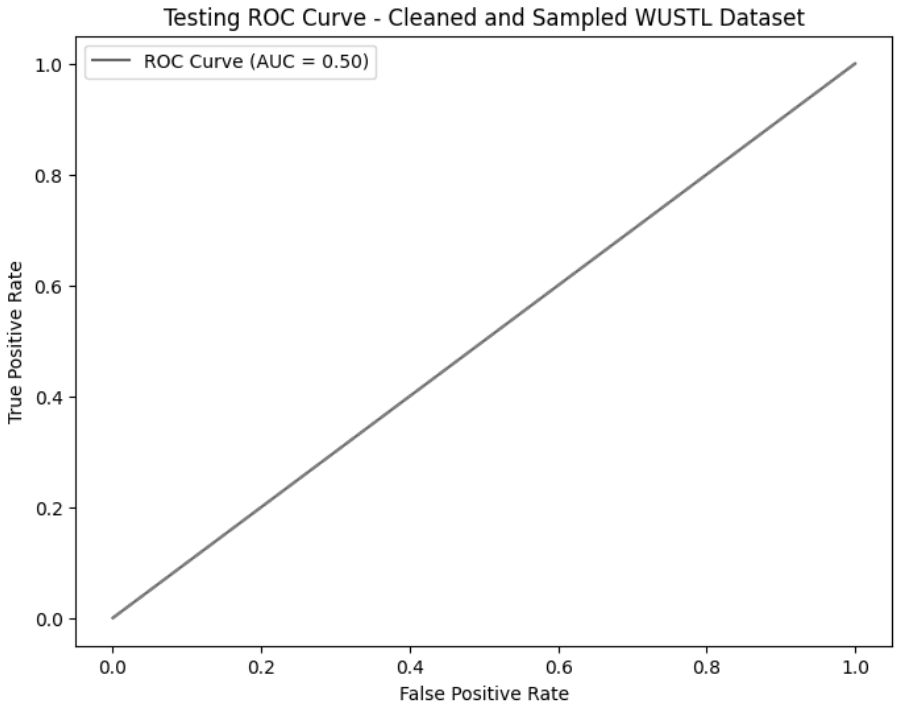


Figure 4.36: Receiver Operating Characteristics curve for five-hidden-layers' 80/20 hybrid model on the WUSTL dataset

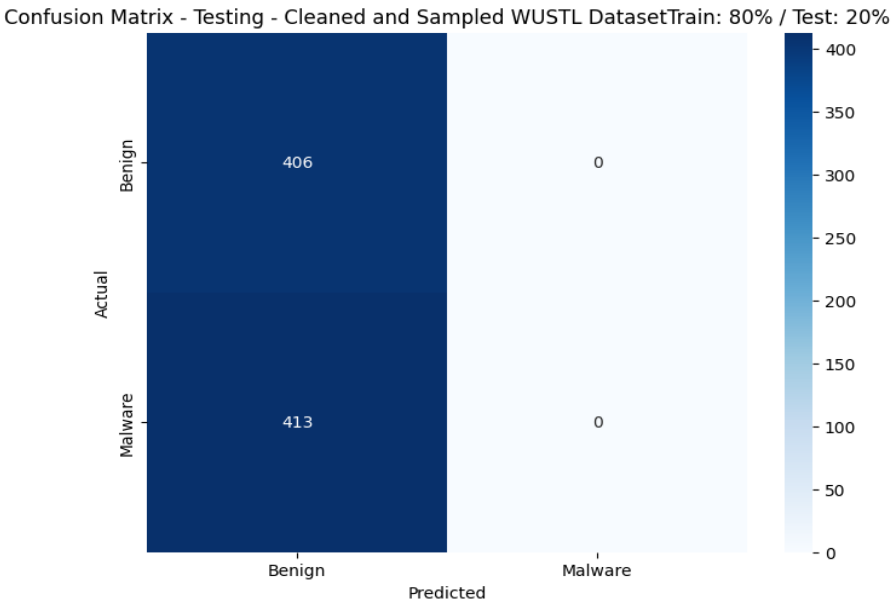


Figure 4.37: Confusion matrix for five-hidden-layers' 80/20 hybrid model on WUSTL dataset

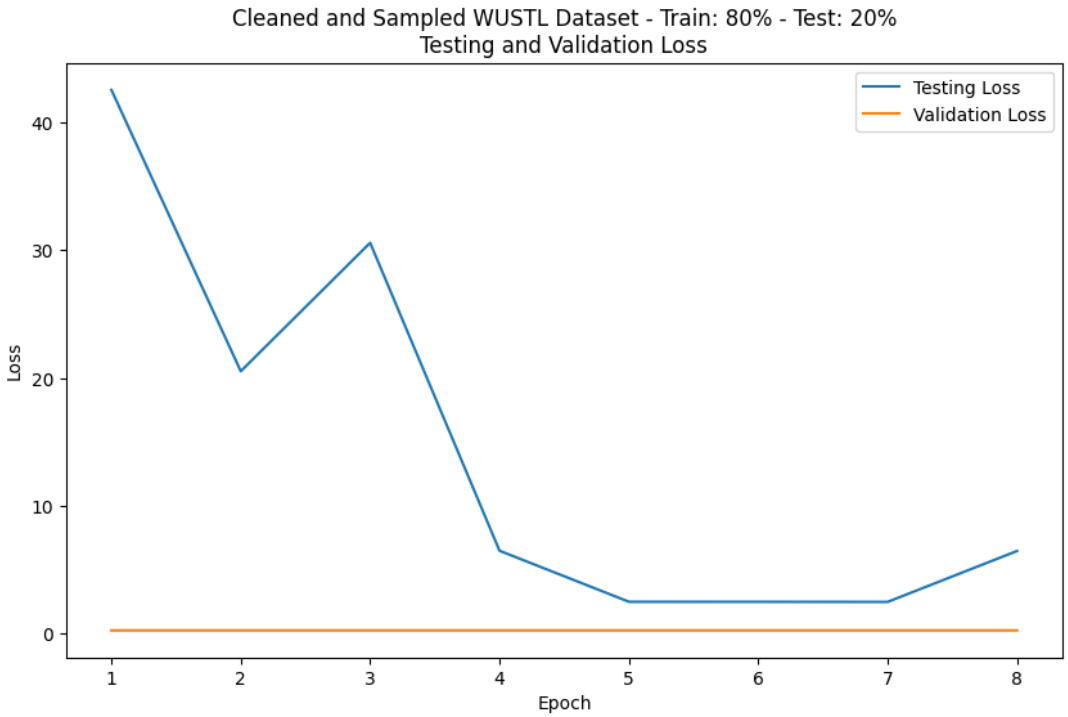


Figure 4.38: Testing and validation loss curves for five-hidden-layers' 80/20 hybrid model on the WUSTL dataset

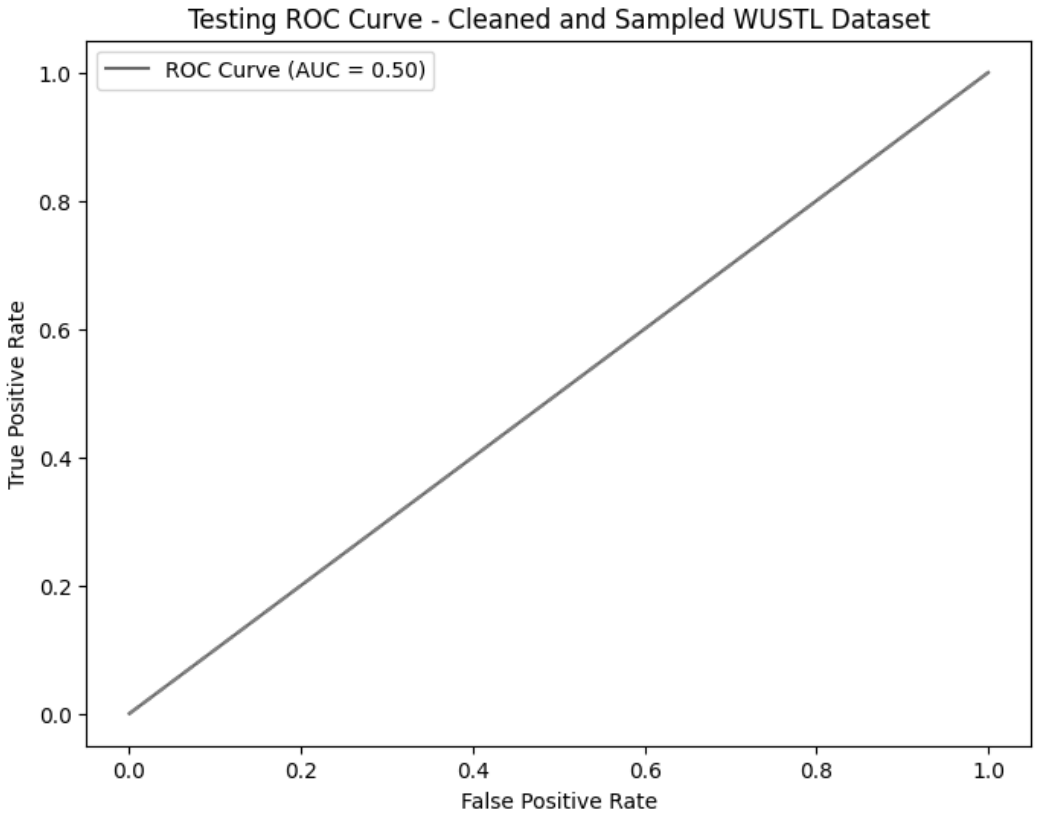


Figure 4.39: Receiver Operating Characteristics curve for a two-hidden-layer 60/40 hybrid model on the WUSTL dataset

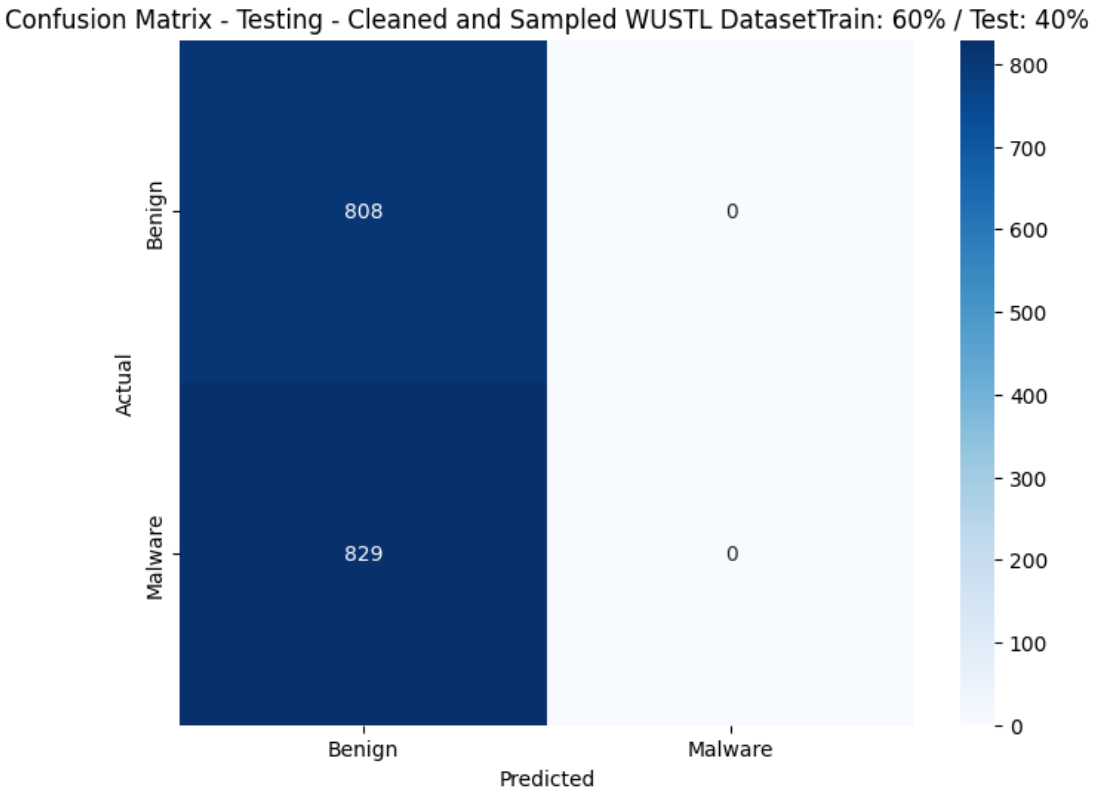


Figure 4.40: Confusion matrix for the two-hidden-layers' 60/40 hybrid model on the WUSTL dataset

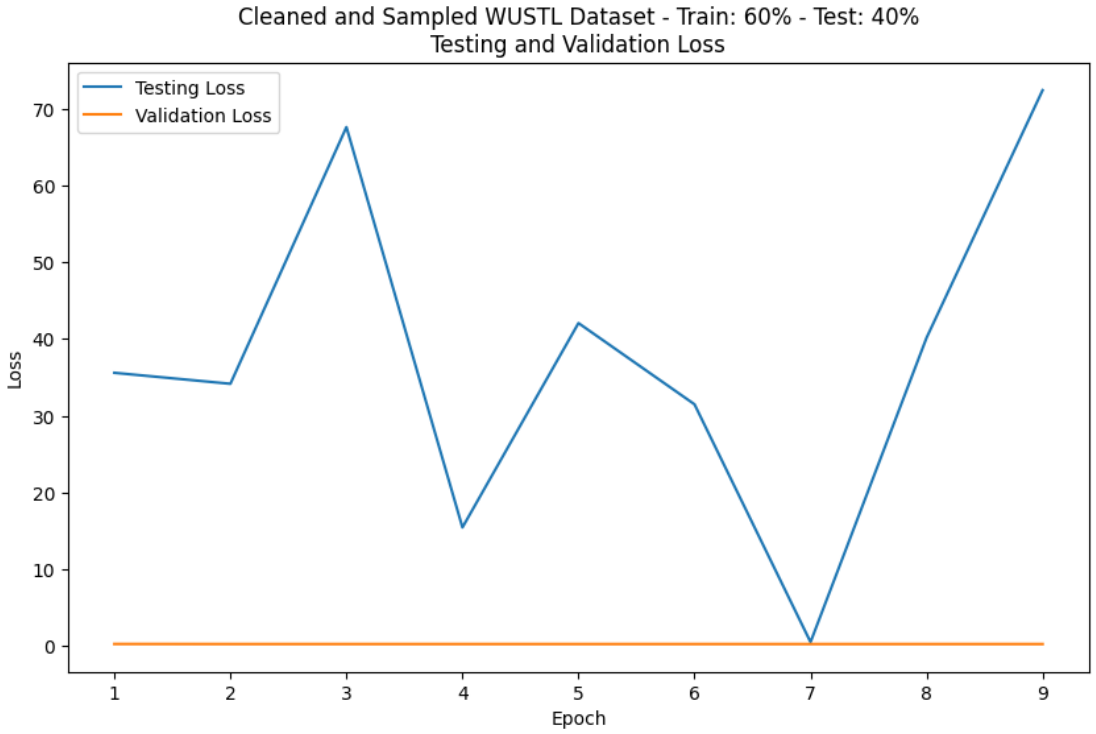


Figure 4.41: Testing and validation loss curves for a two-hidden-layer 60/40 hybrid model on the WUSTL dataset

The analysis of training loss curves for the selected models offers insights into their training behavior, enhancing understanding of the testing performance. For instance, the eight-hidden-layers' 70/30 variant started at 0.2507, before falling to 0.25005 in the second epoch. This score wobbled at this level until the end of the training sessions, whereas that of the test validation remained at 0.25 throughout the training session. This suggests that the model might show negligible improvement with additional training. The 60/40 model with seven hidden layers started at 0.2508 before falling to 0.25 in the second epoch. This value wobbled until the fifth epoch, then slightly stabilized in the sixth. The model's validation loss started slightly above 0.2502 and increased to 0.2504 in the second epoch. This value declined to 0.2501 in the third epoch and continued to decrease until the end of the training session. This situation implies that additional training epochs would have stabilized the model, but would not have yielded significantly different results.

The loss curve for the five-hidden-layers' 80/20 model started at 0.25029 and reached 0.25005 in the third epoch, before wobbling at this range until the end of the training session. The validation loss curve started at 0.25005, dropped to 0.25 in the third epoch, and wobbled through the rest of the training session. While this wobbling was clearly visible in the validation loss curve, the values were still significantly low. This phenomenon implies that the model would not have improved considerably even with additional training epochs. The training loss curve for the two-hidden-layer 60/40 variant started at 0.259, before falling to 0.251 in the second epoch. The score further declined to 0.25 in the third epoch, then rose slightly and stabilized at 0.25 in the fifth epoch. However, the validation loss fluctuated throughout the session, ranging from 0.25 to 0.2501. This situation implies that even with additional training, the model would not achieve better performance as the training loss had already stabilized. These results are shown in Figures 4.42, 4.43, 4.44 and 4.45.

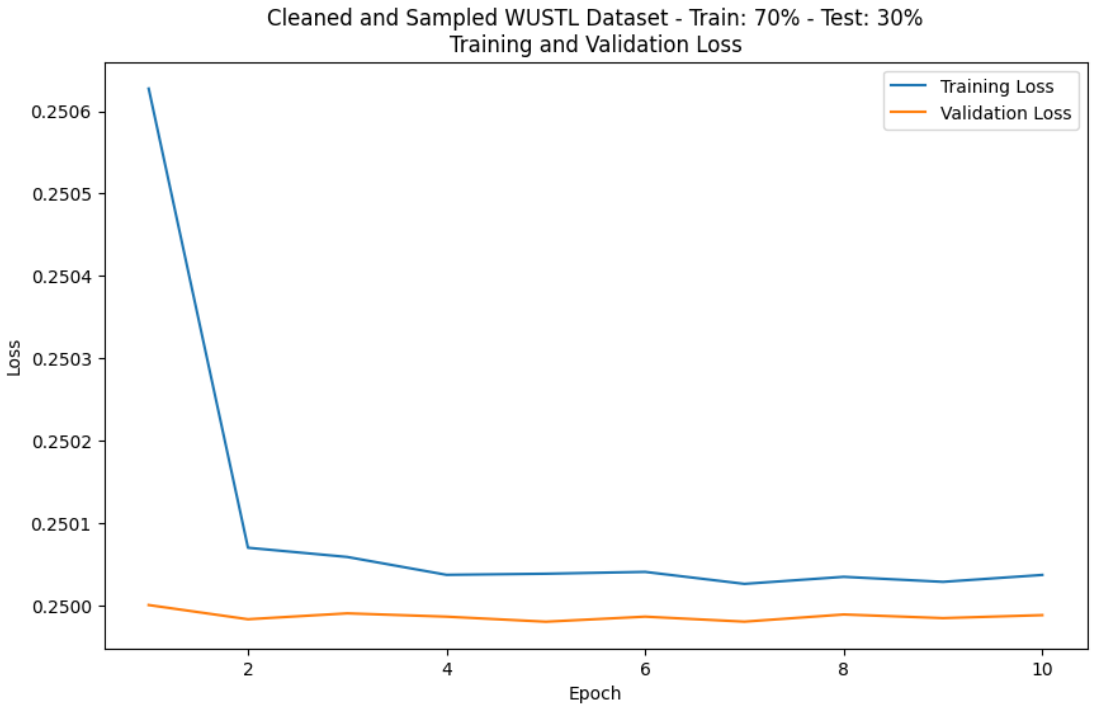


Figure 4.42: Training and validation loss curves for eight-hidden-layers' 70/30 hybrid model on the WUSTL dataset

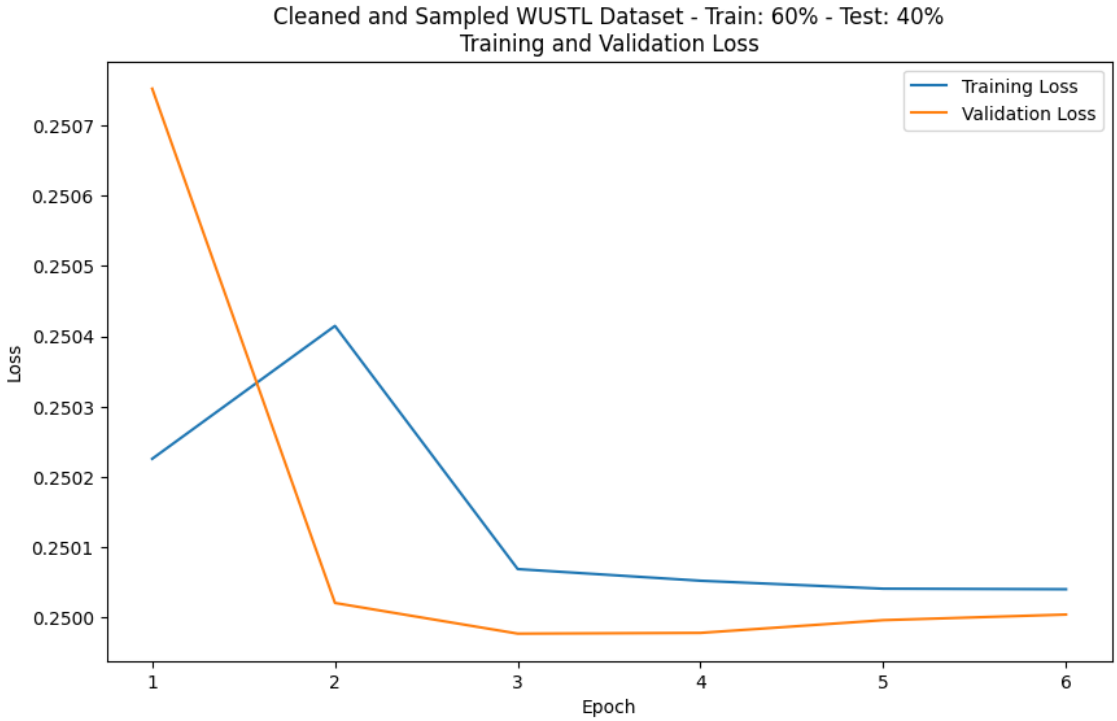


Figure 4.43: Training and validation loss curves for seven layers' 60/40 hybrid model on the WUSTL dataset

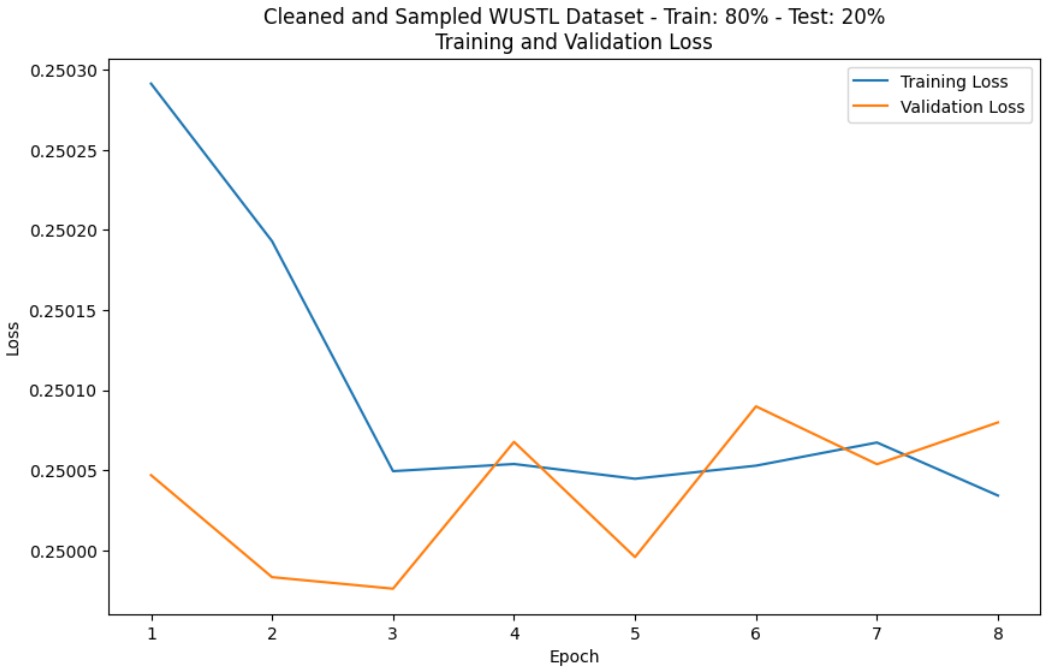


Figure 4.44: Training and validation loss curves for five-hidden-layers' 80/20 hybrid model on the WUSTL dataset

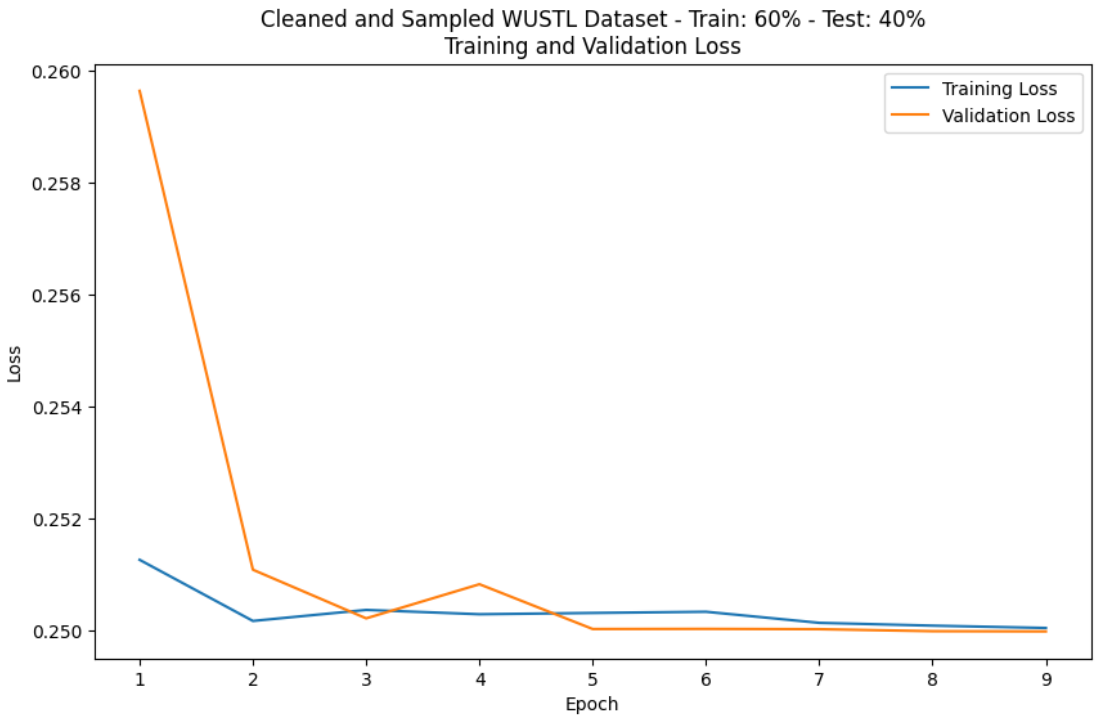


Figure 4.45: Training and validation loss curves for a two-hidden-layer' 60/40 hybrid model on the WUSTL dataset

4.3.3 Hybrid Model on Edge IIoT Dataset

The analysis of recall scores for the Edge IIoT dataset across the baseline and hybrid models shows that the models are sensitive to the general IoT dataset. The CNN's 60/40 variant emerged as the leading model with a score of 63.25%, followed by the 80/20 variant of the hybrid model's seven hidden layers at 50.39%. The LSTM models followed, with the 70/30 variant scoring 49.69%, the 60/40 variant scoring 49.68%, and the 80/20 variant scoring 49.61%. The 80/20 variant of the CNN model scored 49.61%. This situation demonstrates that the hybrid model had opportunities for improvement, enabling it to outperform other models. These results are shown in Figure 4.46.

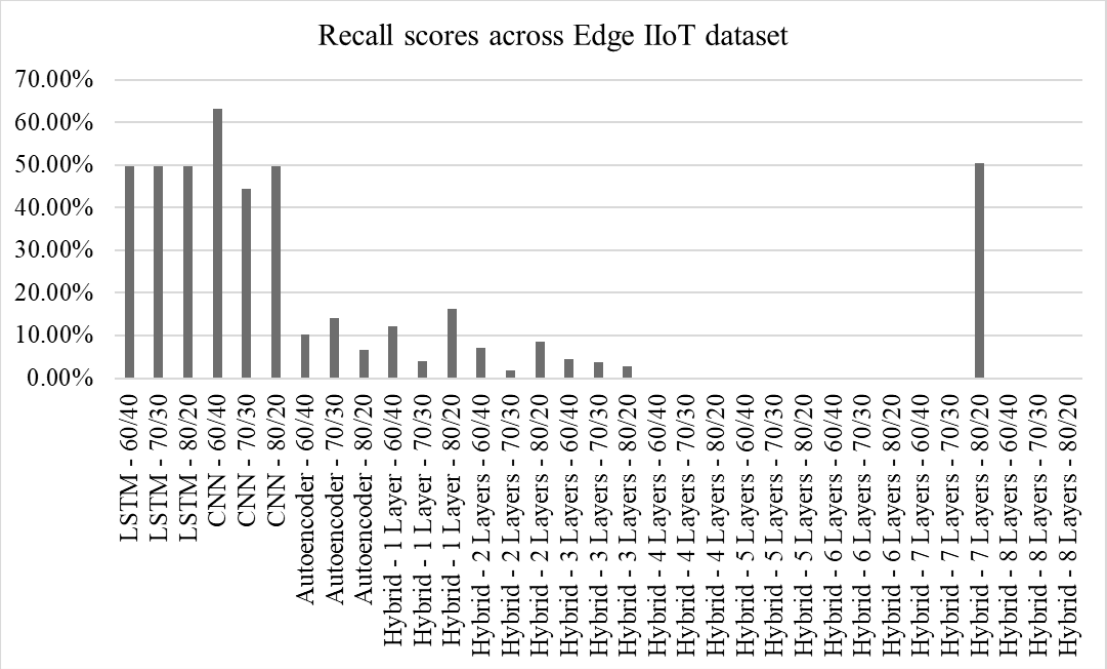


Figure 4.46: Recall scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

The analysis of precision scores for the Edge IIoT dataset on the baseline and hybrid models demonstrates the positive predictive value for these models. The 80/20 variants for the two- and one-hidden-layer hybrid models emerged as the best, with scores of 87.45% and 72.60%, respectively. The 60/40 variant of the three-hidden-layer hybrid model followed, scoring 68.01%. This phenomenon demonstrates that the hybrid model outperformed the baseline models. However, there is room for improvement, as these models did not achieve the 90% mark. These results are shown in Figure 4.47.

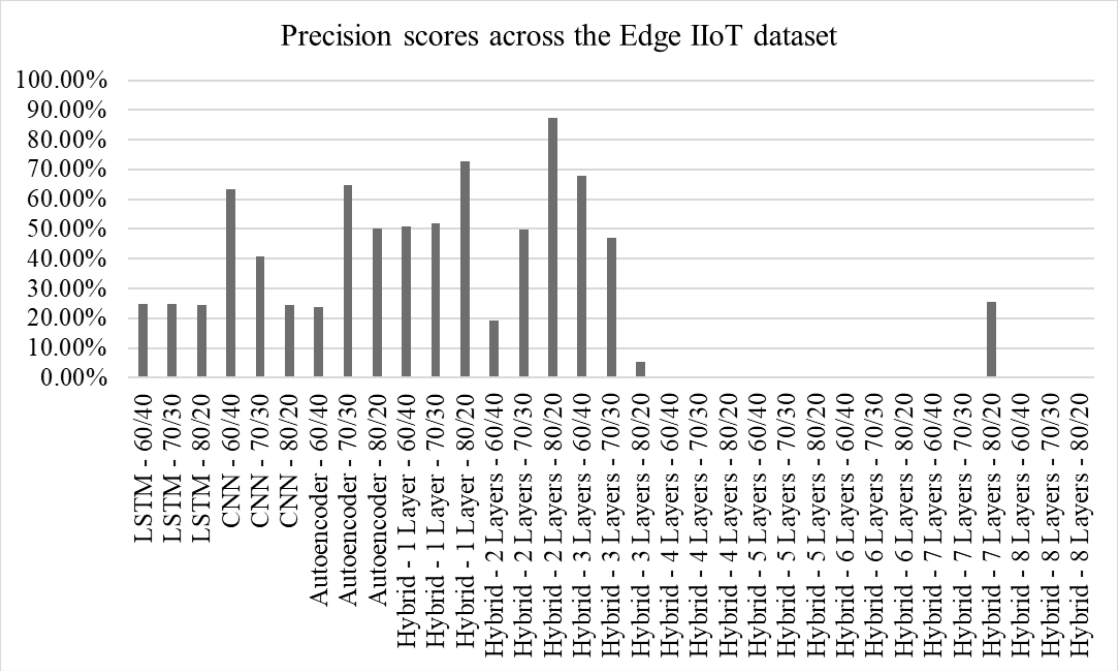


Figure 4.47: Precision scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

The assessment of specificity scores for the Edge IIoT dataset across the baseline and hybrid models shows the true negative rates of the models on the general IoT dataset. The 80/20 variant of the hybrid model with one hidden layer emerged as the best, scoring 91.04%. Although the 70/30 variant of the autoencoder achieved 90.79%, other notable hybrid models also reached 90%. For instance, the two-hidden-layers' 80/20 variant scored 90.76%, one hidden layer's 60/40 variant had 90.51%, and three hidden layers' 60/40 variant had 90.19%. This situation demonstrates that the hybrid model can achieve high specificity scores across a general IoT dataset. These scores are shown in Figure 4.48.

The review of accuracy scores for the Edge IIoT dataset across the baseline and hybrid models shows the general correctness of these models on the IoT dataset. While the CNN's 60/40 variant emerged as the best with 63.25%, the second-best model was the hybrid model's 60/40 variant with seven hidden layers at 50.39%. Other models, including the baseline models, failed to achieve the 50% mark. This situation demonstrates potential areas of improvement for the model, enabling it to outperform the baseline models. These results are shown in Figure 4.49.

The analysis of F1 scores for the Edge IIoT dataset across the baseline and hybrid models shows the balance between the recall and precision scores for these models. The CNN's 60/40 and 70/30 variants emerged as the best, with 63.25% and 38.15%, respectively, followed by the hybrid model's 80/20 variant with seven hidden layers at 33.77%. This phenomenon demonstrates the need to effectively balance precision and recall in the hybrid model, enabling it to outperform baseline models. These results are shown in Figure 4.50.

The assessment of the false-positive rate for the Edge IIoT dataset across baseline and hybrid models shows the detection efficiency of these models in handling threats. The 80/20 variant of the hybrid model with one hidden layer emerged as the best, with an FPR of 8.96%. This model was followed by a 70/30 autoencoder variant at 9.21%, a two-hidden-layer 80/20 variant at 9.24%, and a one-hidden-layer 60/40 variant at 9.49%. Other models, like the autoencoder's 80/20 and the three-hidden-layer 60/40, score 9.72% and 9.81%, respectively, enabling them to reach the below-10 % mark. While the majority of deep autoencoder models fail to reach 25%, a significant number do. This phenomenon demonstrates the promises of the autoencoder feature in achieving low FPR rates. These results are shown in Figure 4.51.

The review of inference time for the Edge IIoT dataset shows the detection speed for the baseline and hybrid models on the general IoT dataset. The CNN's 80/20 and 60/40 variants lead with 1.34 seconds and 2.61 seconds, followed by the LSTM's 80/20 and 70/30 variants at 2.62 seconds. The Hybrid deep autoencoder variant lags in detection speed, demonstrating the significant impact of hidden layers on inference time. These results are shown in Figure 4.52.

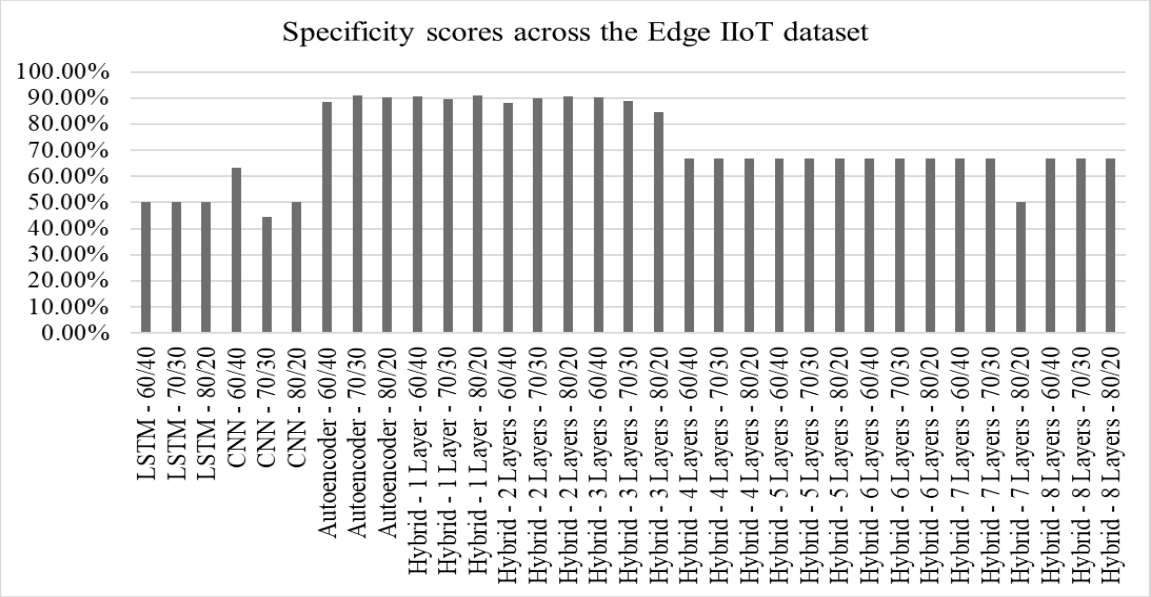


Figure 4.48: Specificity scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

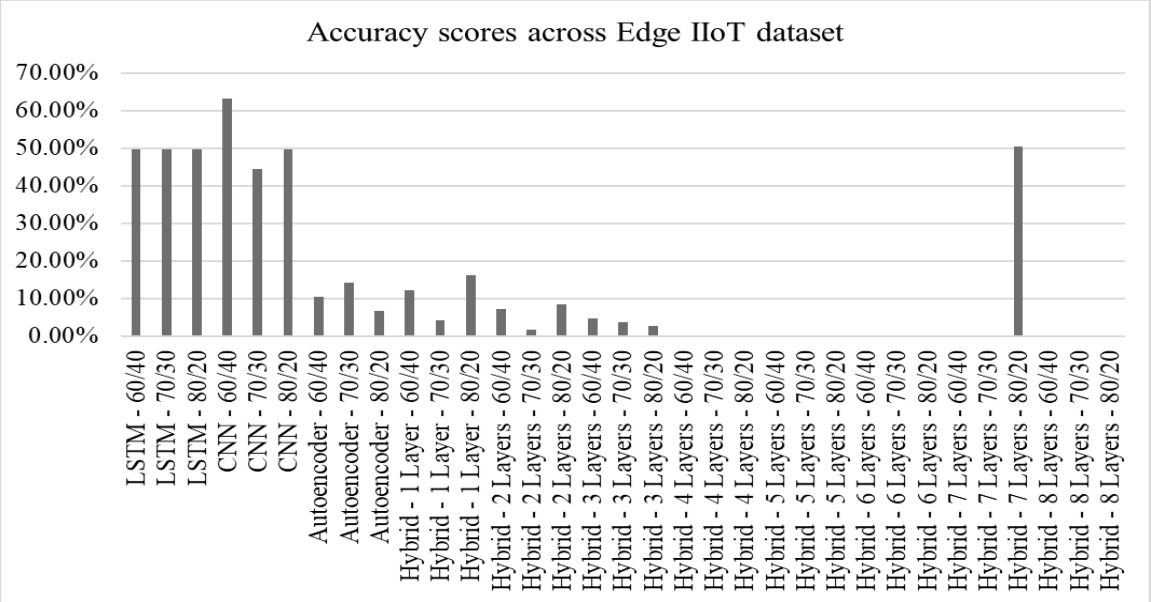


Figure 4.49: Accuracy scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

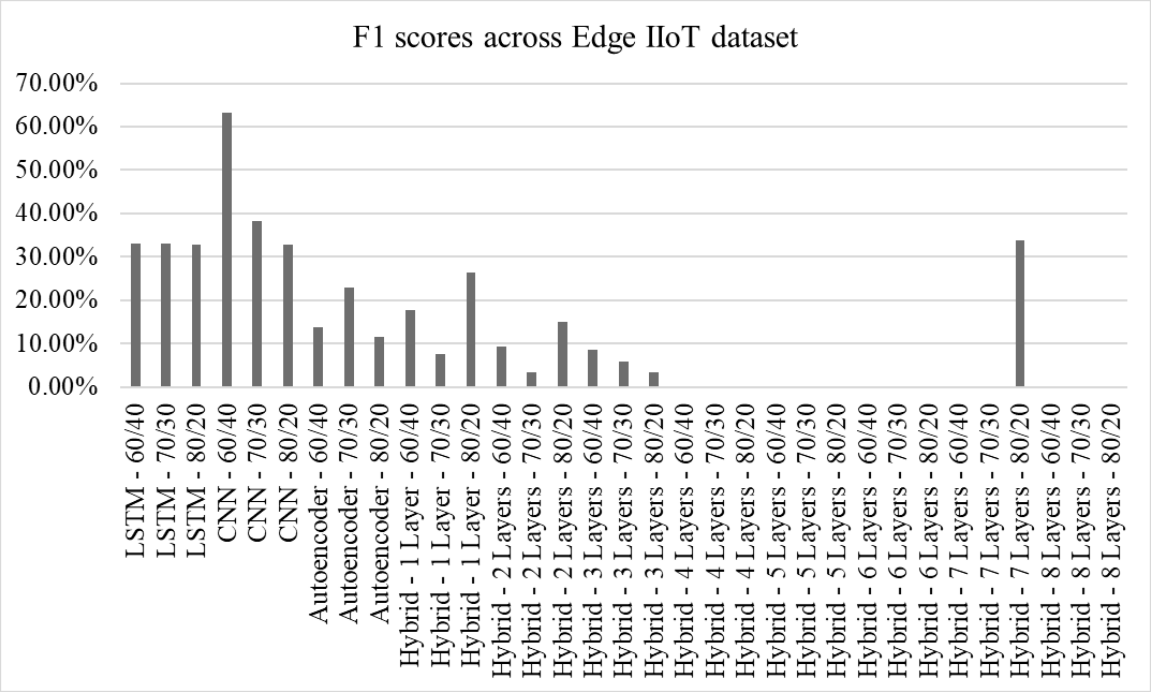


Figure 4.50: F1 scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

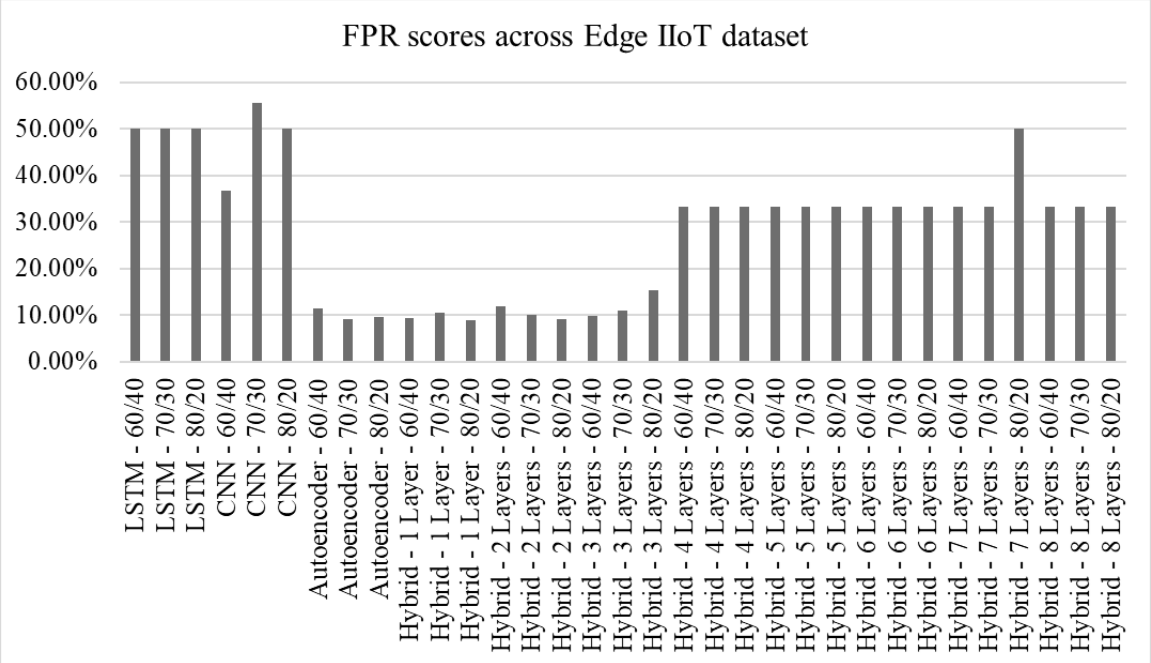


Figure 4.51: False positive rate for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Hybrid deep autoencoder model variants

Analysis of selected models in the Edge IIoT dataset provides insights into the hybrid autoencoder's performance on the general IoT dataset. Two models were selected for analysis: the 80/20 variants of the seven-hidden-layer and one-hidden-layer models.

The 80/20 model with seven hidden layers had an AUC of 0.5 on the ROC curve. This phenomenon demonstrates the model's poor performance, as it failed to discriminate between threats. The model had 9,922 false positives among the 10,078 true positives. The model's validation loss curve was stable at zero, but the testing loss rose from zero to 30 in the third epoch, before falling to zero in the sixth epoch. It then rose again to 40 by the ninth epoch. The model's test loss did not stabilize during testing, indicating the need for redesign. These results are shown in Figures 4.52, 4.53, and 4.54.

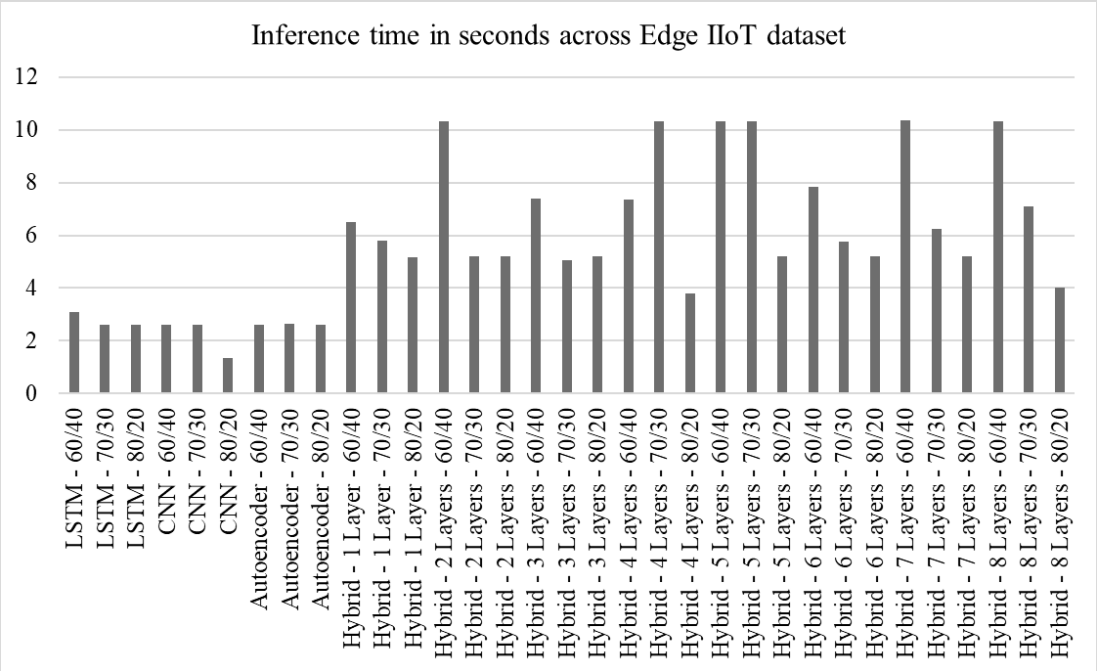


Figure 4.52: Inference time in seconds for the Edge IIoT dataset across the LSTM, CNN, and Autoencoder and Hybrid deep autoencoder model variants

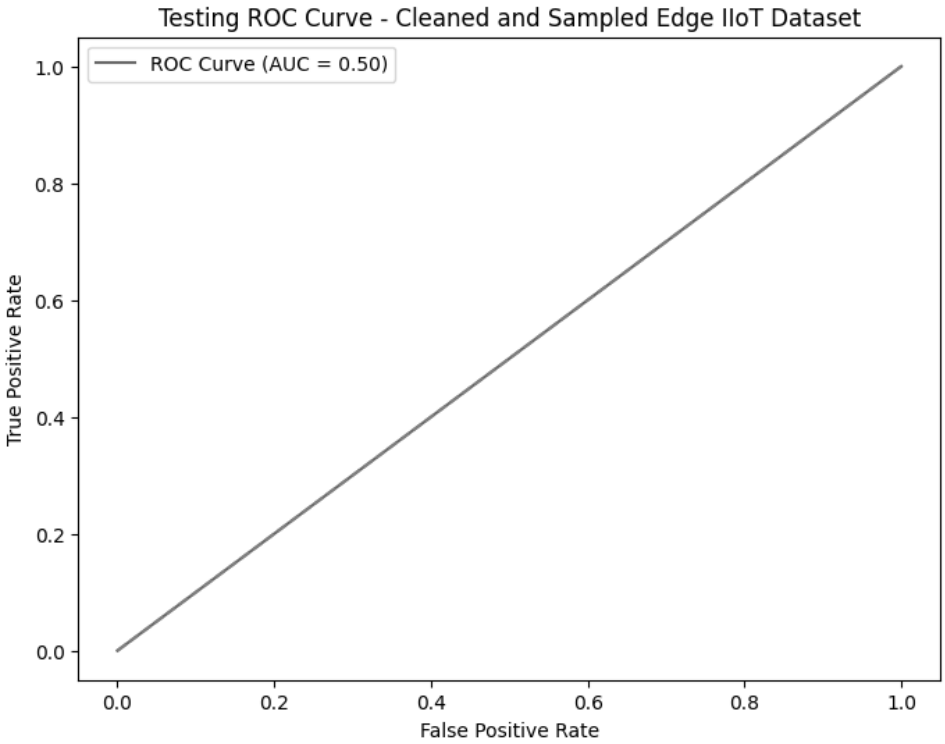


Figure 4.53: Receiver Operating Characteristics curve for seven hidden layers' 80/20 hybrid model on Edge IIoT dataset

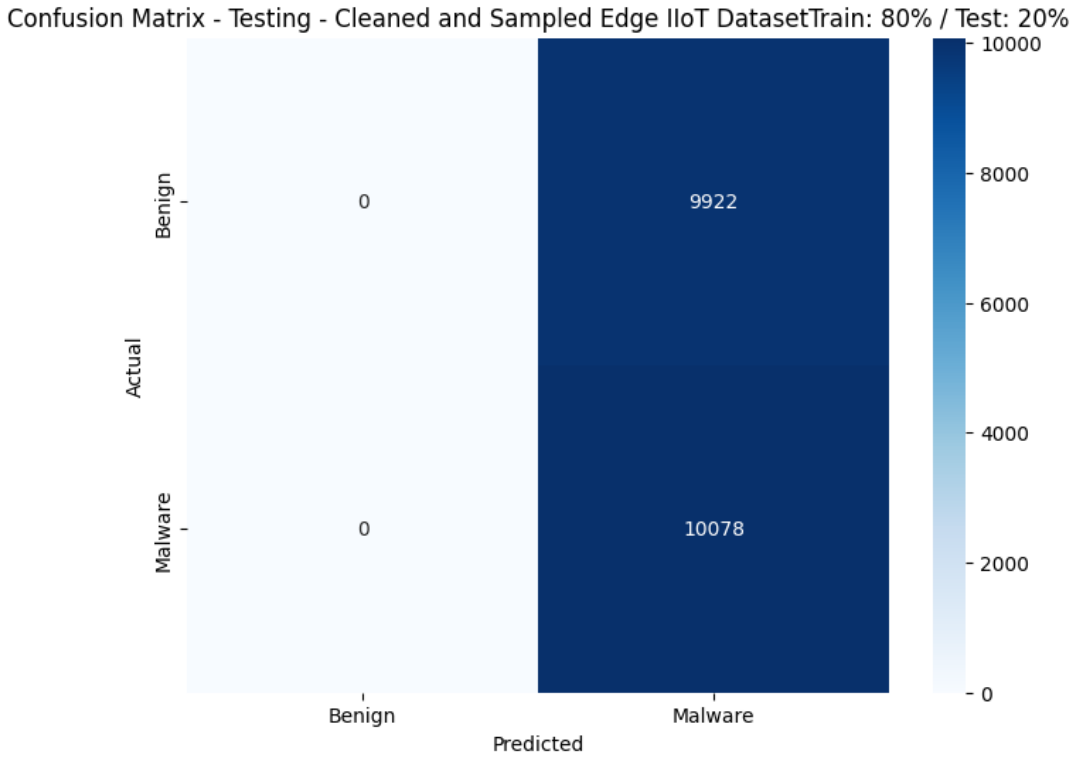


Figure 4.54: Confusion matrix for seven hidden layers' 80/20 hybrid model on Edge IIoT dataset

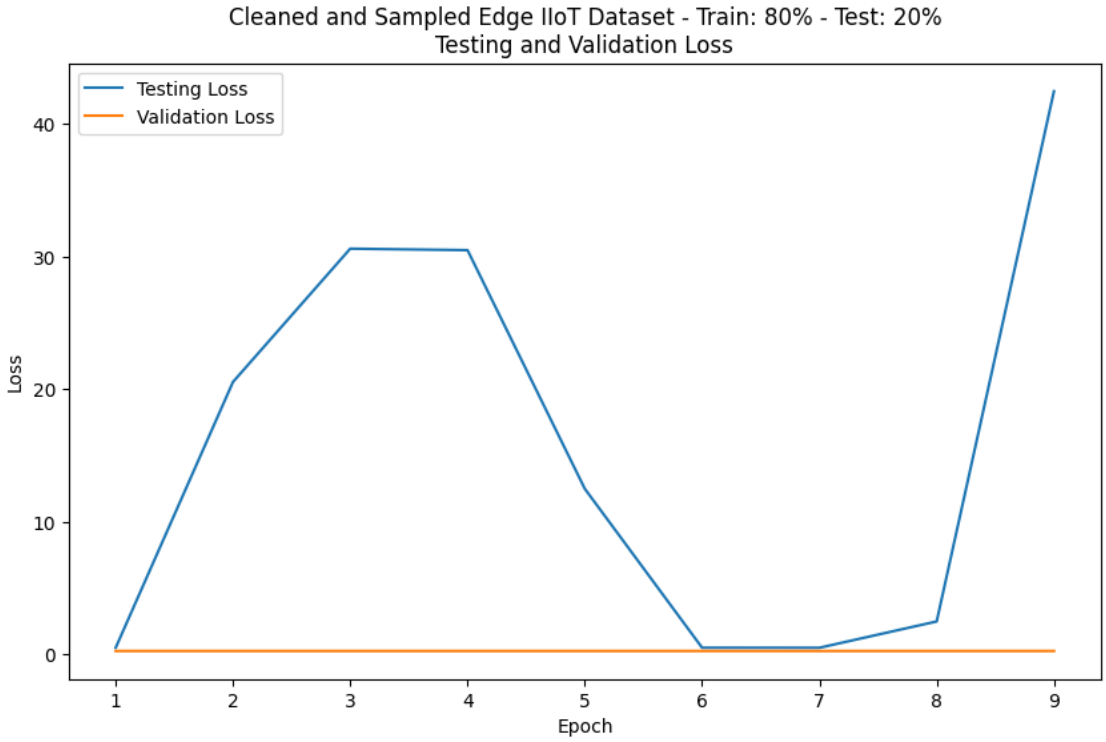


Figure 4.55: Testing and validation loss curves for seven hidden layers' 80/20 hybrid model on Edge IIoT dataset

The analysis of the 80/20 variant of a one-hidden-layer model shows the model's performance on a general IoT dataset as the number of hidden layers is reduced from seven to one, with an 80/20 training/testing split. The model demonstrated an AUC of 0.5 in its ROC curve, indicating non-discriminatory classification of threats. The model's confusion matrix recorded 1,072 false negatives and 1,036 true negatives. Furthermore, there were 8,886 false-positive cases and 9,006 true-positive cases. The review of the loss curves provides more insights into this performance. For instance, the testing loss curve started at about 18, before falling to 5 in the third epoch. It then rose to 30 in the fourth epoch, dropped to 15 in the fifth epoch, and rose to 20 in the sixth epoch. The model stabilized at about 42 from the ninth epoch, and retained this score until the end of the session in the twentieth epoch. The validation loss curve stabilized at zero until the end of the testing session. These results are shown in Figures 4.56, 4.57, and 4.58.

The analysis of training loss curves for the selected models enhances understanding of their testing behavior. For instance, the loss curve for the 70/30 model with seven hidden layers started at 0.25012 and fell to 0.25002 in the second epoch. This value

wobbled downward until the model reached 0.24998 in the sixth epoch. It then rose abruptly to 0.25004 in the seventh epoch, possibly due to the changes in learning rate. It then fell to 0.24997 in the ninth epoch. The model's validation loss wobbled between 0.24004 and 0.25 until the end of the training session. These results imply that there would have been minimal changes in the model's performance even with additional training epochs. Similar behavior is observed in the training loss curves for the 80/20 variant with seven hidden layers. The model's loss curve started at 0.2513, before falling to 0.2505 in the third epoch. It then declined to 0.5202 in the seventh epoch. The validation loss began at 0.2505 and dropped to 0.2499 in the second epoch. It then wobbled until it reached 0.2502 in the seventh epoch. The slight wobbling of these scores indicates that the model would have shown only a slight improvement even with additional training epochs. These results are shown in Figures 4.59 and 4.60.

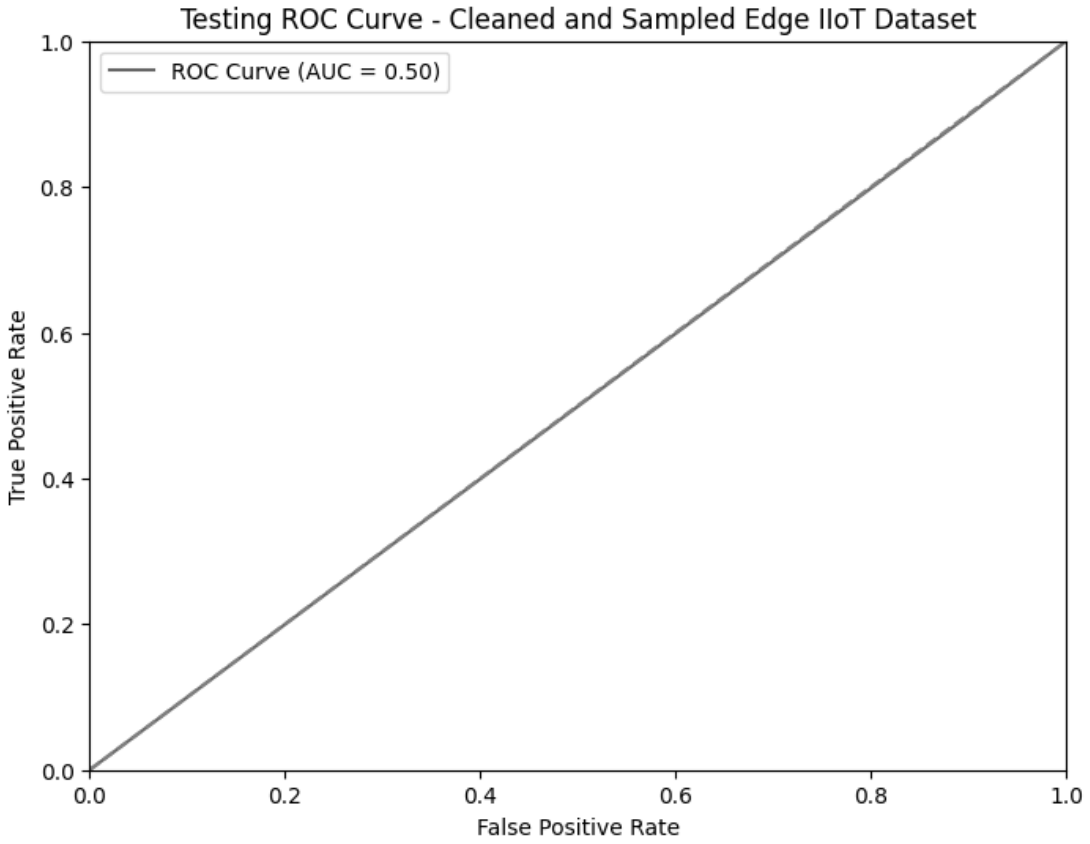


Figure 4.56: Receiver Operating Characteristics curve for one hidden layer's 80/20 hybrid model on Edge IIoT dataset

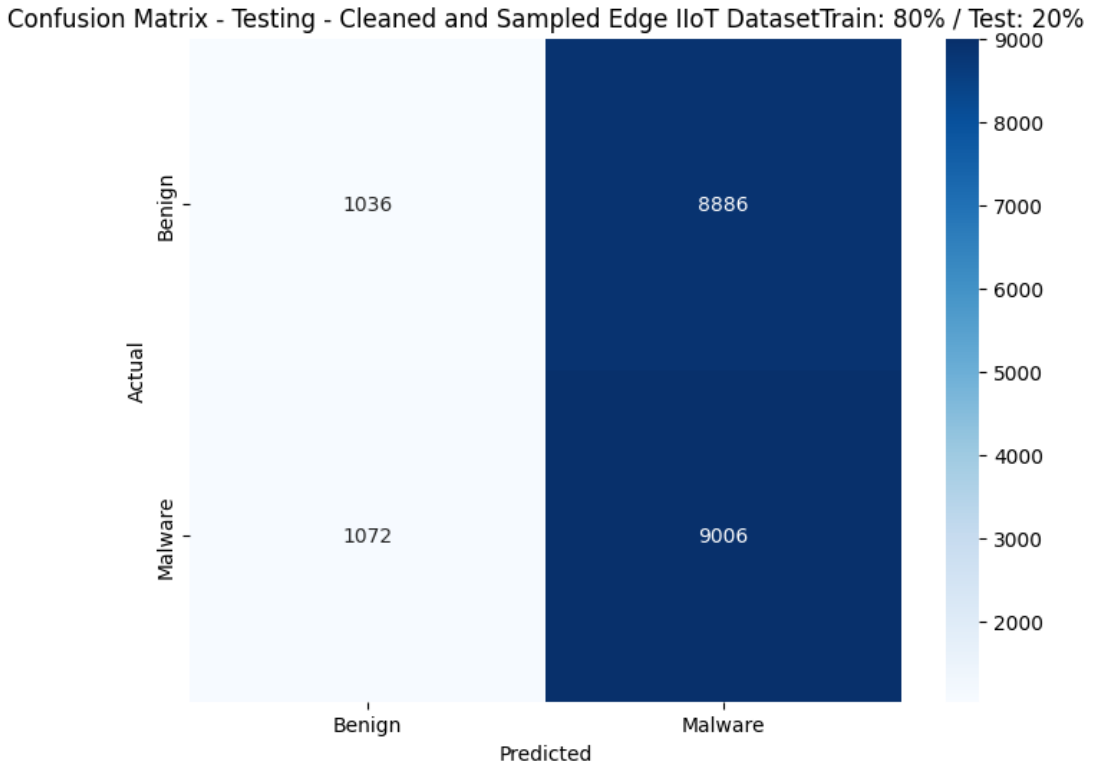


Figure 4.57: Confusion matrix for one hidden layer's 80/20 hybrid model on Edge IIoT dataset

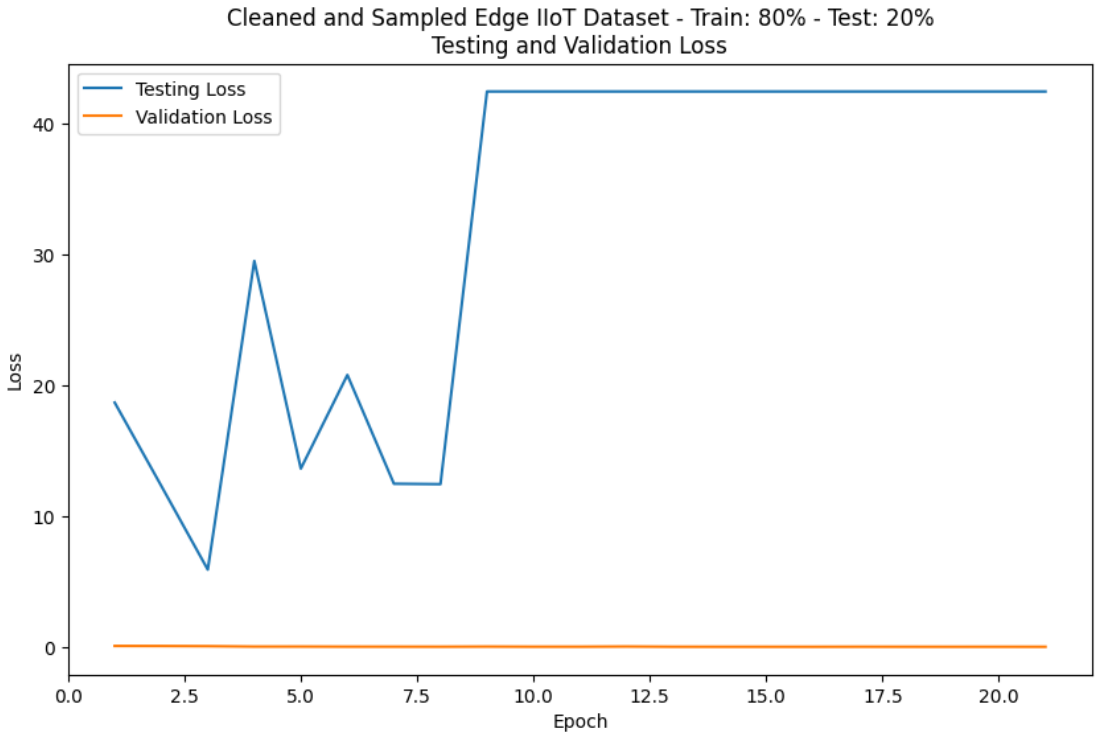


Figure 4.58: Testing and validation loss curves for one hidden layer's 80/20 hybrid model on Edge IIoT dataset

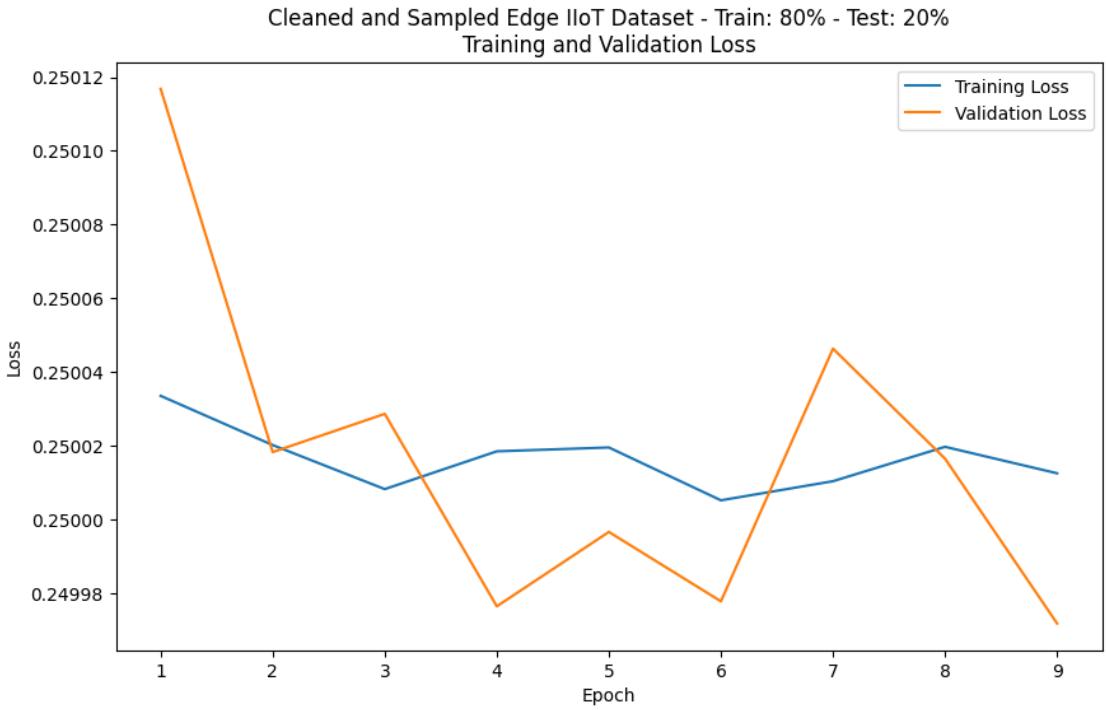


Figure 4.59: Training and validation loss curves for seven hidden layers' 80/20 hybrid model on the Edge IIoT dataset

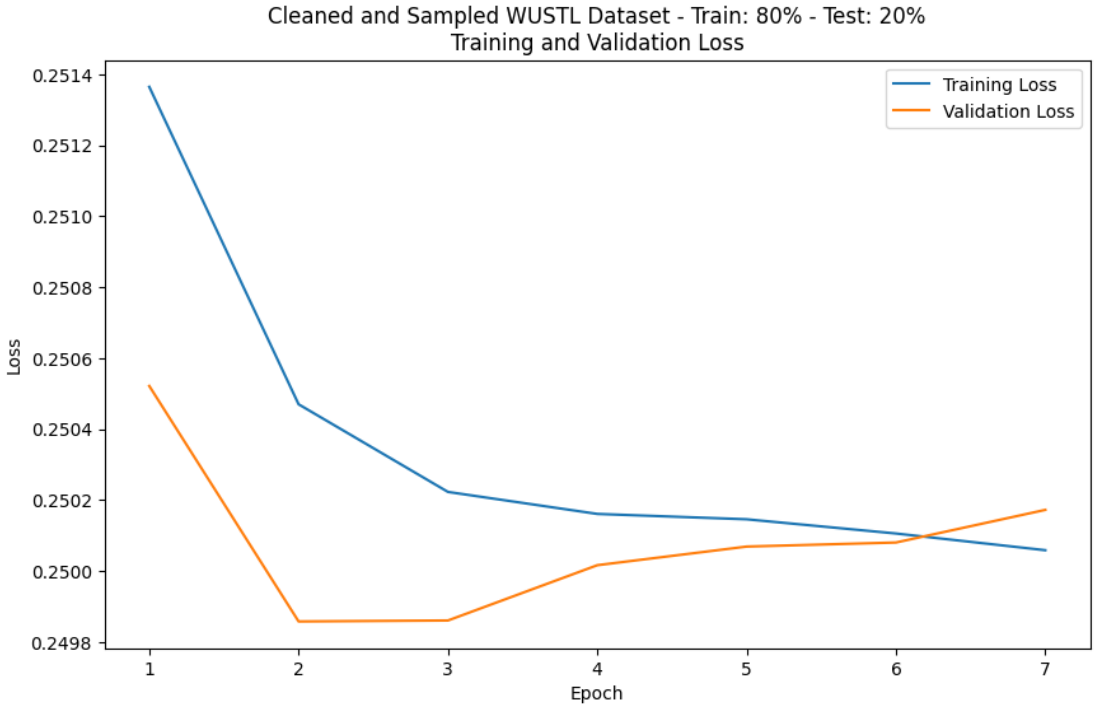


Figure 4.60: Training and validation loss curves for one hidden layer's 80/20 hybrid model on the Edge IIoT dataset

4.4 Nadam-Optimized Hybrid Model

The Nadam optimization function was integrated into the standard model, resulting in an optimized intrusion detection model. The improved model was trained and tested across three datasets, each with three training-to-testing ratios. The training session was conducted in eight phases, corresponding to the number of hidden layers in the encoder/decoder segments, ranging from 1 to 8. The results from these sessions were evaluated based on recall, accuracy, precision, F1, and inference time.

4.4.1 Nadam-Optimized Hybrid Model on ICU Dataset

The analysis of recall scores for the ICU dataset across the baseline and Nadam-optimized hybrid model shows the implications of Nadam optimization on the models' sensitivity. The 80/20 variant of the three-hidden-layer hybrid model emerged as the best, achieving a recall of 99.95% and outperforming the baseline models. This model is followed by the CNN's 70/30 variant at 71.45%, a significant difference between the two. Although other variants of the hybrid model struggle to achieve meaningful results, this improvement demonstrates the promise of optimization for achieving outstanding model performance. These scores are shown in Figure 4.61.

The review of precision scores for the model shows the changes in positive predictive value after implementing Nadam optimization in the hybrid model. The 80/20 variant of the three-hidden-layer model emerged as the best-performing model, achieving a score of 99.98%, outperforming baseline models such as CNN, LSTM, and Autoencoder. The CNN's 70/30 and 80/20 variants follow this model, achieving 79.43% and 75.07%, respectively, which are substantially lower than those presented by the leading Nadam-optimized hybrid model. These results are shown in Figure 4.62.

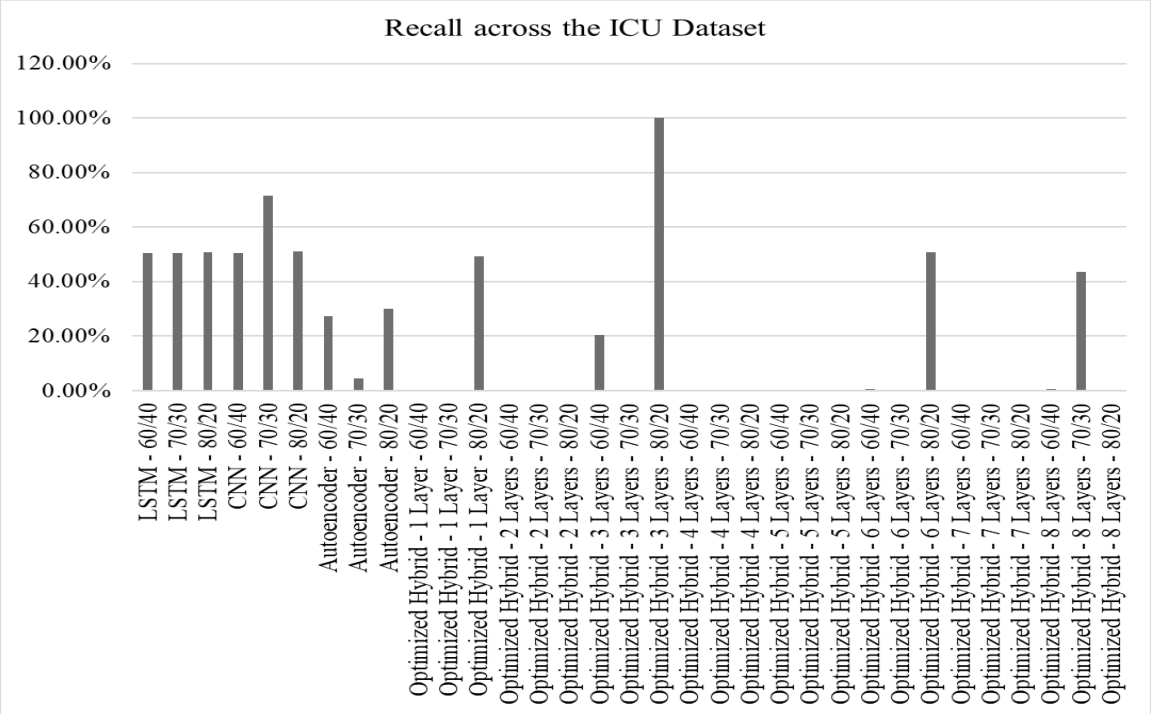


Figure 4.61: Recall scores for the ICU dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

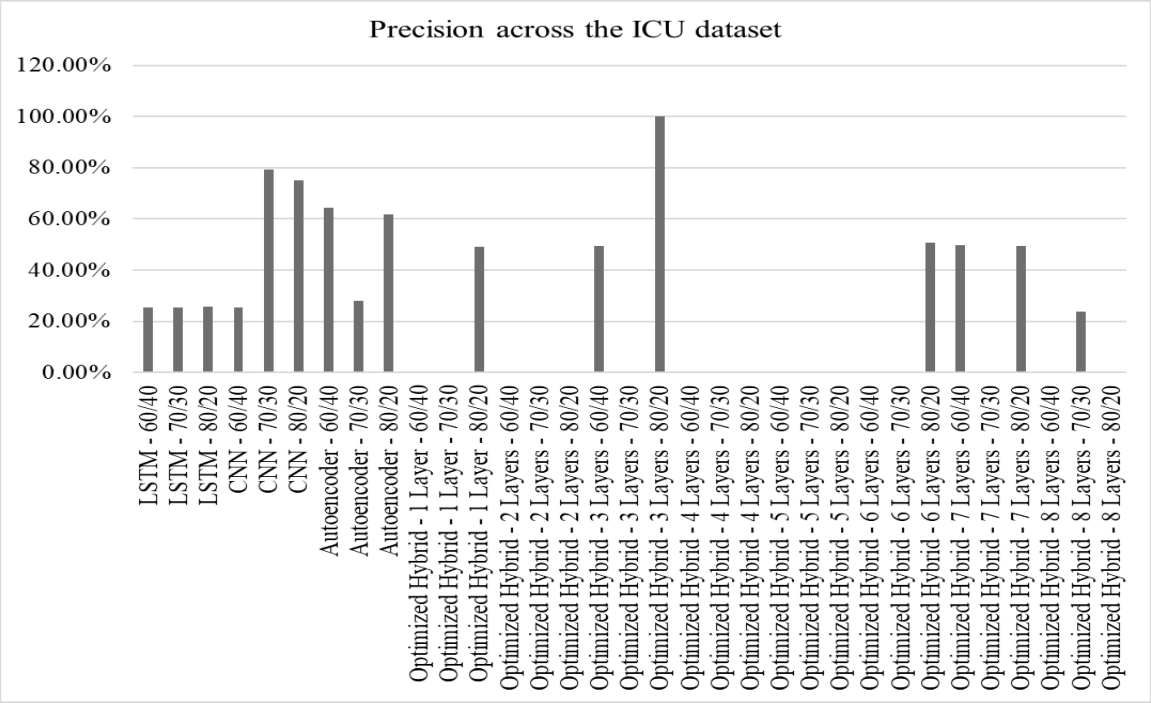


Figure 4.62: Precision scores for the ICU dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The review of specificity scores for the Nadam-optimized model and the baseline models on the ICU dataset shows significant improvement. For instance, the 80/20

variant of the three-hidden-layer model emerged as the best, with a score of 99.98%. This score outperformed the autoencoder, which scored 91.39% on the 60/40 split, 91.21% on the 80/20 split, and 89.90% on the 70/30 split. Numerous variants of the hybrid model achieve 75% or higher precision, demonstrating the importance of optimization in refining the model's performance. These results are shown in Figure 4.63.

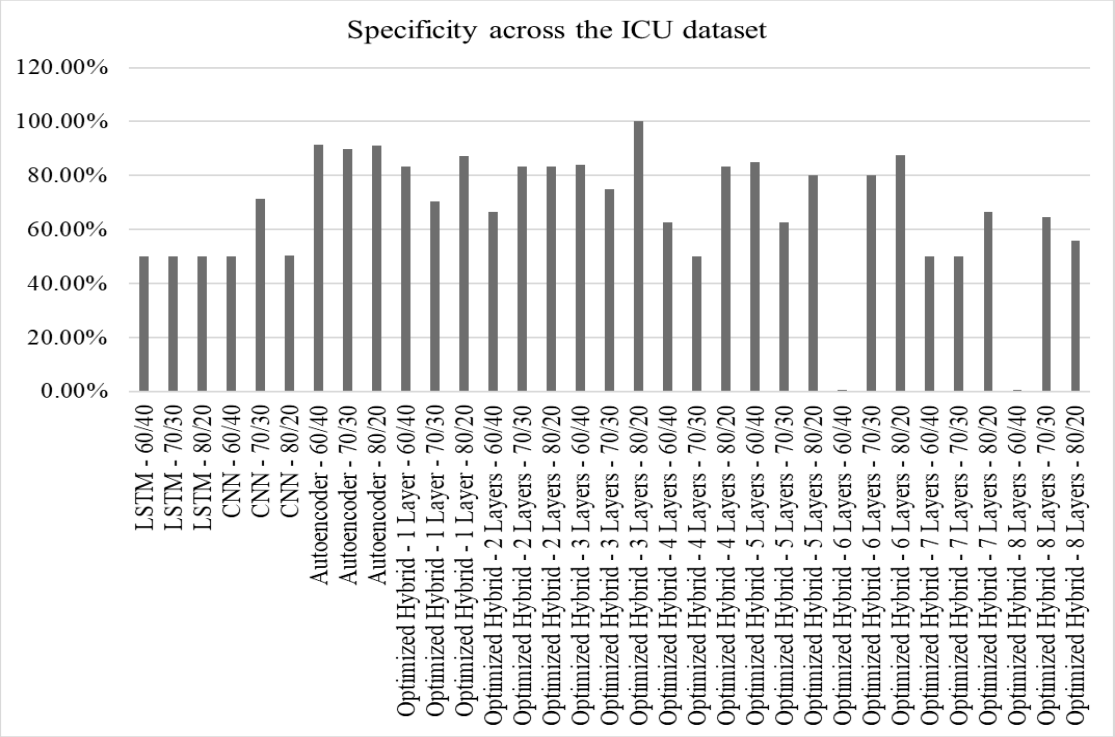


Figure 4.63: Specificity scores for the ICU dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The review of accuracy scores for the Nadam-optimized hybrid model compared with the baseline models shows an improvement in the model's overall correctness. The 80/20 variant of the three-hidden-layer model emerged as the best, achieving a score of 99.95%. This model is followed by the CNN's 70/30 and 80/20 variants, gaining 71.45% and 50.95%, respectively. The six-hidden-layer 80/20 variant of the Nadam-optimized model reached the 50% mark by scoring 50.68%. This phenomenon demonstrates that Nadam optimization improves the model's performance. These results are shown in Figure 4.64.

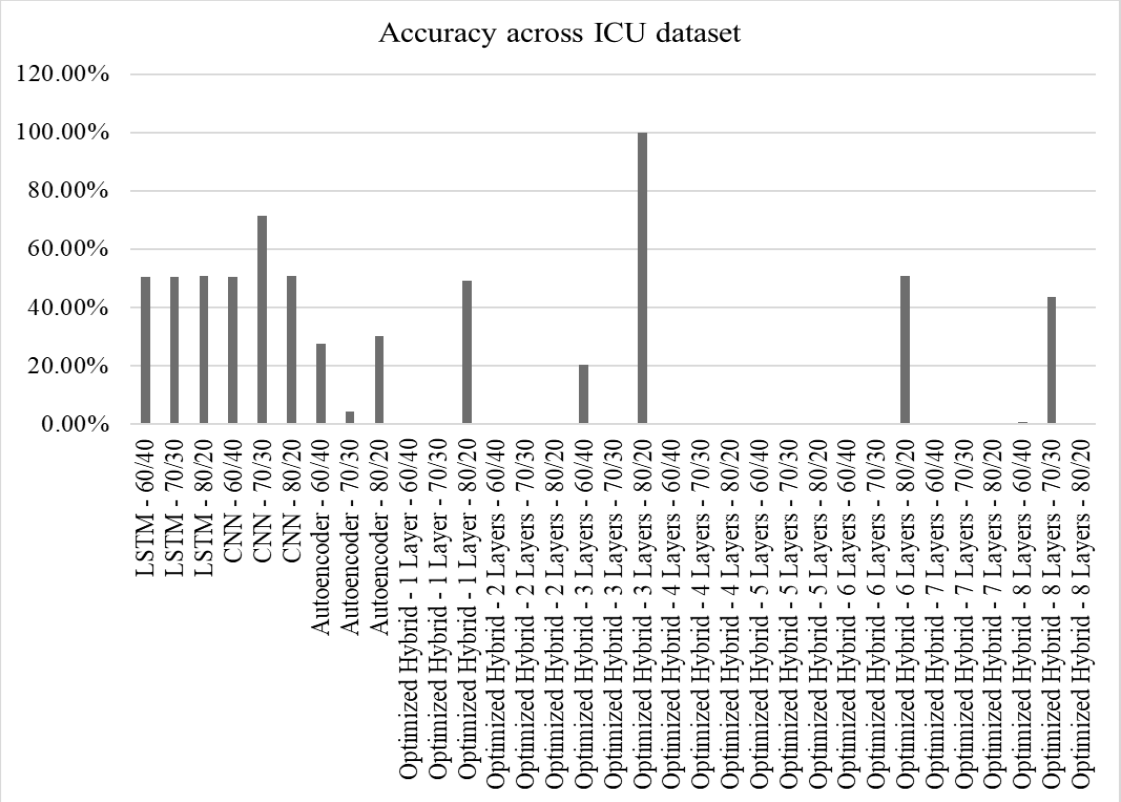


Figure 4.64: Accuracy scores for the ICU dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The assessment of F1 scores for the Nadam-optimized models versus the baseline models on the ICU dataset shows improved balance between recall and precision after optimization. The three hidden layers' 80/20 led with 99.97%, which is followed by the CNN's 70/30 variant at 69.30%. The six-hidden-layer 80/20 variant reached the 50% mark, scoring 50.70%, demonstrating the promise of Nadam optimization for the hybrid model. These results are shown in Figure 4.83. The false-positive rate of the Nadam-optimized model decreased significantly on the ICU dataset, demonstrating the importance of Nadam optimization. The three hidden layers' 80/20 variant had an FPR of 0.02%, which was substantially lower than those reported for the autoencoder models: 8.61% in the 60/40, 8.79% in the 80/20, and 10.10% in the 70/30 variants. The six-hidden-layer 70/30 variant had an FPR of 12.33%, followed by the one-hidden-layer 80/20 variant at 12.70%. These results are shown in Figure 4.65 & 4.66.

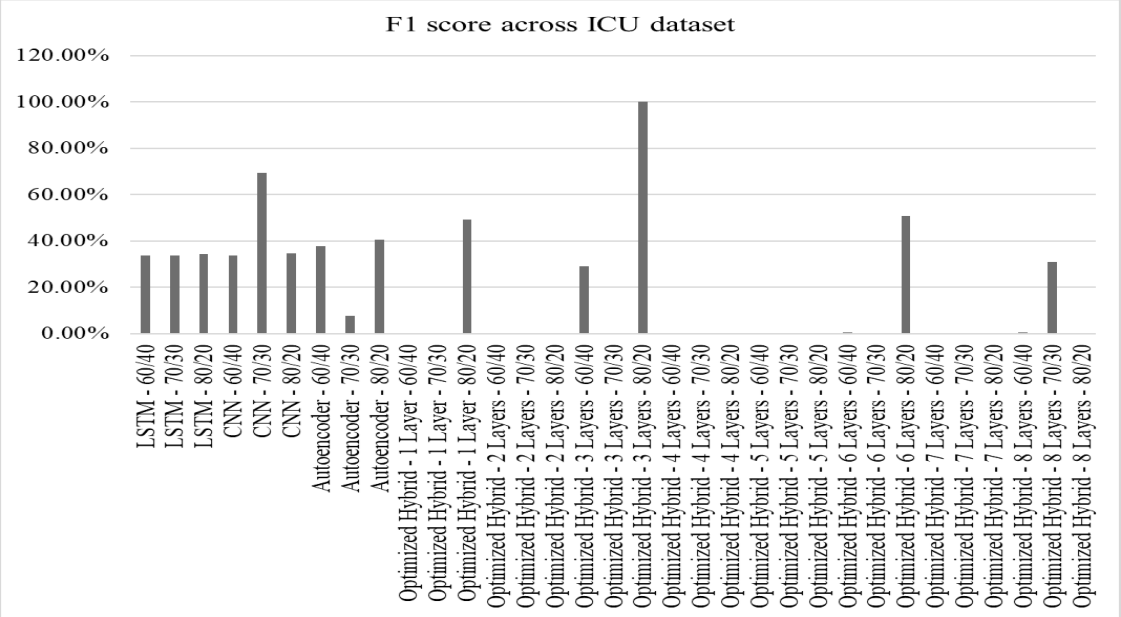


Figure 4.65: F1 scores for the ICU dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

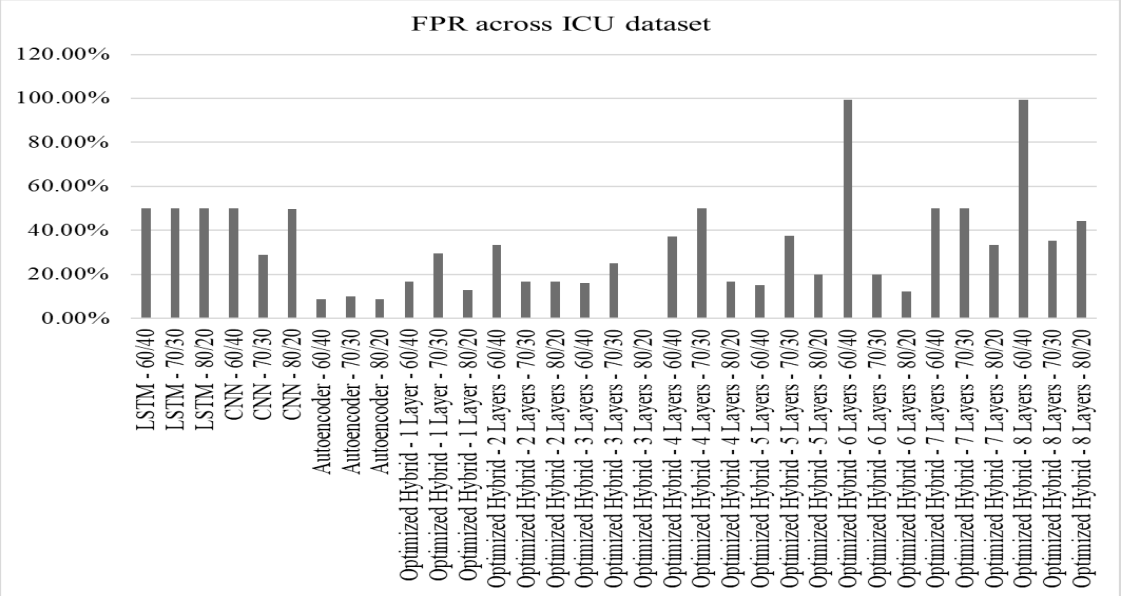


Figure 4.66: False positive rates for the ICU dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The assessment of inference time for the Nadam-optimized model relative to the baseline models showed no significant improvement in detection speed. For instance, the LSTM, CNN, and Autoencoder variants dominated the inference time, recording the lowest detection speeds. One hidden layer's 80/20 variant recorded a speed of 1.20 seconds, outperforming the 80/20 and 70/30 variants of the autoencoder, the 60/40

variant of the CNN, and the 80/20 variant of the LSTM. However, other variants of the Nadam-optimized hybrid model dominated the lower end of the detection speed, demonstrating the model's slowness in identifying threats. These results are shown in Figure 4.67.

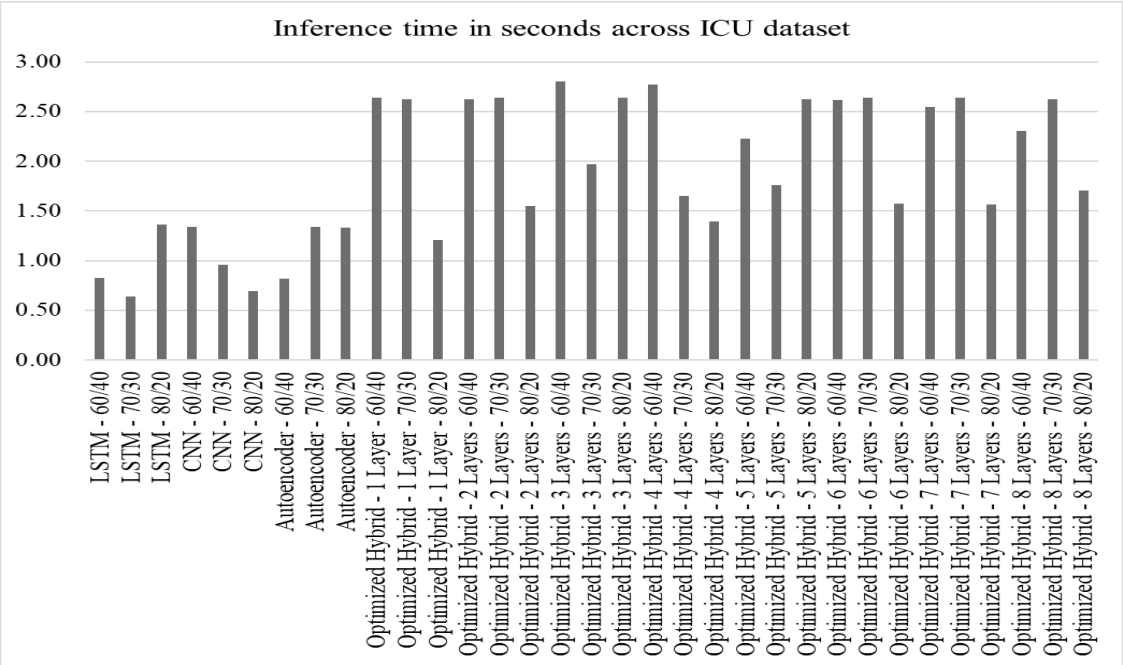


Figure 4.67: Inference time in seconds for the ICU dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The analysis of selected models for the ICU dataset using Nadam-optimized hybrid models enhances understanding of their behavior during optimization. The chosen models were 80/20 variants of three hidden and six hidden layers. The 80/20 variant of the three-hidden-layer model achieved an AUC of 1.00 on the ROC curve, demonstrating perfect separation between the threats. This high performance was replicated in the confusion matrix, where the model recorded two false positives and zero false negatives. The model’s testing loss started at about 32, then increased to nearly 60 in the second epoch before falling to zero in the third. The loss value rose to 35 in the fourth epoch, and wobbled until the end of the training session. Additional testing epochs might have reduced the number of false positives identified by the model. These results are shown in Figures 4.68, 4.69 and 4.70.

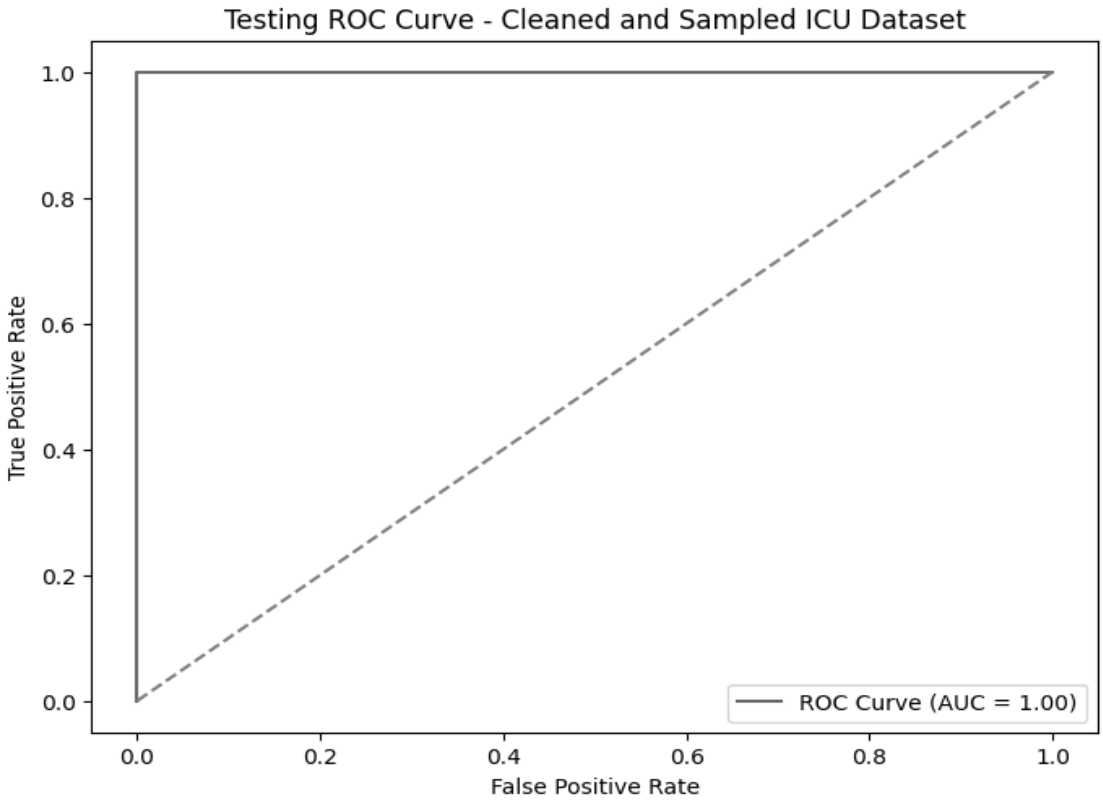


Figure 4.68: Receiver Operating Characteristics curve for three hidden layers' 80/20 Nadam-optimized hybrid model on ICU dataset

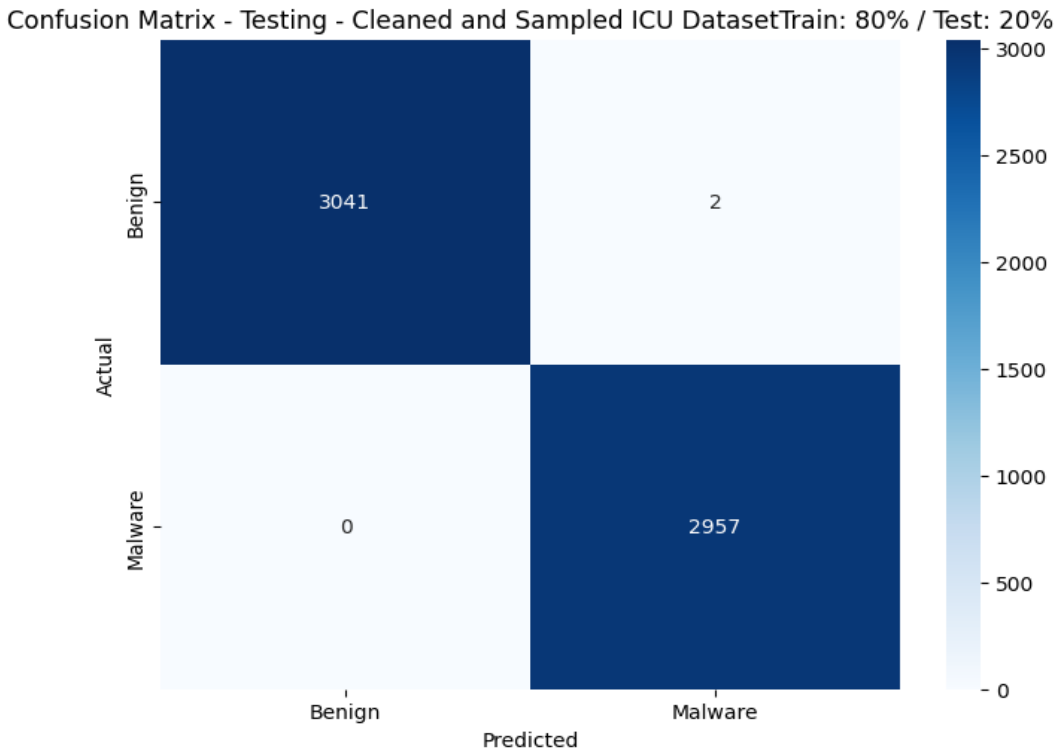


Figure 4.69: Confusion matrix for three hidden layers' 80/20 Nadam-optimized hybrid model on ICU dataset

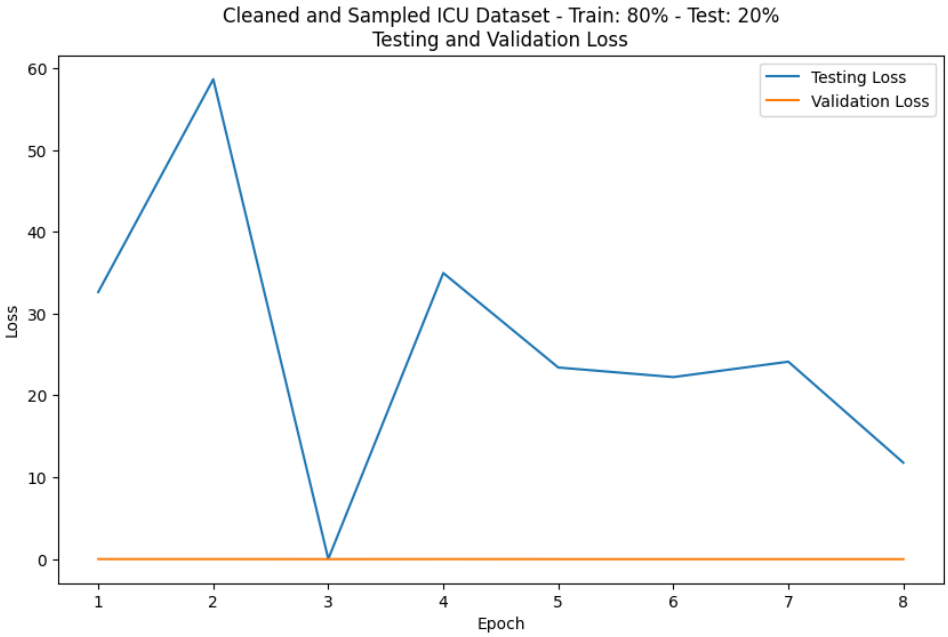


Figure 4.70: Testing and validation loss curves for three hidden layers' 80/20 Nadam-optimized hybrid model on ICU dataset

The analysis of the six-hidden-layer 80/20 variant offers insights into the model's performance as the number of layers increases from three to six. The model had an AUC of 1.00 on the ROC curve, indicating perfect separation between threats and benign cases. The model had two false positives and zero false negatives. Although the model's testing loss started at 0, it rose to 40 in the second epoch and then fell slightly to 30 in the third. It wobbled until the end of the training session, while the validation loss curve remained at zero until the end the testing session. These results are shown in Figures 4.71, 4.72, and 4.73.

The assessment of training curves for the selected models enhances understanding of the testing behavior demonstrated by these models. For instance, the training loss curve for the 80/20 variant with three hidden layers started at 0.0115 and fell to 0.003 in the third epoch. This curve then rose to 0.005 in the fourth epoch and fell to 0.005 in the fifth epoch. It then stabilized at 0 in the sixth epoch and remained there until the end of the training session. The validation loss curve started at 0.007, increased slightly to 0.0075 in the second epoch, then fell to 0 in the third epoch. It then rose to 0.0065 in the fourth epoch and fell to 0.005 in the fifth, stabilizing at this score until the end of the session. The training and validation loss curves for the six-hidden-layer 80/20 variant started at zero and increased to 0.5 by the third epoch, then stabilized at this

level through the end of training. These results demonstrate that the models would not have improved with additional training epochs, as they had stabilized. The results are shown in Figures 4.74 and 4.75.

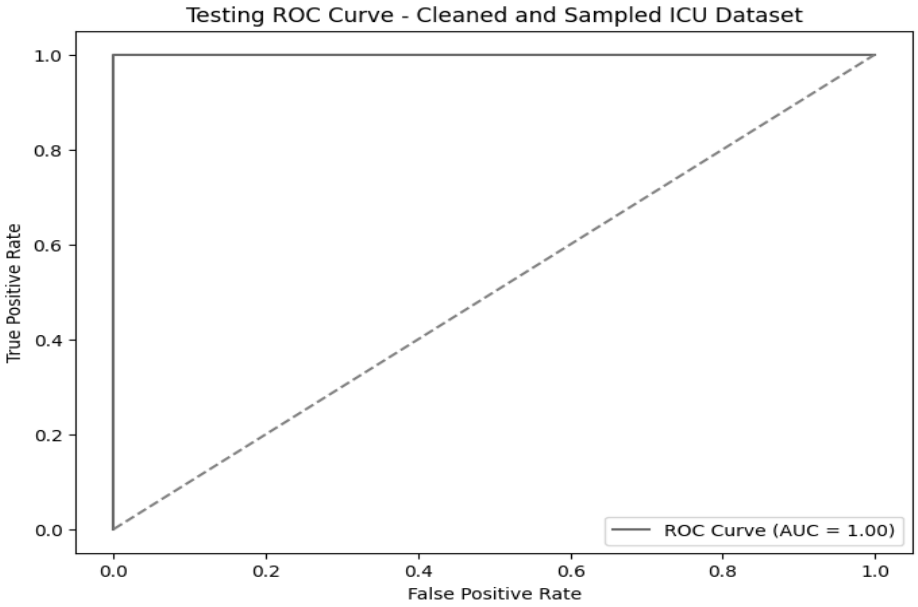


Figure 4.71: Receiver Operating Characteristics curve for a six-hidden-layer 80/20 Nadam-optimized hybrid model on the ICU dataset

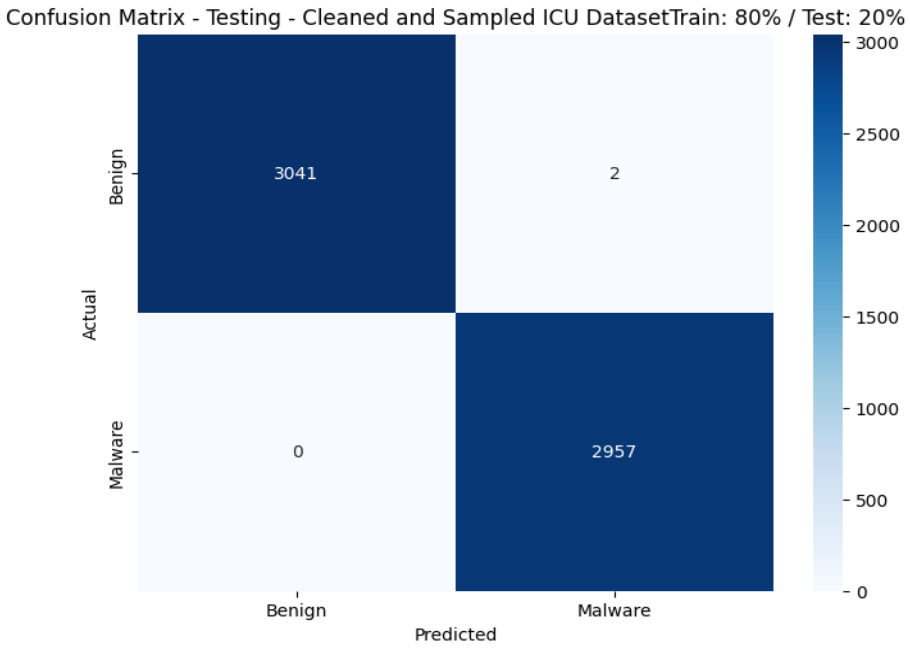


Figure 4.72: Confusion matrix for six-hidden-layer 80/20 Nadam-optimized hybrid model on ICU dataset

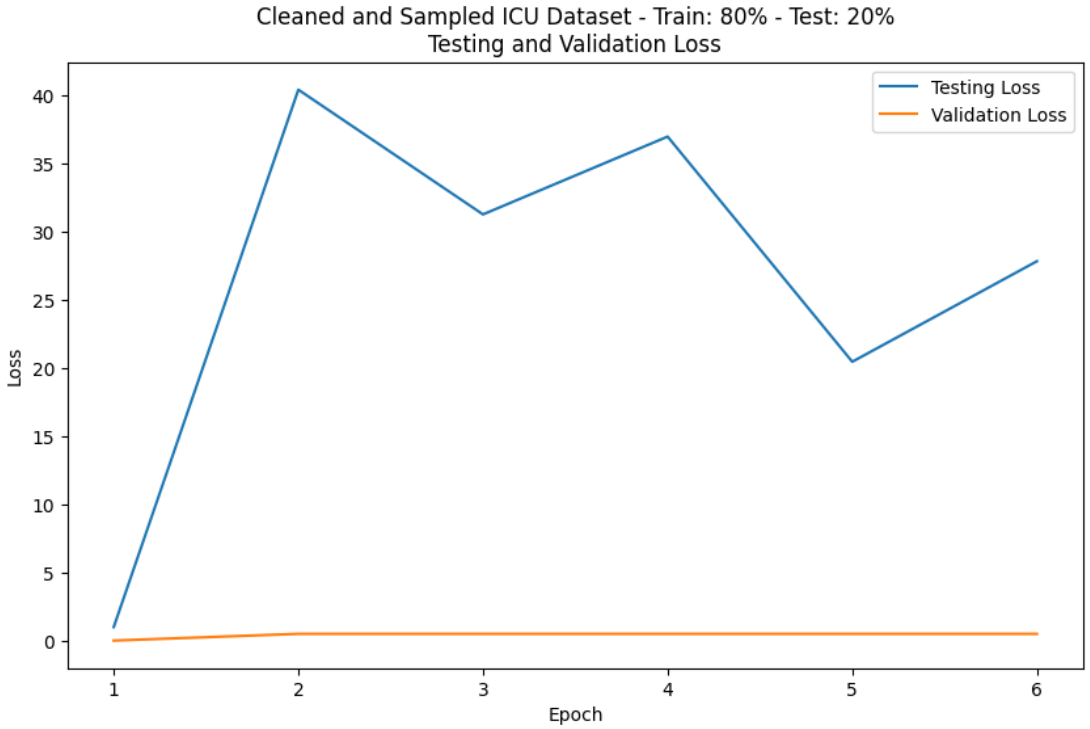


Figure 4.73: Testing and validation loss curves for a six-hidden-layer 80/20 Nadam-optimized hybrid model on the ICU dataset

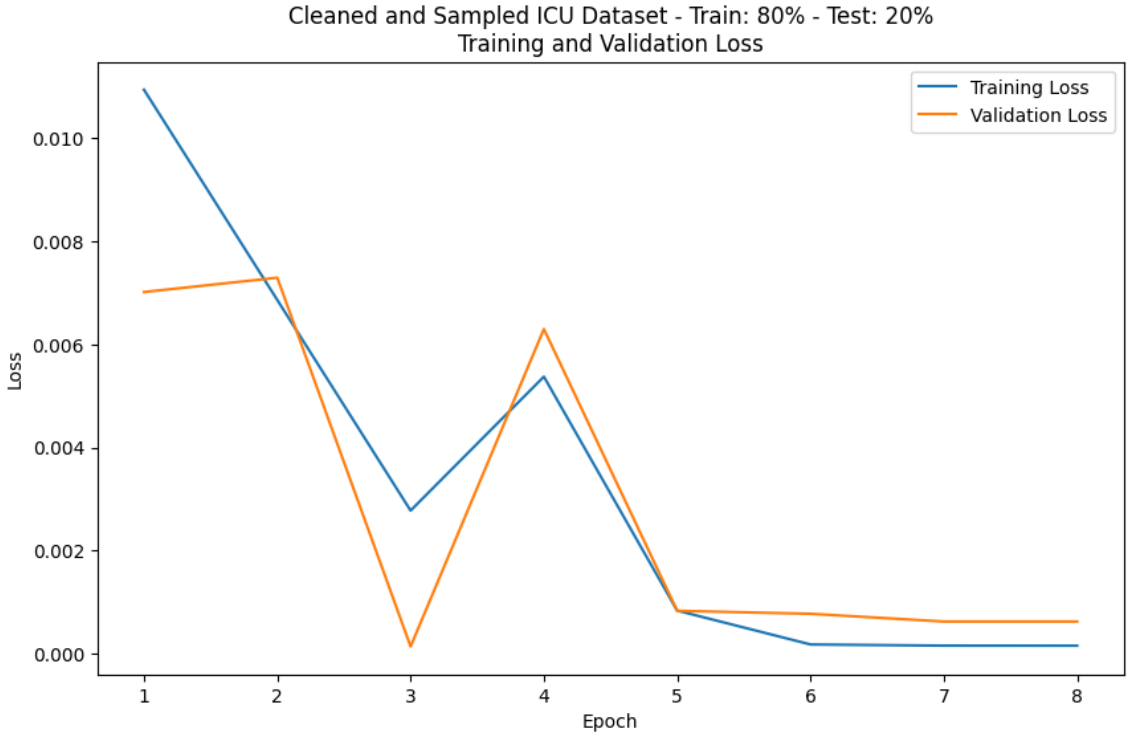


Figure 4.74: Training and validation curves for three-hidden-layer 80/20 Nadam-optimized hybrid model on the ICU dataset

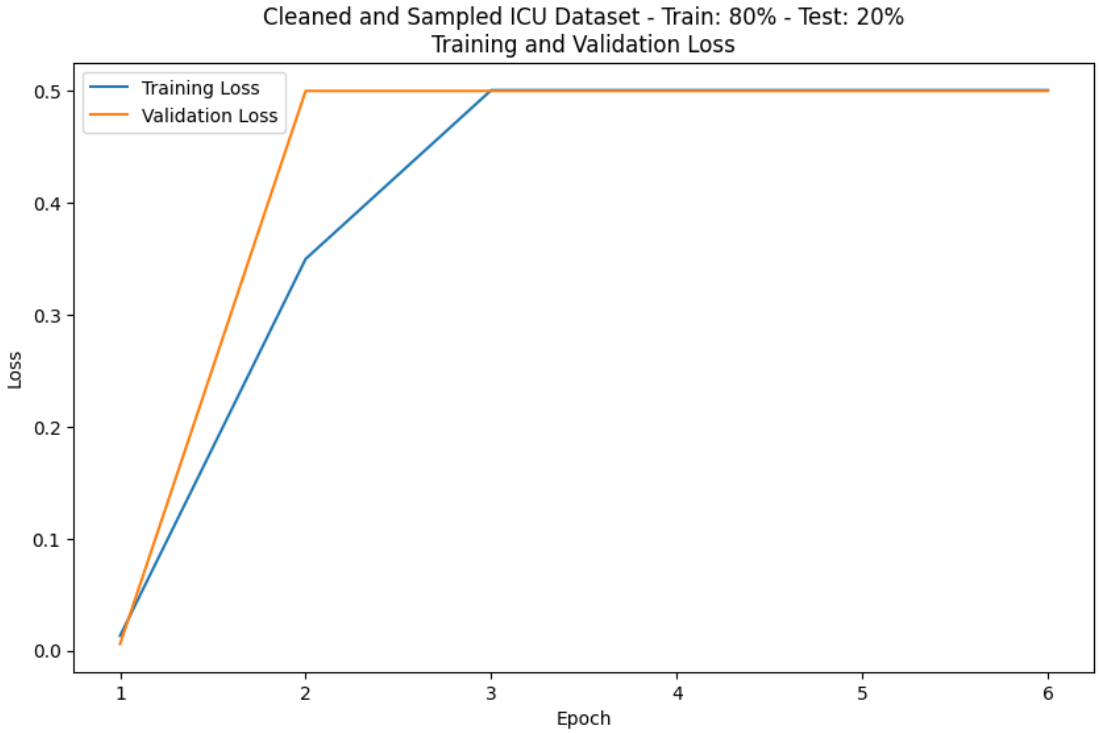


Figure 4.75: Training and validation curves for a six-hidden-layer 80/20 Nadam-optimized hybrid model on the ICU dataset

4.4.2 Nadam-Optimized Hybrid Model on WUSTL Dataset

Analysis of Nadam-optimized models against the baseline models on the WUSTL dataset shows that recall scores did not improve with Nadam optimization. The CNN and LSTM variants still maintained their dominant positions in recall, though neither reached 50%. The best Nadam-optimized hybrid model was a three-hidden-layer 80/20 variant, which scored 31.50%. These results are shown in Figure 4.76.

The review of precision scores for the WUSTL dataset, comparing baseline and Nadam-optimized models, shows the impact of optimization on the positive predictive value. The hybrid models led in this category, with a 60/40 variant with one hidden layer scoring 88.68%. The three-hidden-layer 70/30 and four-hidden-layer 60/40 variants scored 87.21% and 82.07%, respectively, placing the models at 75% and above. This phenomenon shows that Nadam optimization effectively improved the positive predictive value of the models. These results are shown in Figure 4.77.

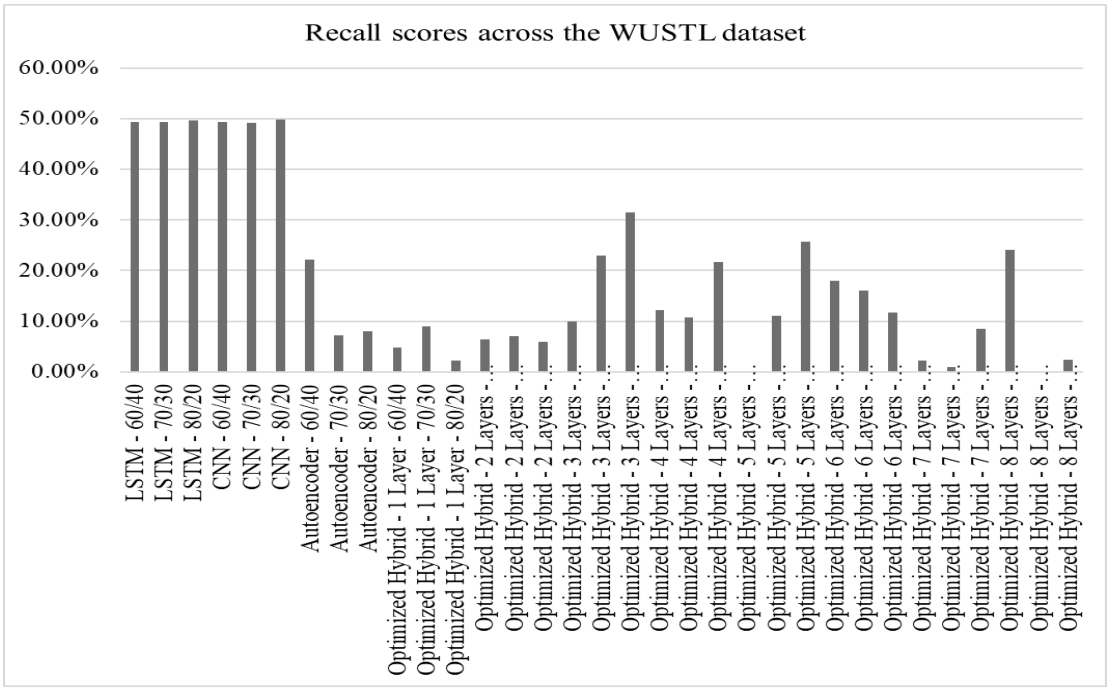


Figure 4.76: Recall scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

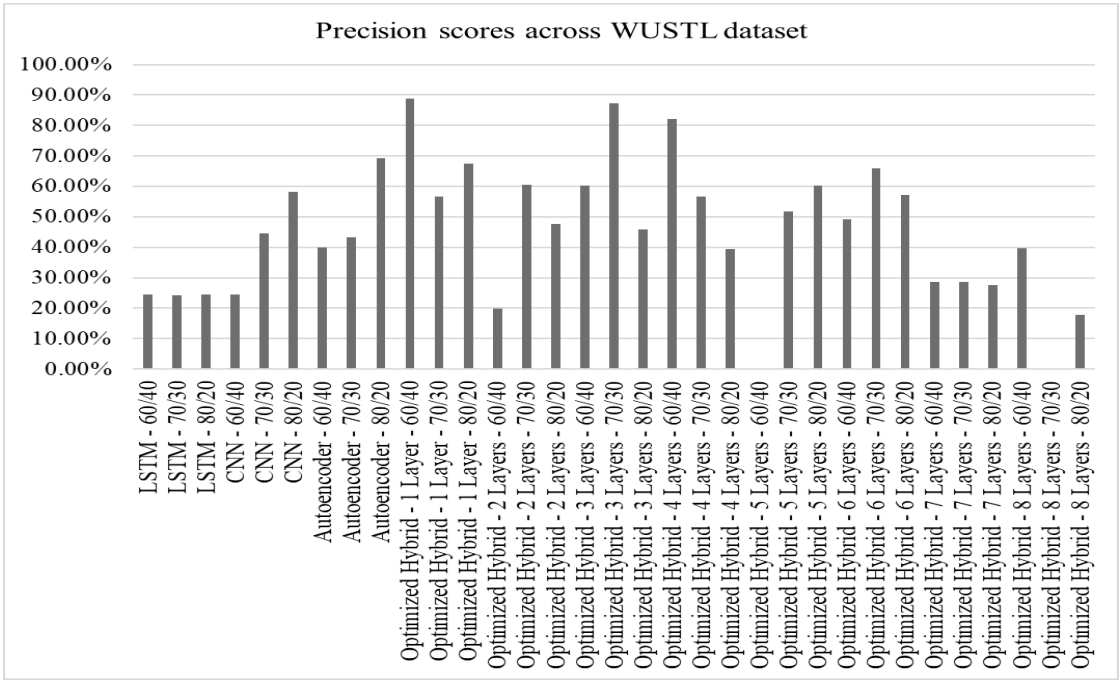


Figure 4.77: Precision scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The assessment of specificity scores for the WUSTL dataset on Nadam-optimized hybrid models against the baseline models highlights the benefits of Nadam

optimization in correctly identifying negative cases. The 80/20 five-hidden-layer variant and the 70/30 three-hidden-layer variant led with 92.15% and 92.11%, respectively. These models outperformed baselines such as CNNs, LSTMs, and autoencoders, demonstrating the benefits of Nadam optimization for improving model efficiency. These results are shown in Figure 4.78.

The analysis of accuracy scores for the WUSTL dataset on the Nadam-optimized hybrid model, compared with baseline models, shows the overall correctness of the hybrid model. The CNN and LSTM models achieved the highest accuracy, indicating that Nadam optimization did not improve overall accuracy. The best hybrid model was the 80/20 variant with three hidden layers, which scored 31.50%. These results are shown in Figure 4.79.

The review of F1 scores for the WUSTL dataset on Nadam-optimized hybrid and baseline models shows the implications of Nadam optimization on the models. Although the 70/30 variant of the three-hidden-layer model emerged as the best, it scores 34.96%. This phenomenon implies that the model failed to meet the 50% mark. The CNN's 80/20 variant scores 33.82%, while the five-hidden-layers' 80/20 had 33.48%. These results are shown in Figure 4.80.

The analysis of the false-positive rate for the Nadam-optimized hybrid model compared with the baseline models on the WUSTL dataset demonstrates the benefits of optimization for detection accuracy. The hybrid models recorded the lowest false-positive rates, with the 80/20 five-hidden-layer variant leading at 7.85%. This model was followed by three hidden layers, 70/30 at 7.89%. Other Nadam-optimized hybrid models scored below 10%, demonstrating the benefits of Nadam optimization. These results are shown in Figure 4.81.

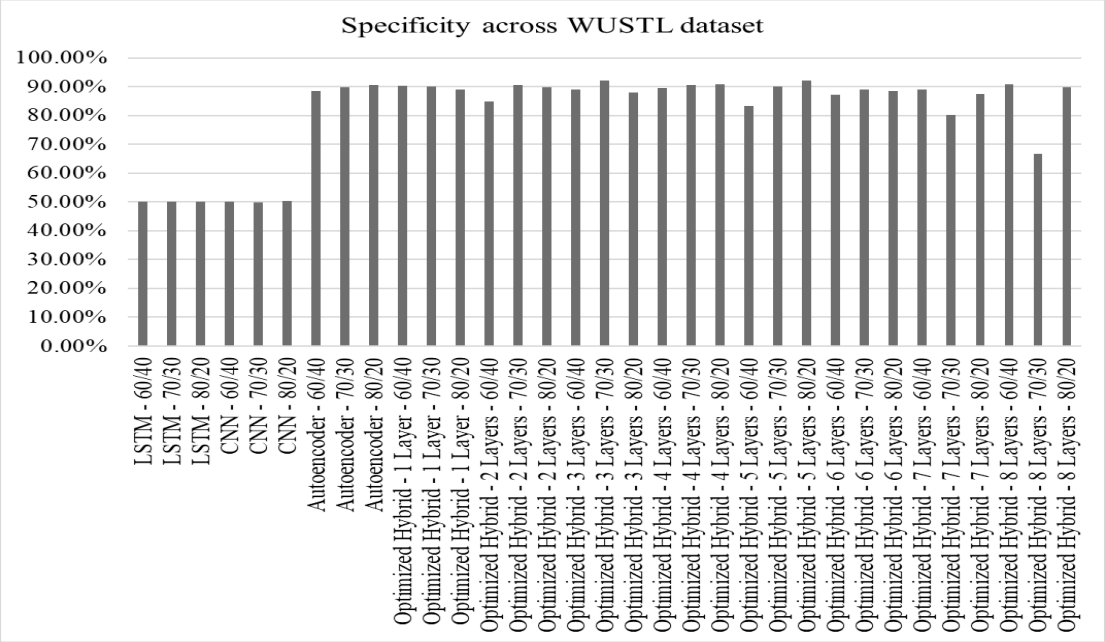


Figure 4.78: Specificity scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

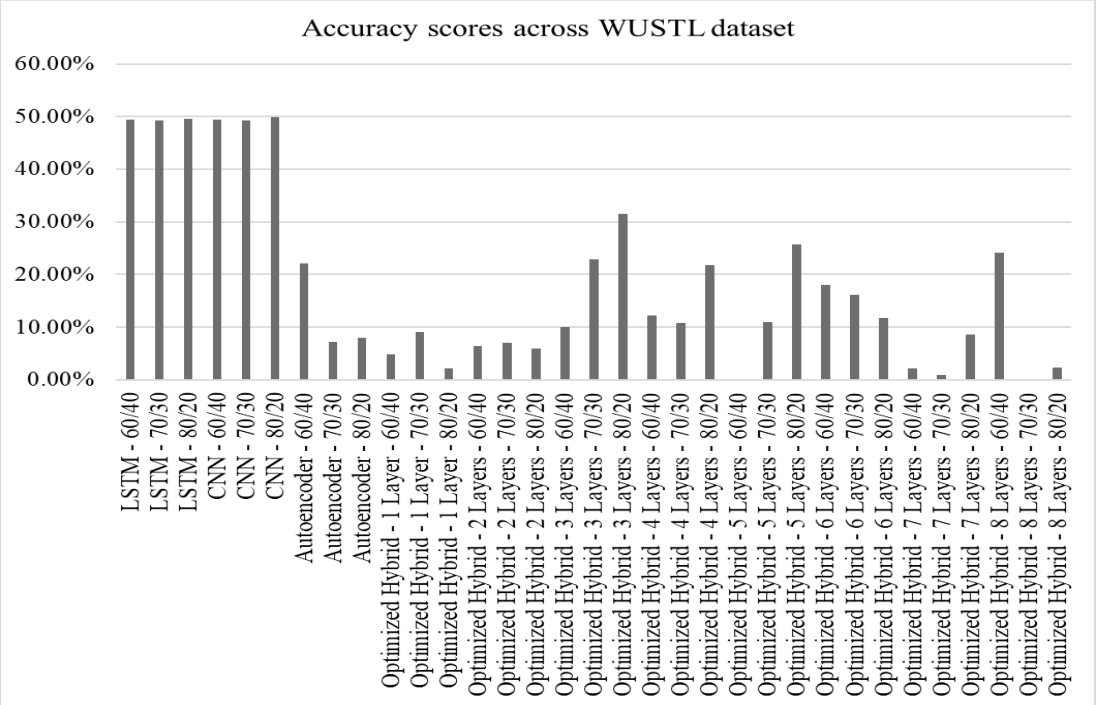


Figure 4.79: Accuracy scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

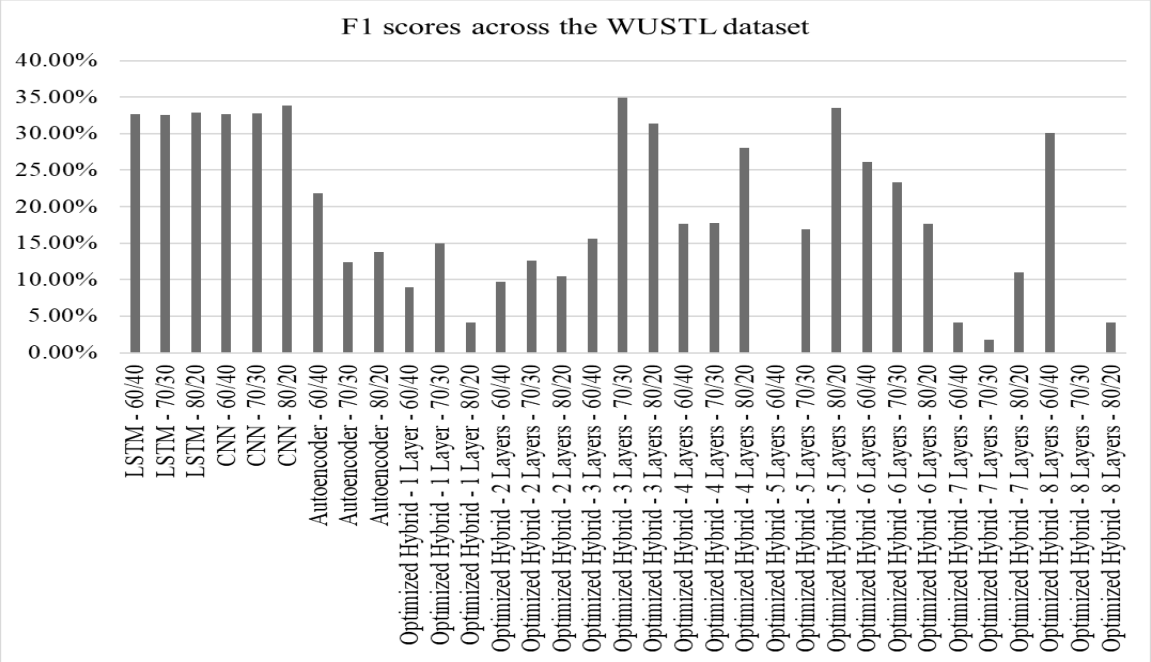


Figure 4.80: F1 scores for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

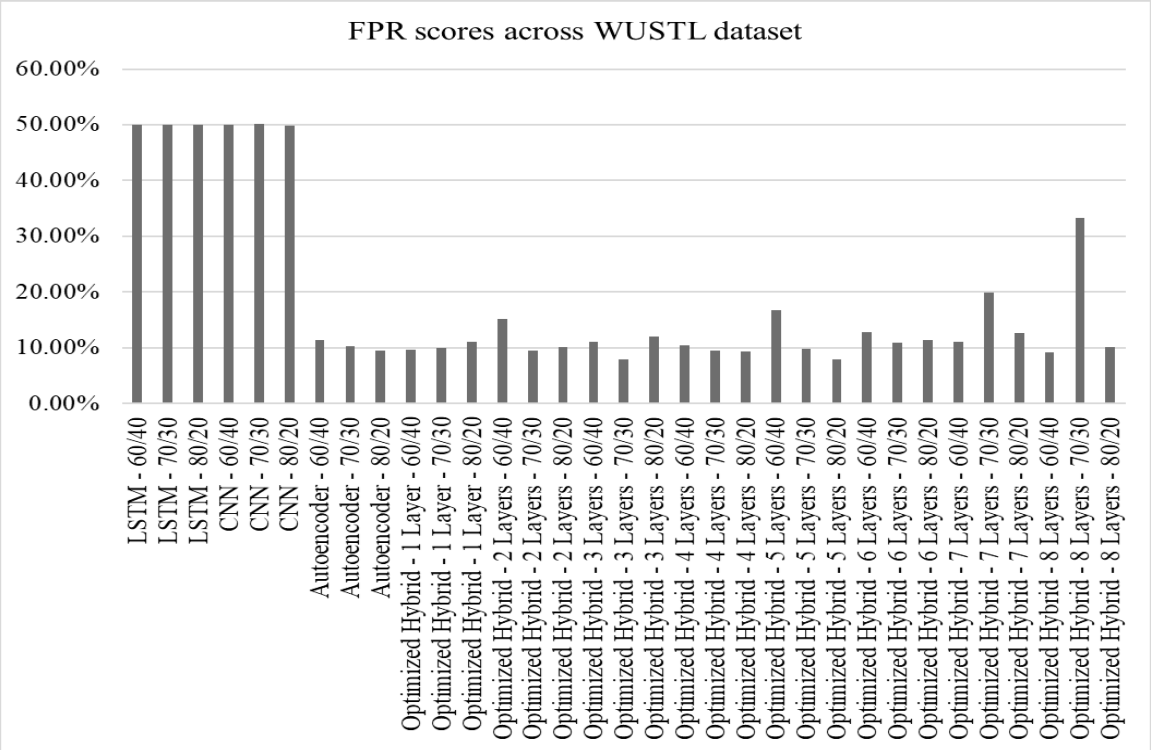


Figure 4.81: False positive rates for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The assessment of inference time for the Nadam-optimized hybrid model against the baseline models on the WUSTL dataset shows the implications of optimization on the

model's detection speed. The LSTM-based models emerged as the fastest, with speeds ranging from 0.19 to 0.21 seconds. Different optimized model variants followed the LSTM, outperforming the autoencoder and the deep autoencoder. These models achieved detection speeds as low as 0.25 seconds. This phenomenon demonstrates that Nadam optimization subtly improved inference speed for the medical IoT dataset. These results are shown in Figure 4.82.

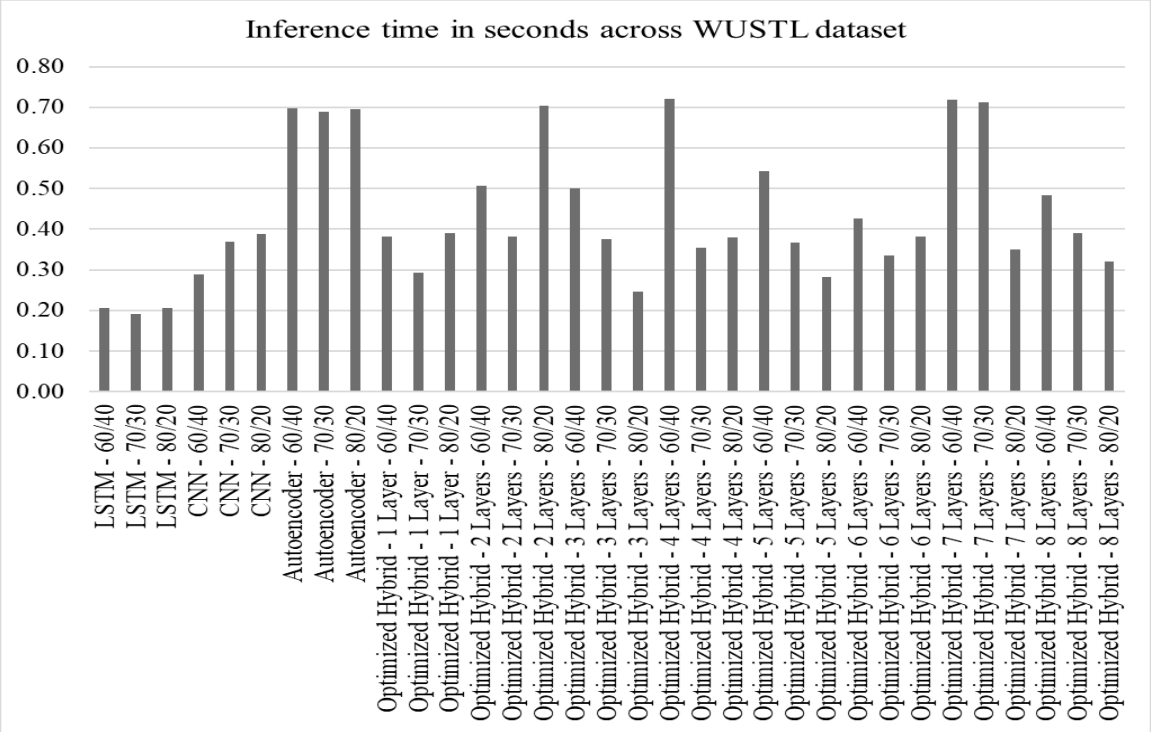


Figure 4.82: Inference time in seconds for the WUSTL dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The analysis of selected Nadam-optimized hybrid models on the WUSTL dataset provides insights into their performance on medical IoT datasets. The models chosen for analysis were the 80/20 variants of the three- and five-hidden-layer models. The 80/20 variant of the three-hidden-layer model achieved an AUC of 0.5 on the ROC curve, indicating no discrimination between threats and benign cases. This significant dip in the model's performance was confirmed by the confusion matrix, which showed 375 false positives and 378 true positives. Also, the model had 35 false negatives against 31 true negatives. The model's testing loss started at about 32 and rose to 60 in the second epoch, before falling to 0 in the third epoch. This loss value rose to 35 and wobbled until it stabilized at 10 in the ninth epoch. The model's validation loss

stabilized at zero until the end of testing. These results are shown in Figures 4.83, 4.84 and 4.85.

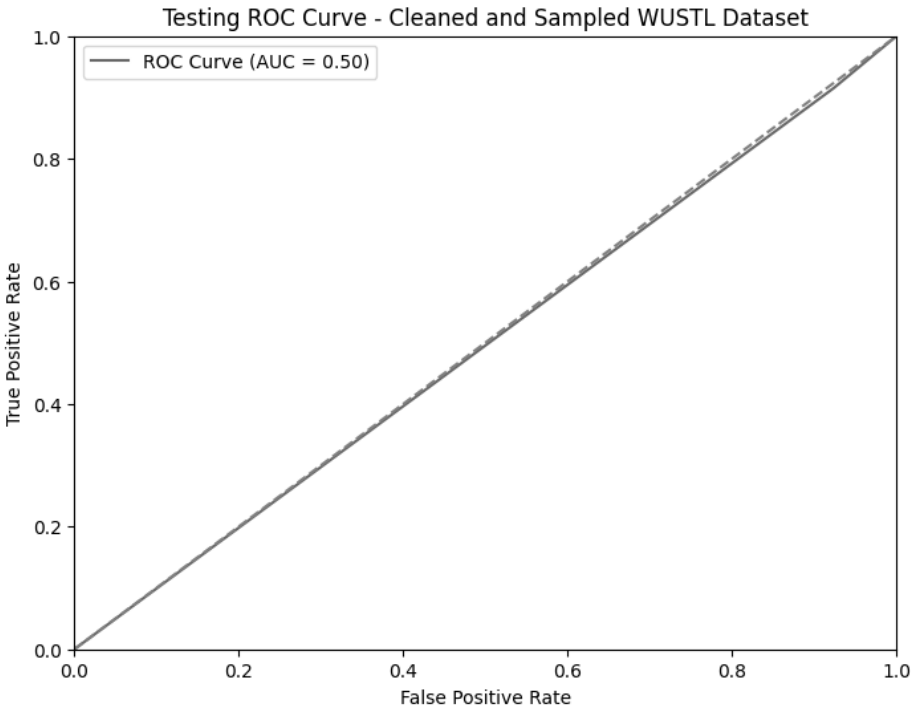


Figure 4.83: Receiver Operating Characteristics curve for three hidden layers' 80/20 Nadam-optimized hybrid model on the WUSTL dataset

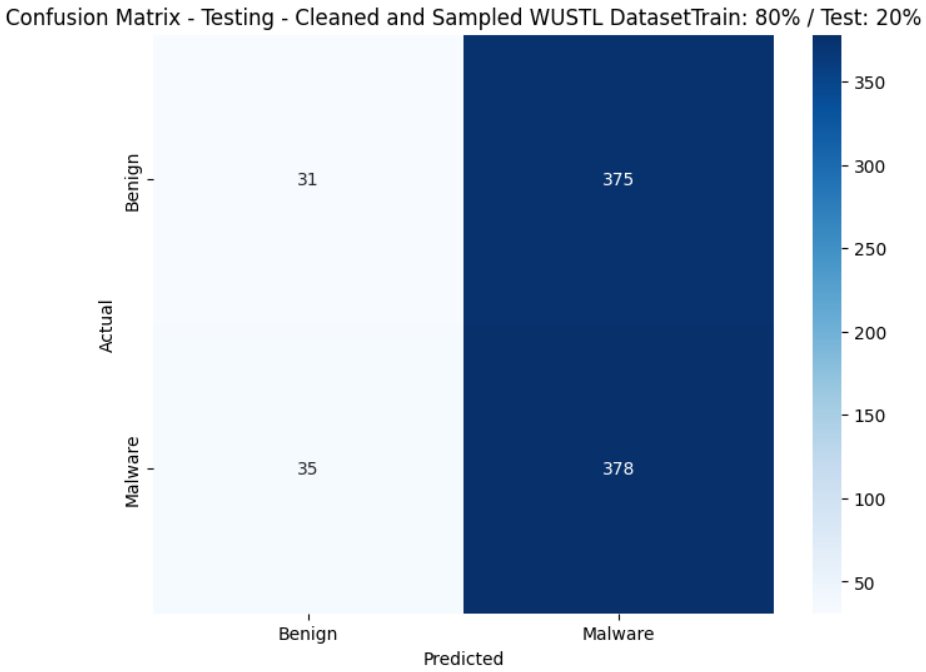


Figure 4.84: Confusion matrix for three hidden layers' 80/20 Nadam-optimized hybrid model on WUSTL dataset

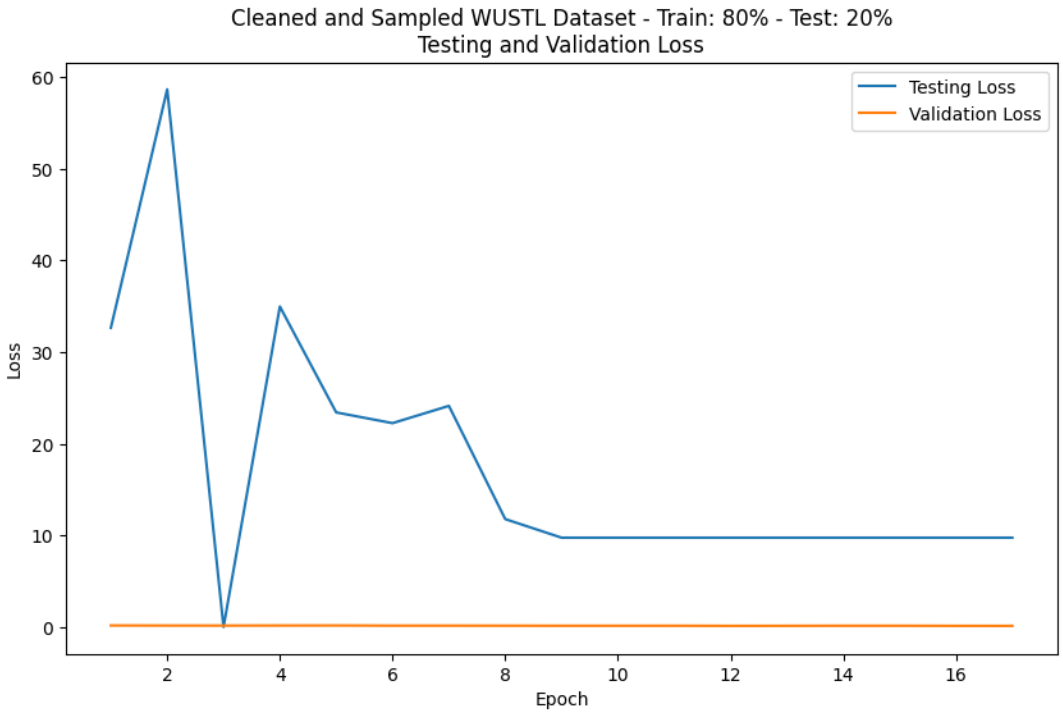


Figure 4.85: Testing and validation loss curves for three hidden layers' 80/20 Nadam-optimized hybrid model on the WUSTL dataset

The assessment of the 80/20 split across five hidden layers shows the model's performance as the number of hidden layers increases, for the same training/testing ratio. The model's AUC score was 0.49, which demonstrated erroneous classification of threats. This weak score was confirmed by the confusion matrix, which showed 401 false positives, 12 false negatives, 401 true positives, and five true negatives. The model's test loss started at 40 and fell to about 13 in the second epoch. This score rose to 30 in the third epoch, then fell to 5 in the fifth, before rising to about 25 in the tenth and stabilizing at this level until the end of the testing session. The model's validation loss stabilized at zero until the end the session. These results are shown in Figures 4.86, 4.87 and 4.88.

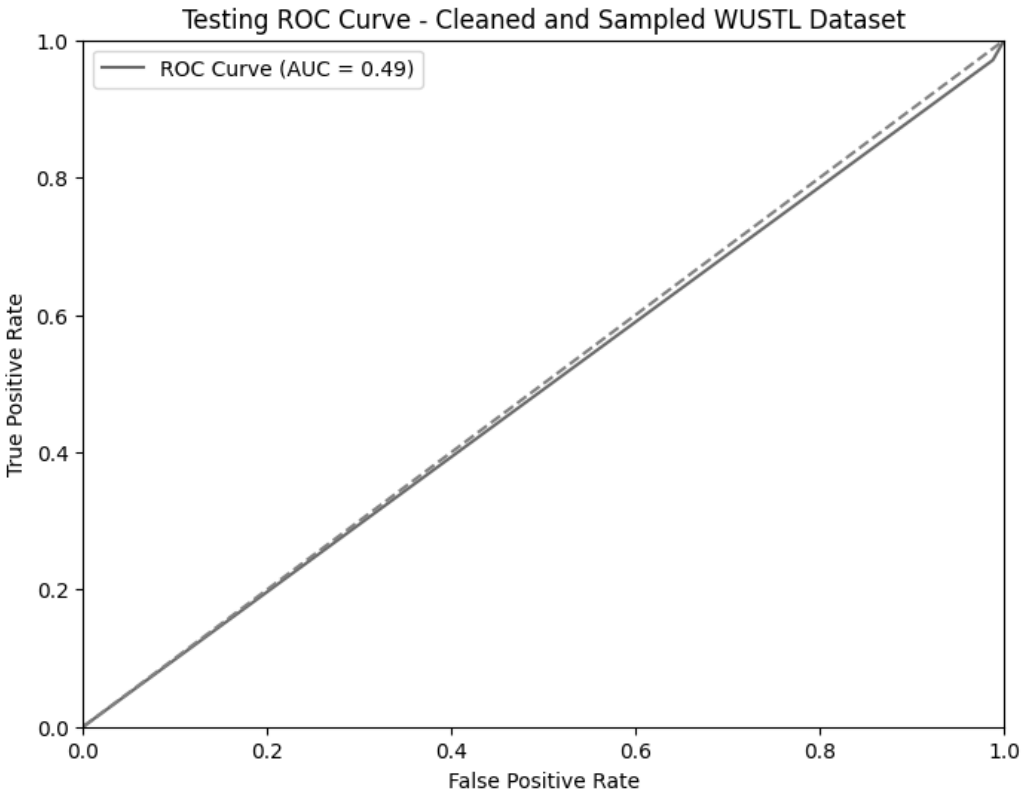


Figure 4.86: Receiver Operating Characteristics curve for five-hidden-layers' 80/20 Nadam-optimized hybrid model on the WUSTL dataset

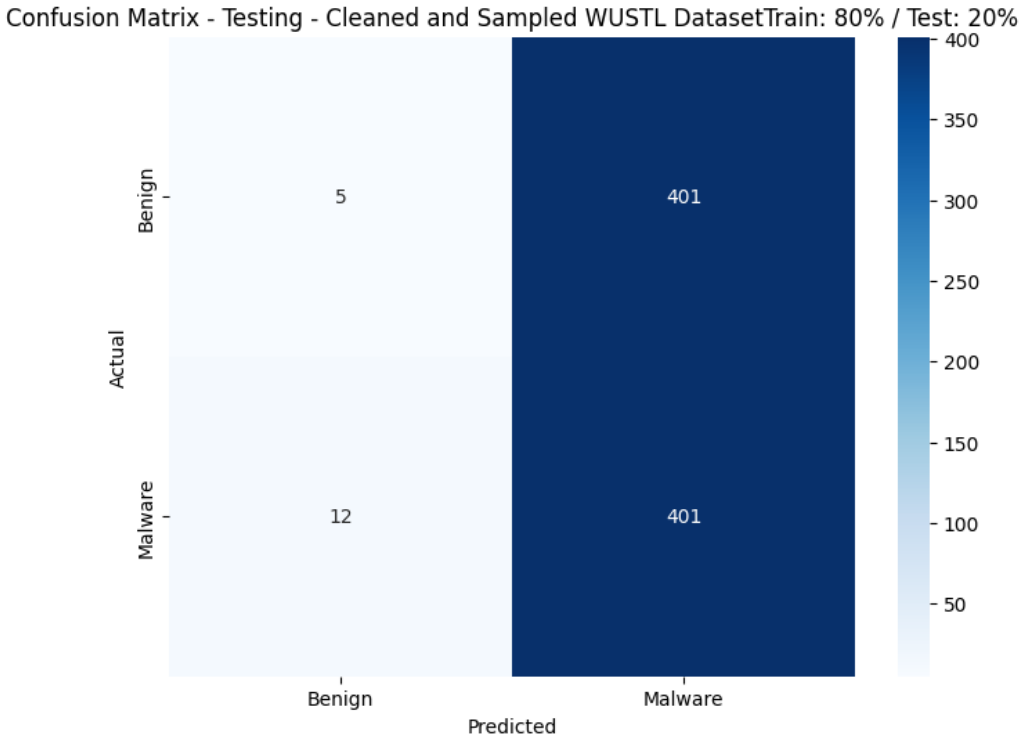


Figure 4.87: Confusion matrix for five-hidden-layers' 80/20 Nadam-optimized hybrid model on WUSTL dataset

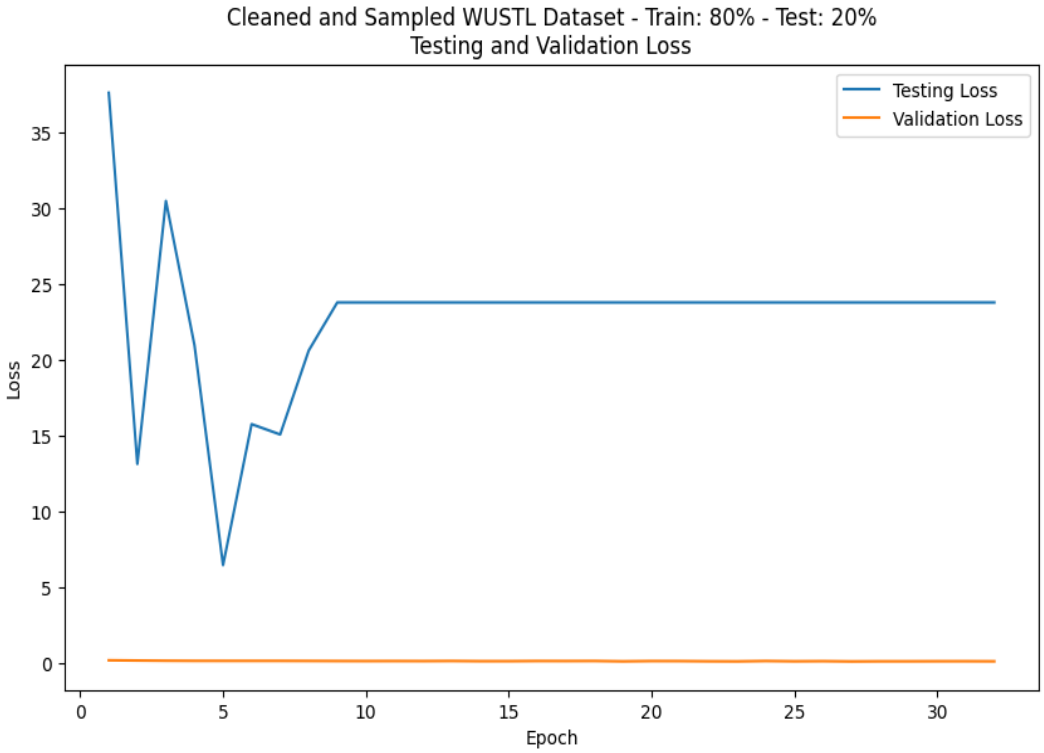


Figure 4.88: Testing and validation loss curves for five-hidden-layers' 80/20 Nadam-optimized hybrid model on WUSTL dataset

The review of training loss curves for the selected models offers insights into their testing performance. For instance, the loss curve for the 80/20 variant with three hidden layers started at 0.205 and fell to 0.18 in the second epoch. This value wobbled downwards, reaching 0.15 by the 17th epoch. The validation loss curve started at 0.185, rose slightly to 0.155 in the fifth epoch, and then wobbled downwards to 0.14 by the 17th epoch. These results suggest that the model may have changed slightly with additional training epochs. The loss curve for the five-hidden-layers' 80/20 variant started at 0.22, while the validation loss began at 0.21. The two curves wobbled downwards, with the training loss curve ending at 0.13 and the validation loss at 0.14. These results indicate that the model might have improved slightly with additional training. These results are shown in Figures 4.89 and 4.90.

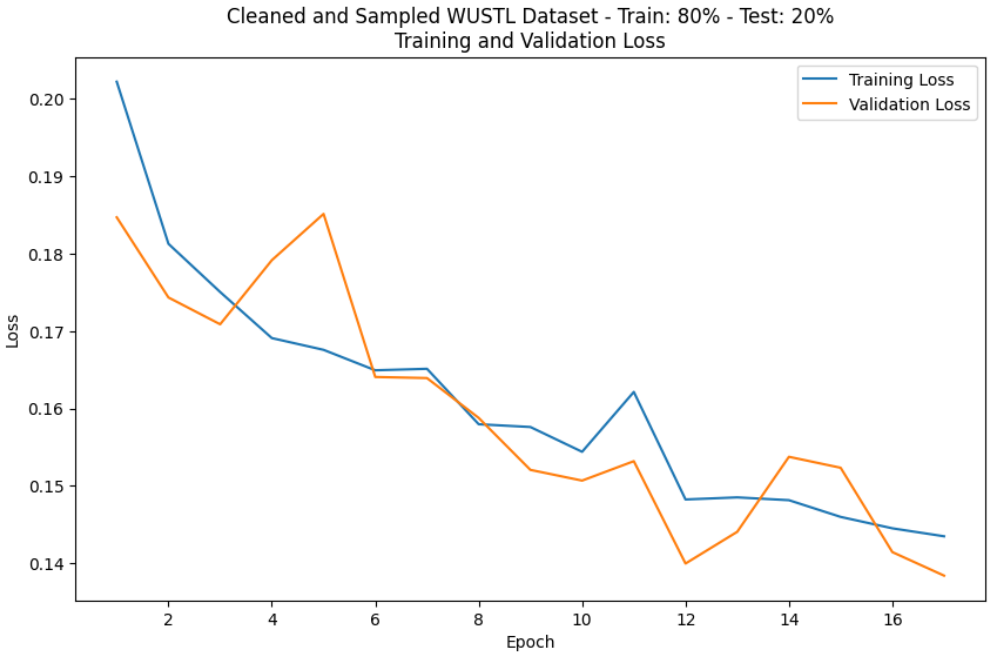


Figure 4.89: Training and validation loss curves for three hidden layers' 80/20 Nadam-optimized hybrid model on the WUSTL dataset

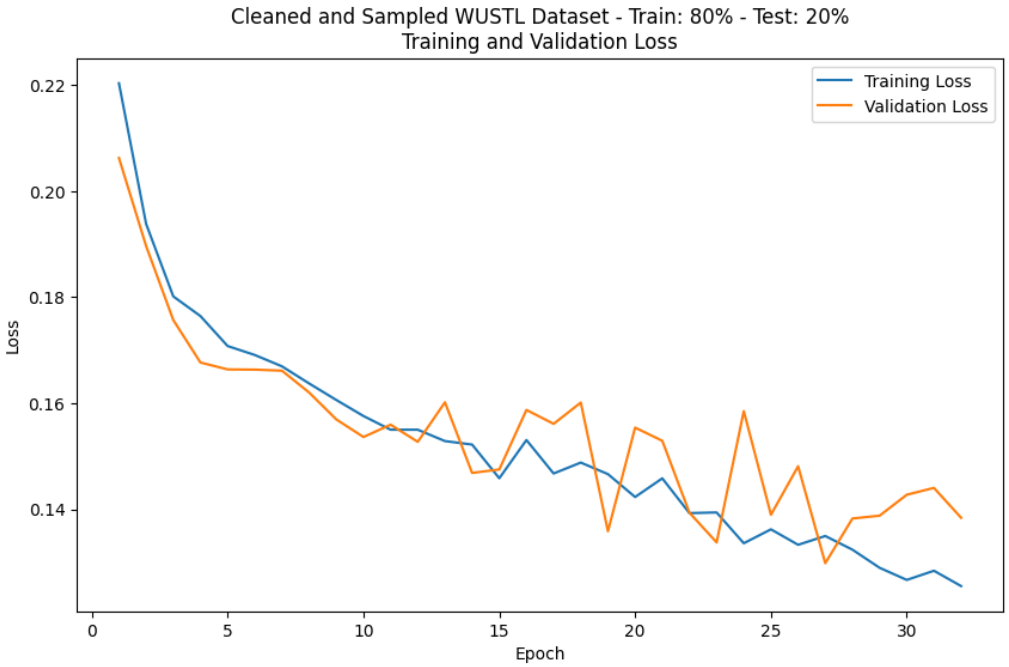


Figure 4.90: Training and validation loss curves for a five-hidden-layer' 80/20 Nadam-optimized hybrid model on the WUSTL dataset

4.4.3 Nadam-Optimized Hybrid Model on Edge IIoT Dataset

The analysis of recall scores for the Edge IIoT dataset, comparing the Nadam-optimized model with baseline models, demonstrates the performance of the hybrid models with optimization. The baseline models dominated the recall scores, with the CNN and LSTM variants topping the list. The 80/20 variant of the seven hidden layers emerged as the best hybrid model, but it scored 26.22%. This phenomenon demonstrates that the optimization failed to meet the desired goals. These results are shown in Figure 4.91.

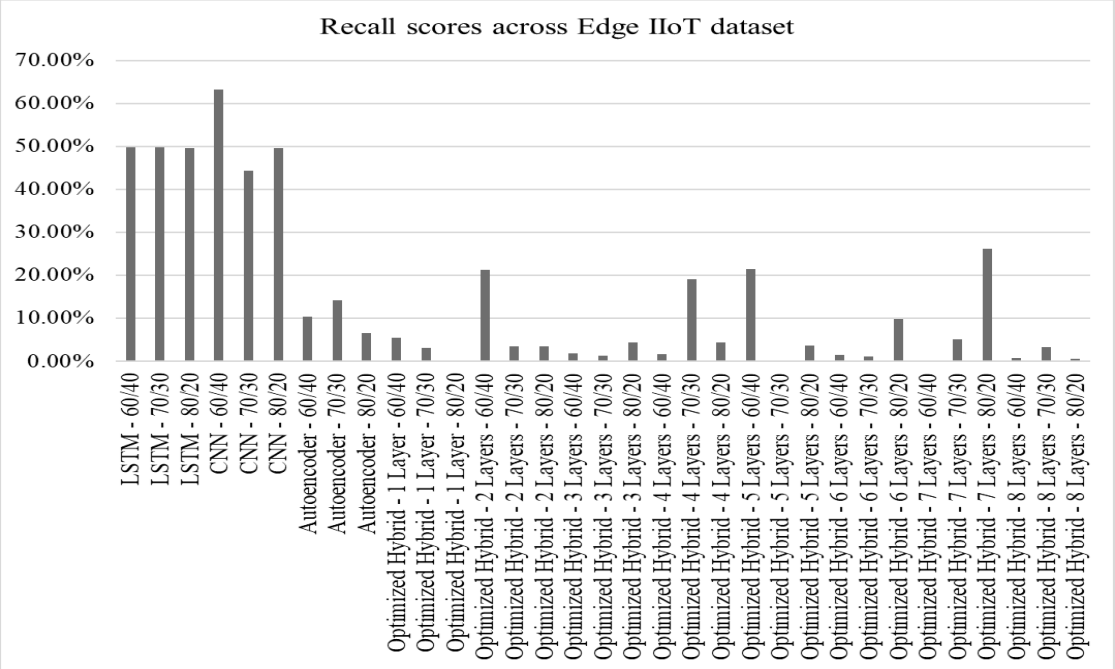


Figure 4.91: Recall scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The review of precision scores for the Edge IIoT dataset on Nadam-optimized hybrid and baseline models demonstrates the impact of optimization on threat identification. Different variants of the hybrid model dominated the top scores, with the six-hidden-layer 80/20 variant leading at 92.08%. The five-hidden-layers' 60/40 had a score of 90.09% and the two-hidden-layers' 70/30 variant scored 81.83%. The ability of these hybrid models to achieve the highest precision scores demonstrates that Nadam optimization was beneficial. These results are shown in Figure 4.92.

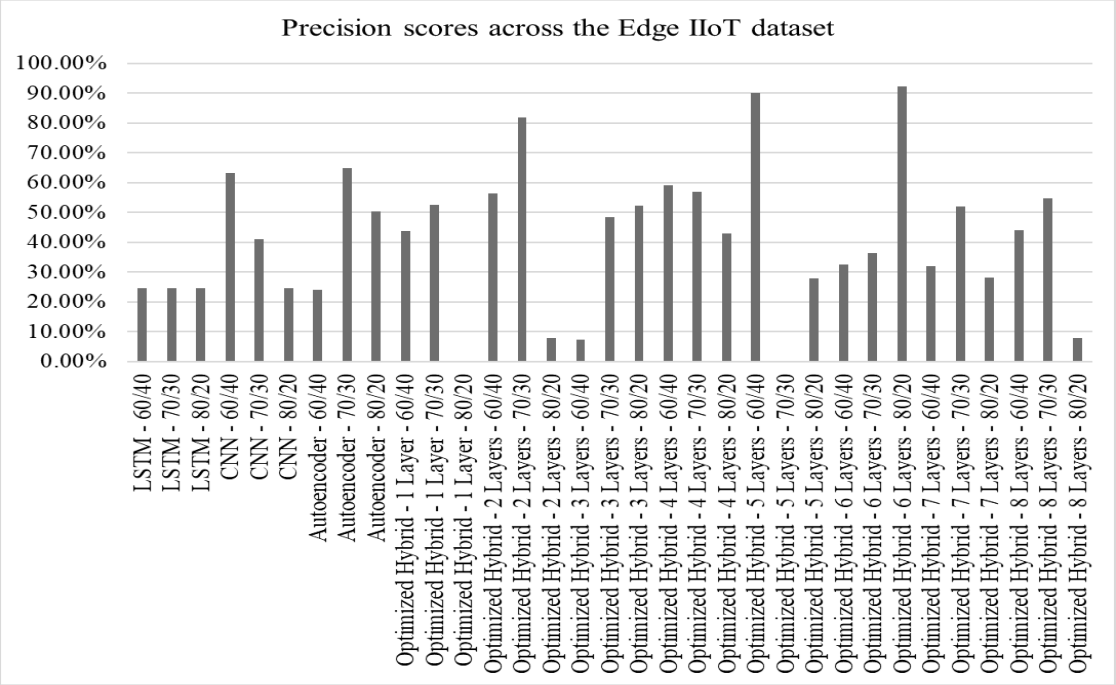


Figure 4.92: Precision scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The review of specificity scores for the Edge IIoT dataset on the Nadam-optimized hybrid and baseline models shows the impacts of optimization on these models. The hybrid models dominated the top of the list, achieving over 90%. For instance, the two-hidden-layers' 60/40 variant recorded 92.03%, while the five-hidden-layers' 60/40 variant had 92.01%. The lower bottom was dominated by the baseline models, demonstrating the significant improvement of hybrid models with optimization. These results are shown in Figure 4.93.

The assessment of accuracy scores for the Edge IIoT dataset on Nadam-optimized hybrid and baseline models demonstrates the models' general correctness. The baseline models emerged as the dominant variants in this category, with the CNN's 60/40 variant scoring 63.25%. The hybrid models lagged, with the best model being a seven-layer 80/20 variant at 26.22%. This phenomenon demonstrates that Nadam optimization did not improve the model's accuracy. These results are shown in Figure 4.94.

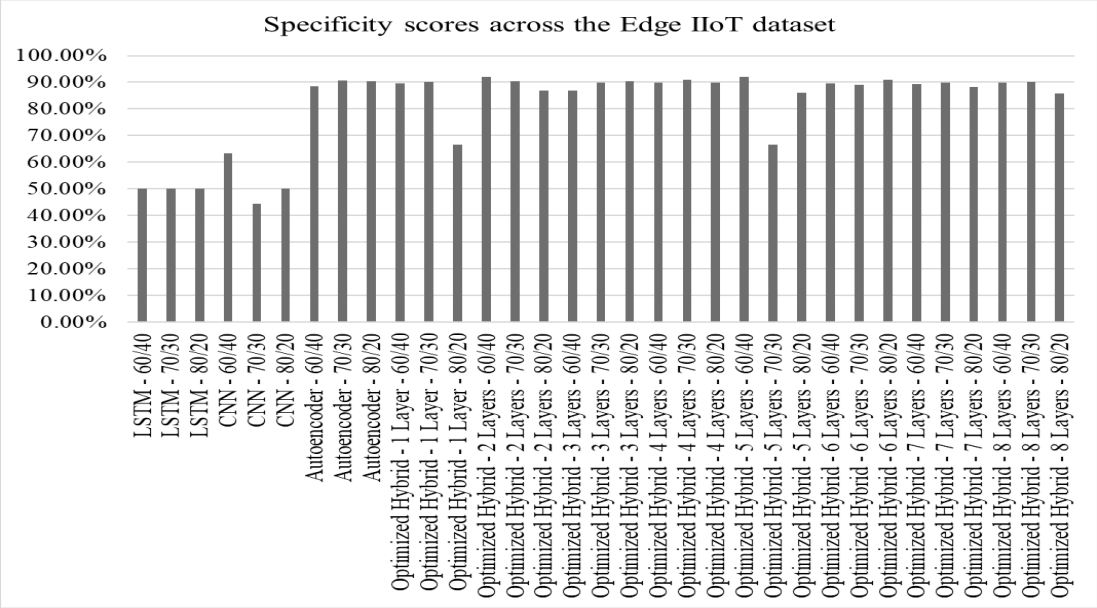


Figure 4.93: Specificity scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

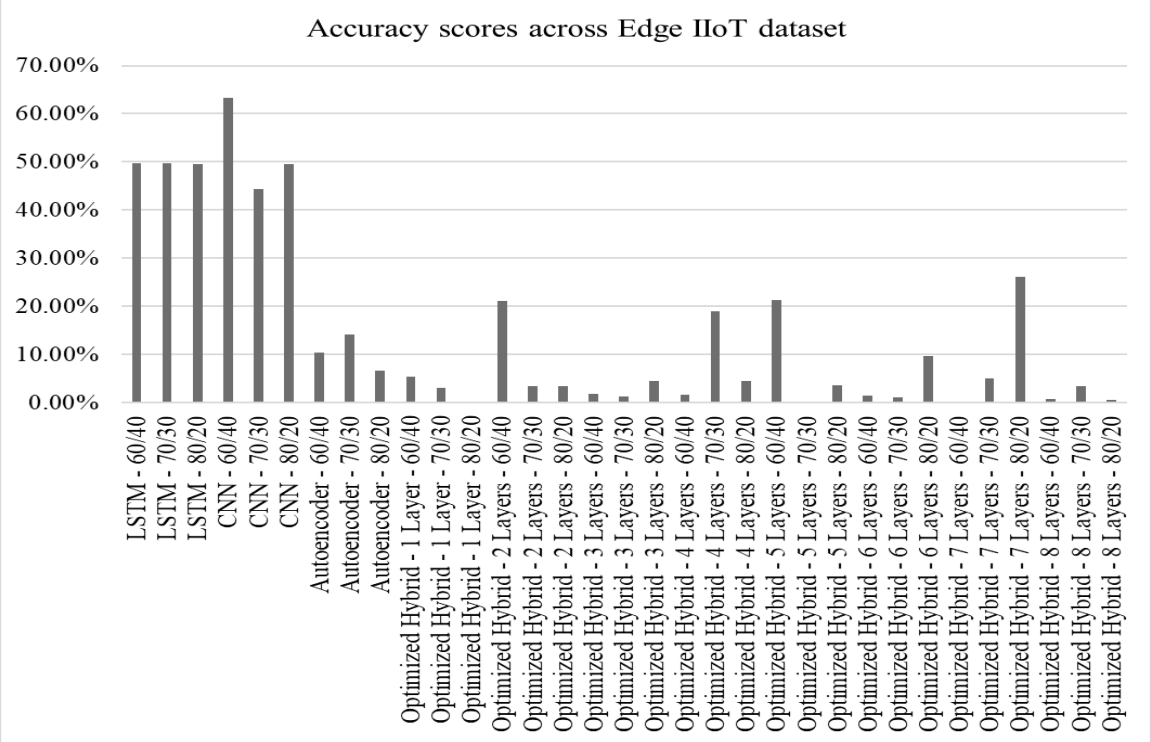


Figure 4.94: Accuracy scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The evaluation of F1 scores for the Edge IIoT dataset on Nadam-optimized hybrid and baseline models shows that these models balance recall and precision. The baseline

models dominated this balance, with the CNN's 60/40 variant scoring 63.25%. While other models failed to reach 50%, the hybrid models performed worse. The best hybrid model was the five-layer 60/40 variant, which scored 30.07%. This situation demonstrates that the Nadam optimization was ineffective, underscoring the need for alternative enhancement strategies. These results are shown in Figure 4.95.

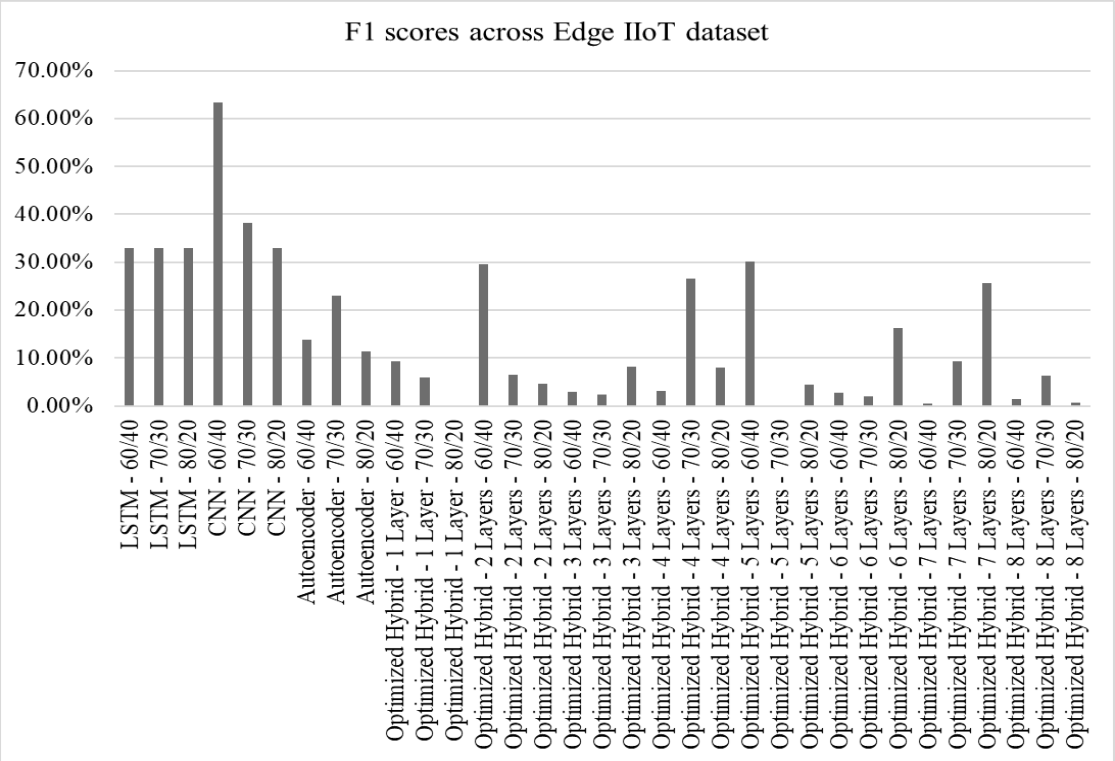


Figure 4.95: F1 scores for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The assessment of the false-positive rate for the Edge IIoT dataset using baseline and Nadam-optimized hybrid models shows the impact of optimization on these models. The hybrid models dominated the list, with the two-hidden-layers' 60/40 variant topping with 7.97%. Other hybrid models like five-hidden-layers' 60/40, four-hidden-layers' 70/30, and six-hidden-layer 80/20 variants scored 7.99%, 9.01% and 9.18%. The baseline models dominated the lower end of the FPR scores, indicating that the optimization was effective in reducing the error rate. These results are shown in Figure 4.96.

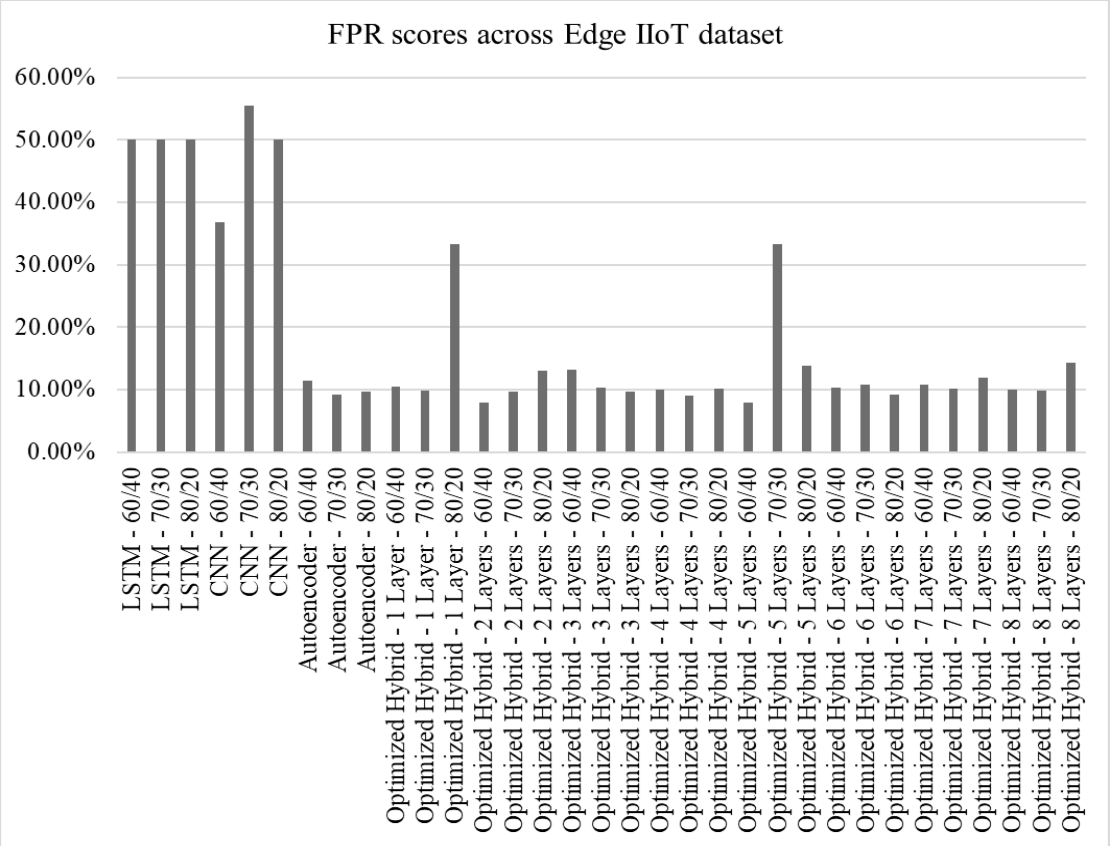


Figure 4.96: False positive rates for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The analysis of inference time on the Edge IIoT dataset for the Nadam-optimized hybrid and baseline models shows the effect of optimization on detection speed. The baseline models dominated the fast inference, with models like CNN's 80/20 variant leading with 1.34 seconds. Hybrid models lagged behind the baseline models, indicating that optimization was ineffective at improving detection speeds. These results are shown in Figure 4.97.

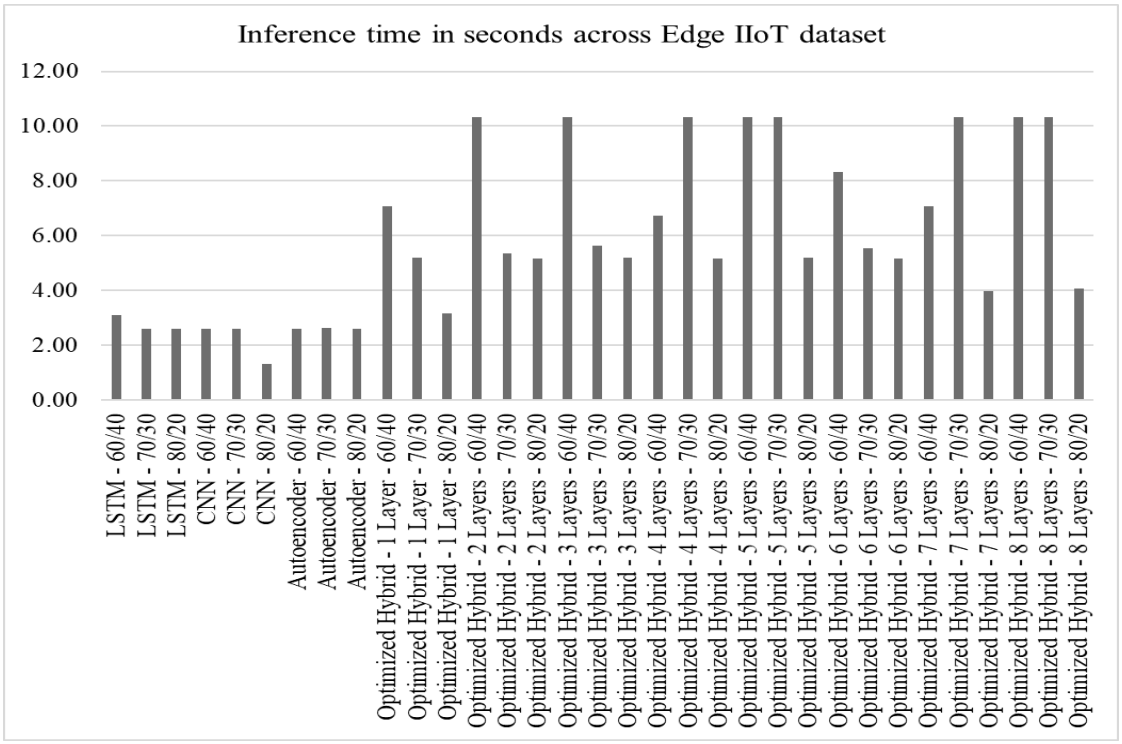


Figure 4.97: Inference time in seconds for the Edge IIoT dataset across the LSTM, CNN, Autoencoder, and Nadam-optimized Hybrid deep autoencoder model variants

The analysis of Nadam-optimized hybrid models on the Edge IIoT dataset provides insights into the model's performance with optimization. Three models were selected for this analysis: the 80/20 variant with seven hidden layers and the 60/40 variants with five and two hidden layers. The 80/20 split across the seven hidden layers had an AUC of 0.42 on the ROC curve, indicating weak performance in classifying threats. This performance was confirmed by the confusion matrix, which showed the model recorded 6,482 false negatives and 4,849 true negatives. Further, there were 5,073 false positives against 3,596 true positives. The assessment of the model's loss curves provides more insights into this behavior. The model's testing loss started at 15, fell to about 2 in the second epoch, then rose to 35 in the fourth epoch. It fell to 15 in the sixth epoch before rising to 50 in the eighth. It then fell to 20 and stabilized at this level until the end of the testing session. The model's validation loss remained stable at zero until the end of the session. These results are shown in Figures 4.98, 4.99 and 4.100.

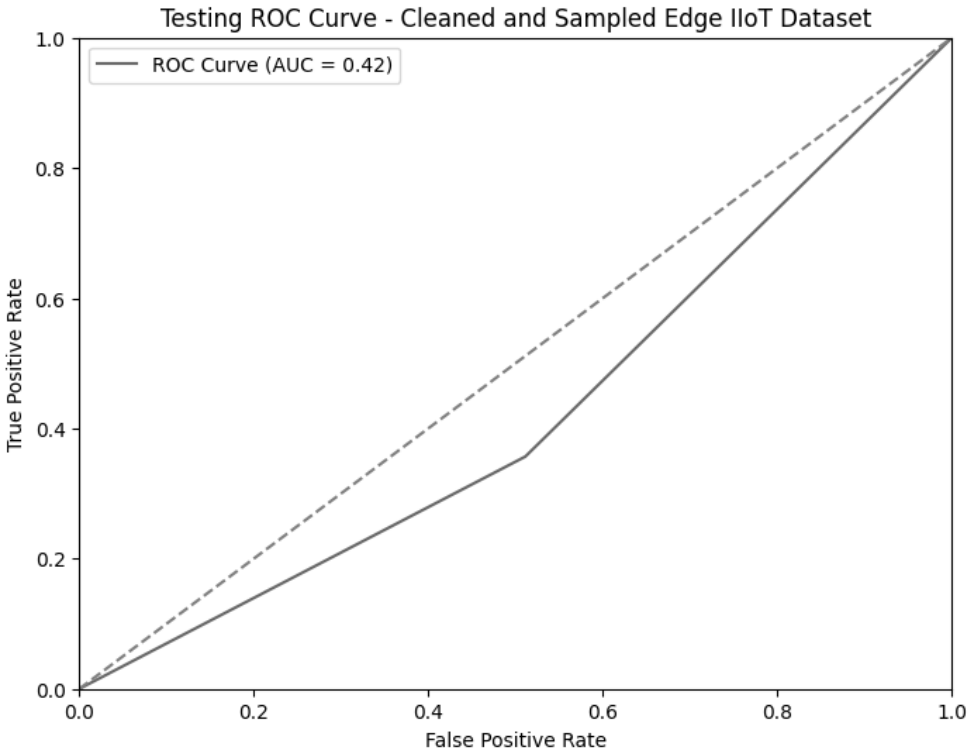


Figure 4.98: Receiver Operating Characteristics curve for seven hidden layers' 80/20 Nadam-optimized hybrid model on Edge IIoT dataset

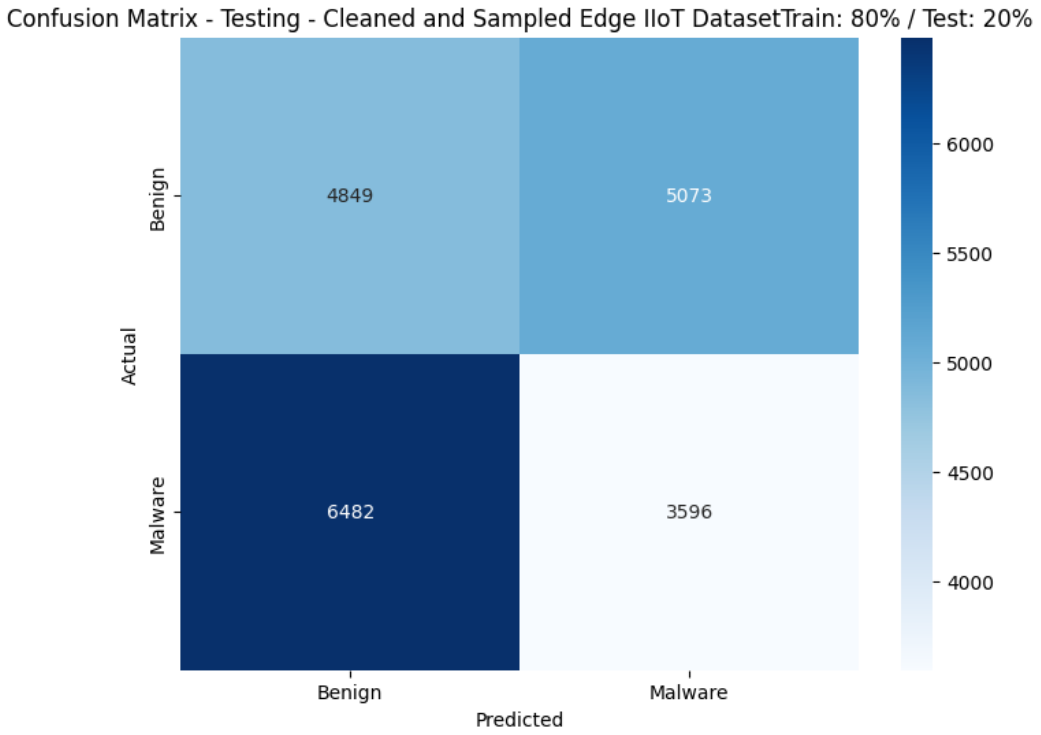


Figure 4.99: Confusion matrix for seven hidden layers' 80/20 Nadam-optimized hybrid model on Edge IIoT dataset

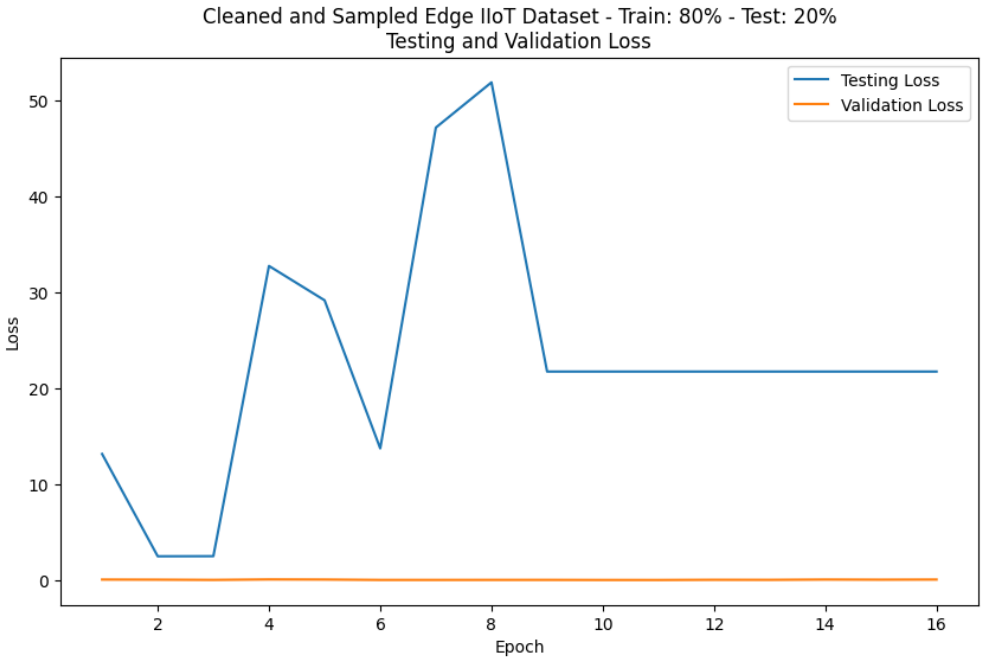


Figure 4.100: Testing and validation loss curves for seven hidden layers' 80/20 Nadam-optimized hybrid model on Edge IIoT dataset

The analysis of the five-hidden-layers' 60/40 variant offers insights into the model's performance in this training/testing ratio. The model achieved an AUC of 0.57 on the ROC curve, indicating weak separation between threats and benign cases, but this was higher than that observed in the 80/20 variant with seven hidden layers. Additionally, the model recorded 11,578 false positives against 19,669 true positives, and 458 false negatives against 8,295 true negatives. The model's validation loss curve maintained a score of 0 throughout the testing session, while the testing loss started at 40, fell to about 12, then rose to 30. It fell to 5, and rose until 20, where it stabilized until the end of the testing session. These results are shown in Figures 4.101, 4.102, and 4.103.

The review of the two-hidden-layer 60/40 variant shows the impact of Nadam optimization on the model, with a reduction in hidden layers for the 60/40 training/testing ratio. The model had an AUC of 0.5 on the ROC curve, indicating non-discriminatory classification of threats. There were 19,851 false positive cases against 19,997 true positives, and 130 false negatives against 22 true negatives. The model's validation loss curve maintained a score of 0 until the end of the session, while the testing loss curve started at about 22 before falling to 8 in the second epoch and then rose to 20 in the fourth epoch. It wobbled until the end of the testing session, slightly

stabilizing at about 18 in the ninth and tenth epochs. These results are shown in Figures 4.104, 4.105, and 4.106.

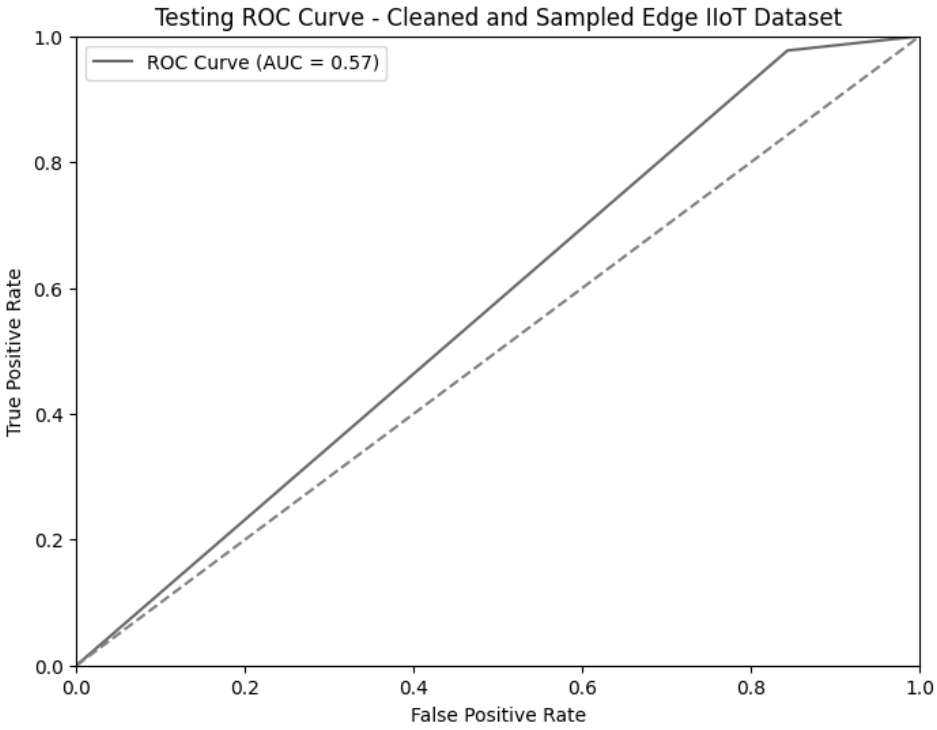


Figure 4.101: Receiver Operating Characteristics curve for five-hidden-layers' 60/40 Nadam-optimized hybrid model on Edge IIoT dataset

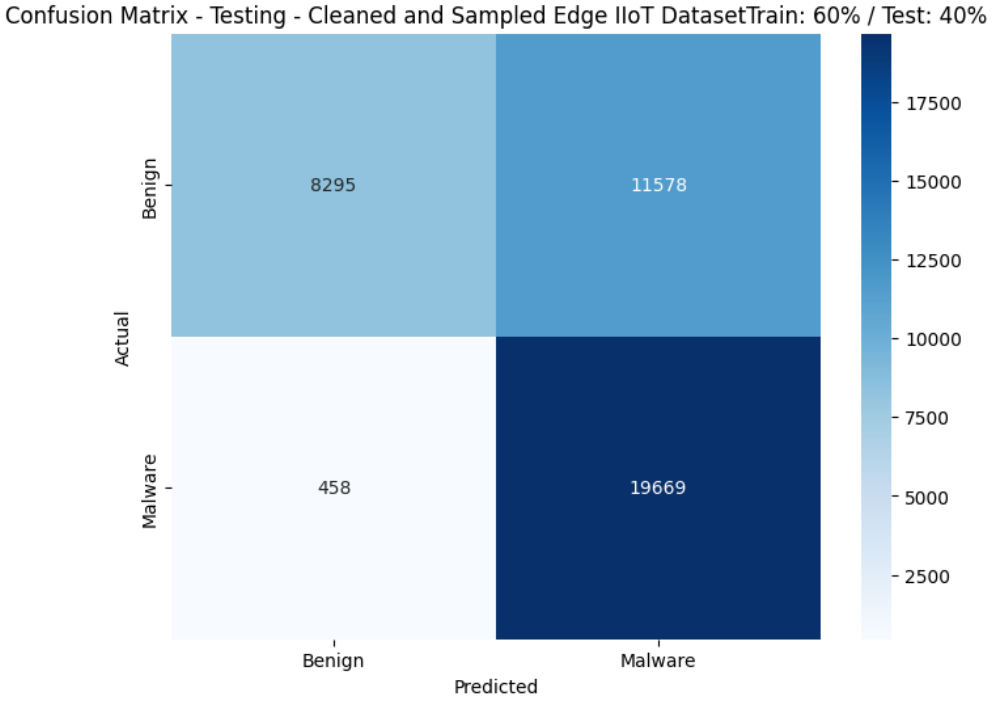


Figure 4.102: Confusion matrix for five-hidden-layers' 60/40 Nadam-optimized hybrid model on Edge IIoT dataset

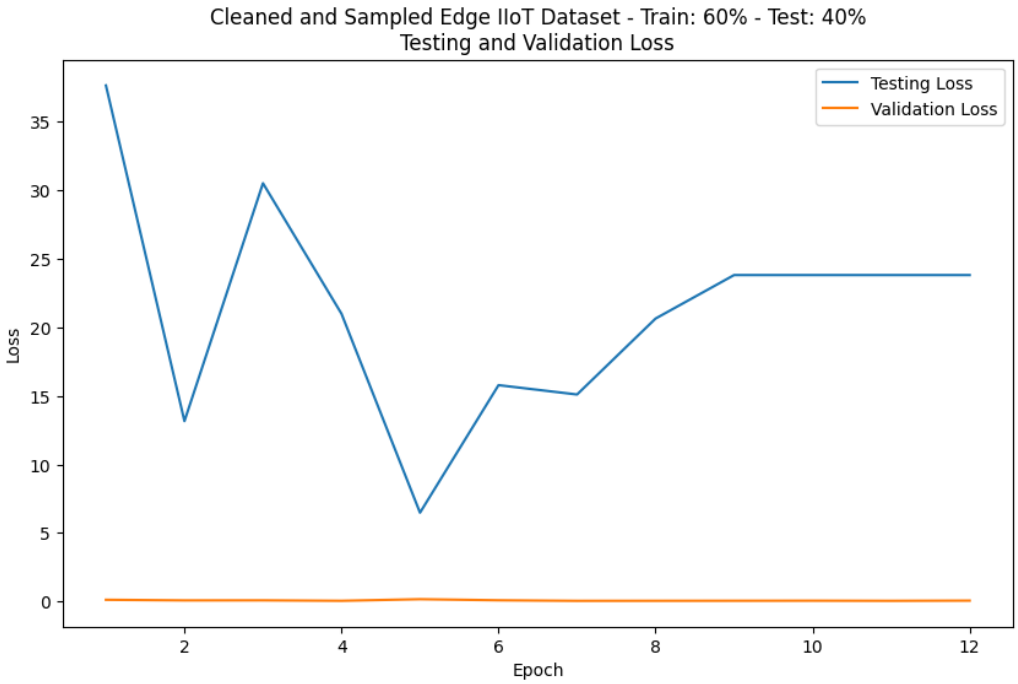


Figure 4.103: Testing and validation loss curves for five-hidden-layers' 60/40 Nadam-optimized hybrid model on Edge IIoT dataset

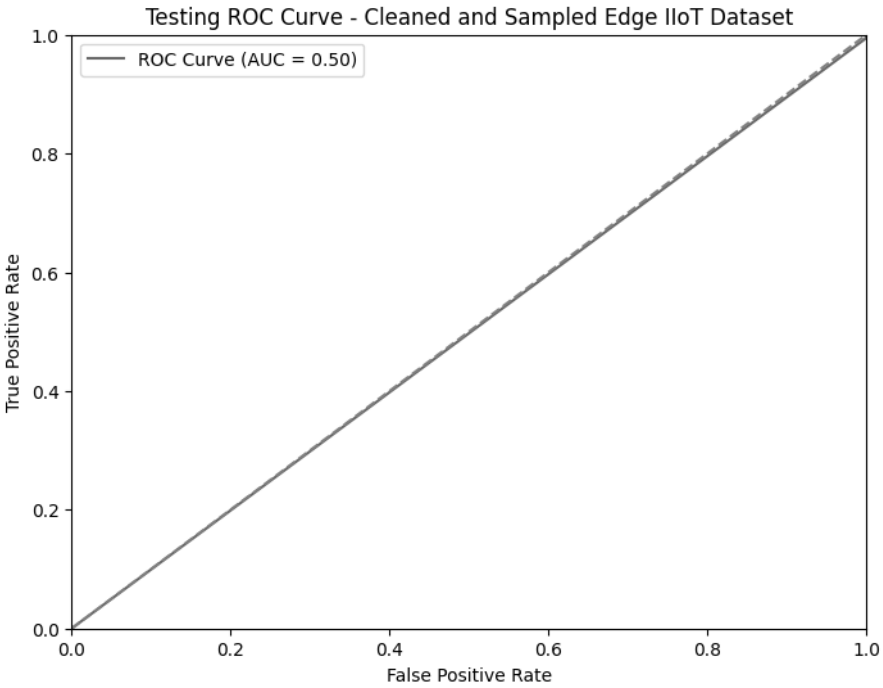


Figure 4.104: Receiver Operating Characteristics curve for a two-hidden-layer' 60/40 Nadam-optimized hybrid model on the Edge IIoT dataset

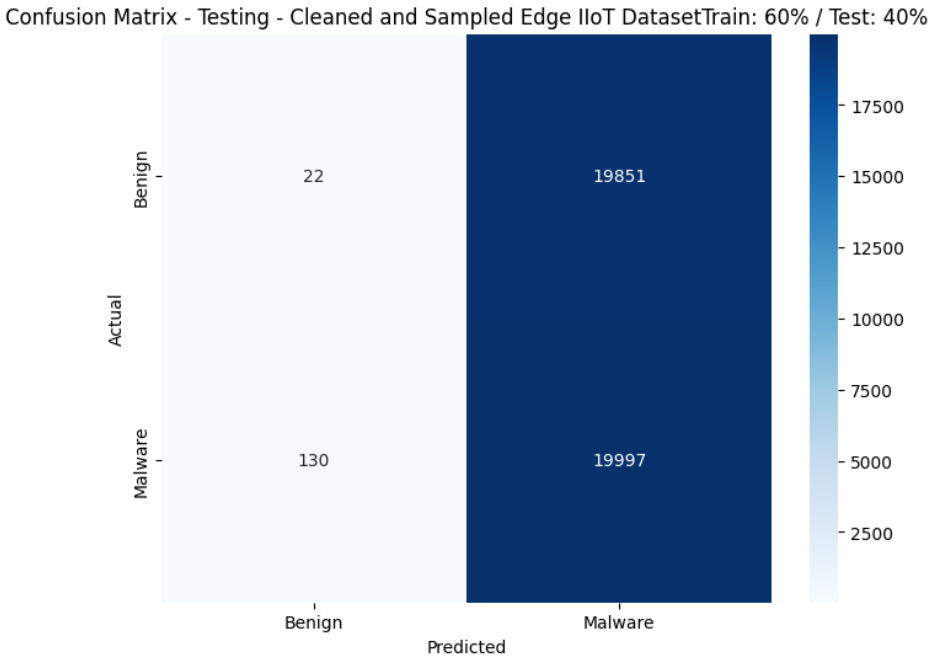


Figure 4.105: Confusion matrix for two-hidden-layers' 60/40 Nadam-optimized hybrid model on Edge IIoT dataset

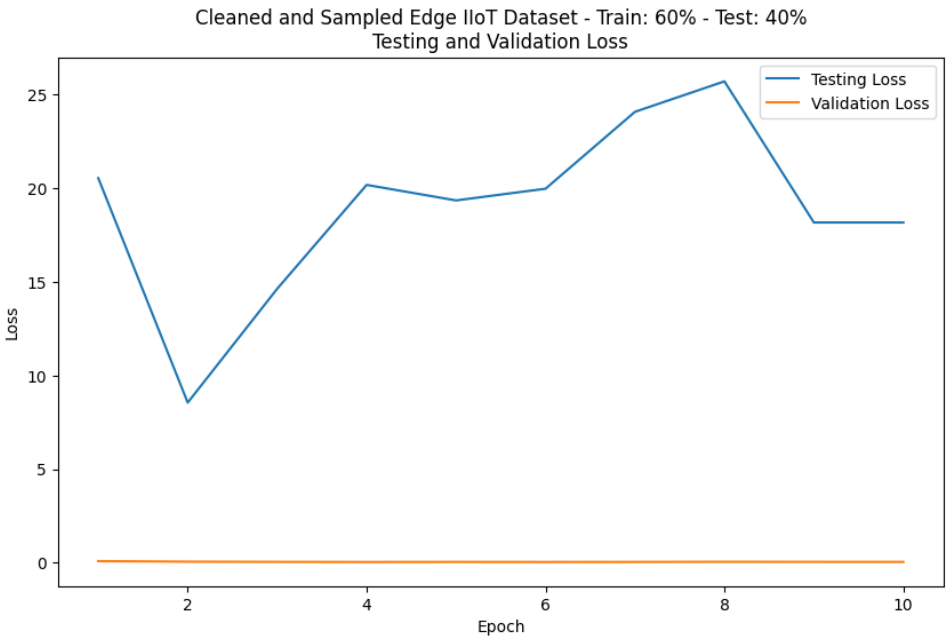


Figure 4.106: Testing and validation loss curves for a two-hidden-layer' 60/40 Nadam-optimized hybrid model on the Edge IIoT dataset

The review of training loss curves for the selected models enhances understanding of their testing performance. For example, the loss curve for the 80/20 model with seven hidden layers started at 0.115, wobbled downwards to 0.05 in the 10th epoch, then wobbled upwards to 0.095 in the 16th epoch. The validation loss started at 0.085, fell to

0.05 in the third epoch, and rose to 0.10 in the fourth epoch. It then fell to 0.045 in the sixth epoch, and wobbled upwards until 0.085 in the 17th epoch. This performance suggests that the model might have improved significantly with additional training epochs, as they would have stabilized it.

The training loss curve for the five-hidden-layers' 60/40 variant provides a slightly different experience. The training loss curve started at 0.16 and wobbled downwards until it reached 0.07 in the 12th epoch. The model's validation loss curve began at 0.13, fell to 0.06 in the fourth epoch, then rose to 0.17 in the fifth, and fell to 0.05 in the seventh. It then wobbled until the end of the training session. This suggests that the model could have stabilized with a few additional epochs, though overall performance would have changed slightly. The training loss curve for the two-hidden-layer 60/40 model started at 0.105, while the validation loss began at 0.09. The two curves wobbled downwards, ending in the tenth epoch, with the validation loss at 0.05 and the training loss at 0.06. This phenomenon suggests that the model might have improved with additional training epochs, resulting in better scores. These results are shown in Figure 4.107,108 & 109

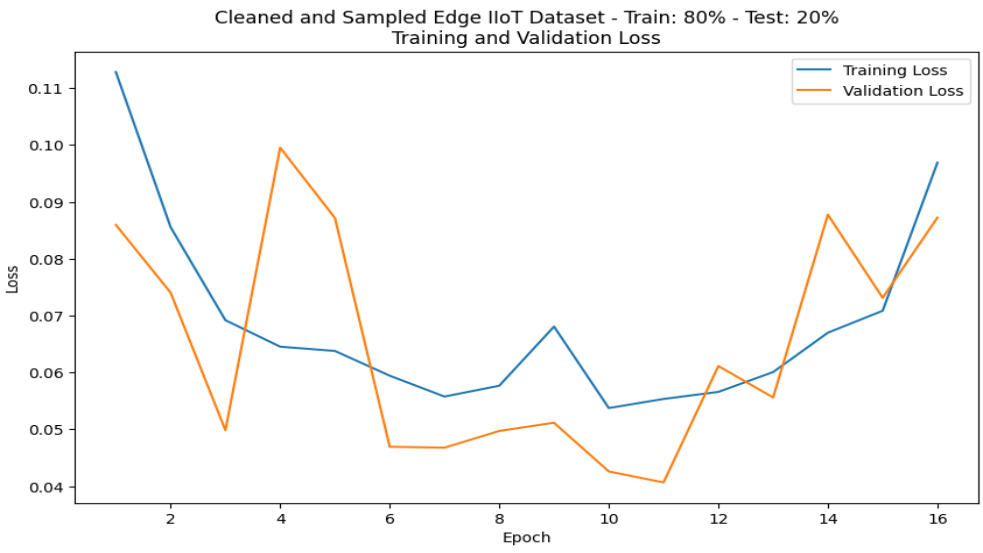


Figure 4.107: Training and validation loss curves for seven hidden layers' 80/20 Nadam-optimized hybrid model on Edge IIoT dataset

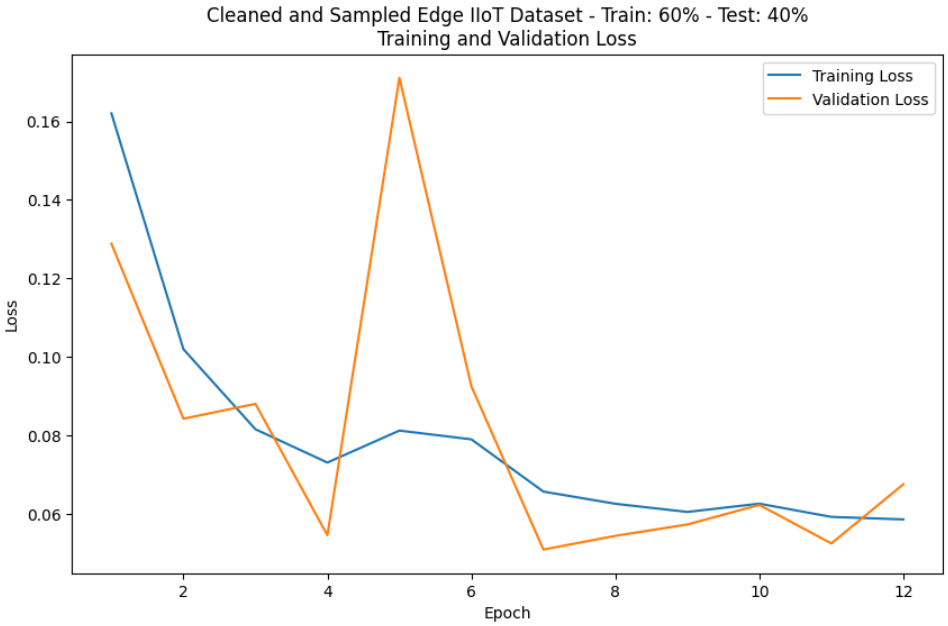


Figure 4.108: Training and validation loss curves for five-hidden-layers' 60/40 Nadam-optimized hybrid model on Edge IIoT dataset

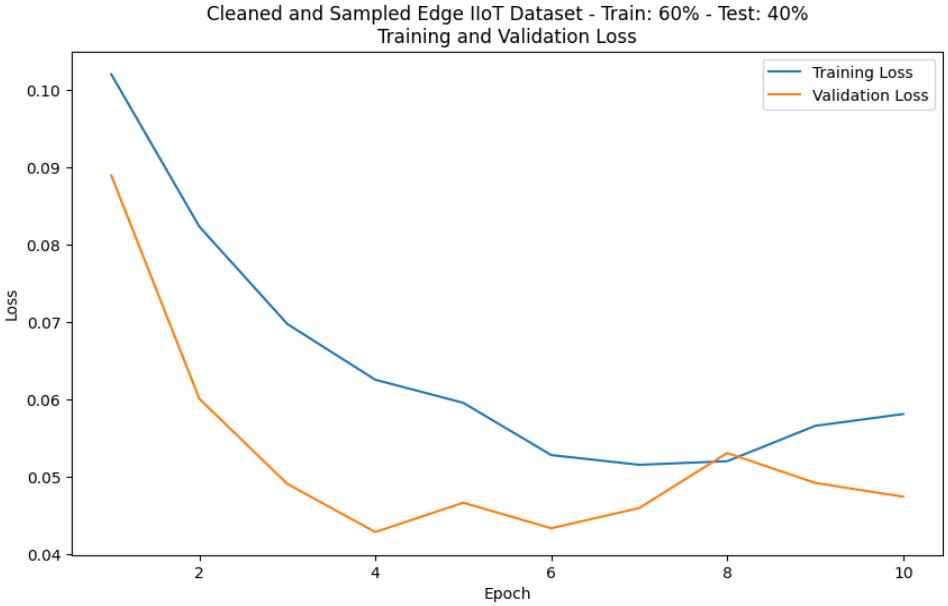


Figure 4.109: Training and validation loss curves for a two-hidden-layer' 60/40 Nadam-optimized hybrid model on the Edge IIoT dataset

4.5 Comparative Results for Machine Learning Models

Although Nadam optimization had been implemented in the Deep Autoencoder model, substantial performance setbacks compromised its adoption at this stage. For instance, three three-hidden-layer 80/20 ratios recorded the best scores for the model. This configuration had a recall and accuracy of 99.95%, precision and specificity of 99.98%,

an F1 score of 99.97% and a false positive rate of 0.02. Datasets such as WUSTL and Edge IIoT lacked a clear model configuration, indicating inefficiencies in Nadam optimization in refining the proposed model. As a result, there was a need to consider alternative approaches to improve the model's performance. One such possible integration was machine learning models. Preliminary analyses of random forests, multilayer perceptrons, and gradient boosting conducted in previous studies showed promising performance in the Medical IoT sector (Kirimi, 2023). As a result, these models were examined with the expanded datasets to evaluate their performance and identify potential integration into the proposed model.

4.5.1 Machine Learning Models on the ICU Dataset

The comparative assessment of machine learning models against Nadam-optimized hybrid models demonstrates the hybrid models' weaknesses relative to machine learning models. For instance, there were 100% scores for the F1, accuracy, precision, recall, and specificity for machine learning, baseline, and hybrid models, except for the multilayer perceptron's 80/20 configuration. The multilayer perceptron's 80/20 configuration recorded 99.97% for F1, accuracy, precision, recall, and specificity. This feature enabled the 80/20 variant of the three-hidden-layer Nadam-optimized model to outperform it in precision and specificity, achieving 99.98%. The MLP's 80/20 variant had a false positive rate of 0.03%, which was higher than that of the 80/20 variant with three hidden layers, Nadam-optimized, at 0.02%. The gradient boosting model recorded the lowest inference time at 0.029s, while the Nadam-optimized hybrid models had the highest, reaching 2.80s. However, the 80/20 variant of a one-hidden-layer Nadam-optimized model outperformed the 70/30 and 80/20 variants of the MLP in 1.20 seconds. The MLP variants spent 1.33 and 1.36 seconds classifying threats in the dataset. This phenomenon demonstrates that the hybrid models experienced a significant setback in detection speed, despite performing well in other performance metrics. These results are shown in Figures 4.110 to 4.116.

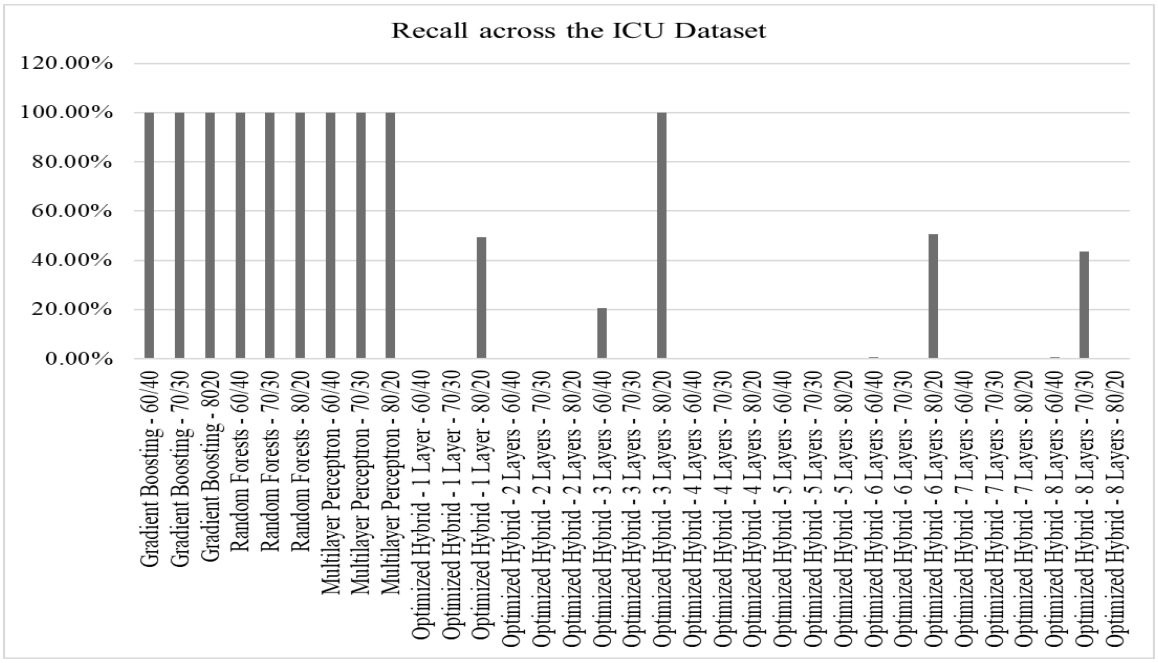


Figure 4.110: Recall scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on ICU dataset

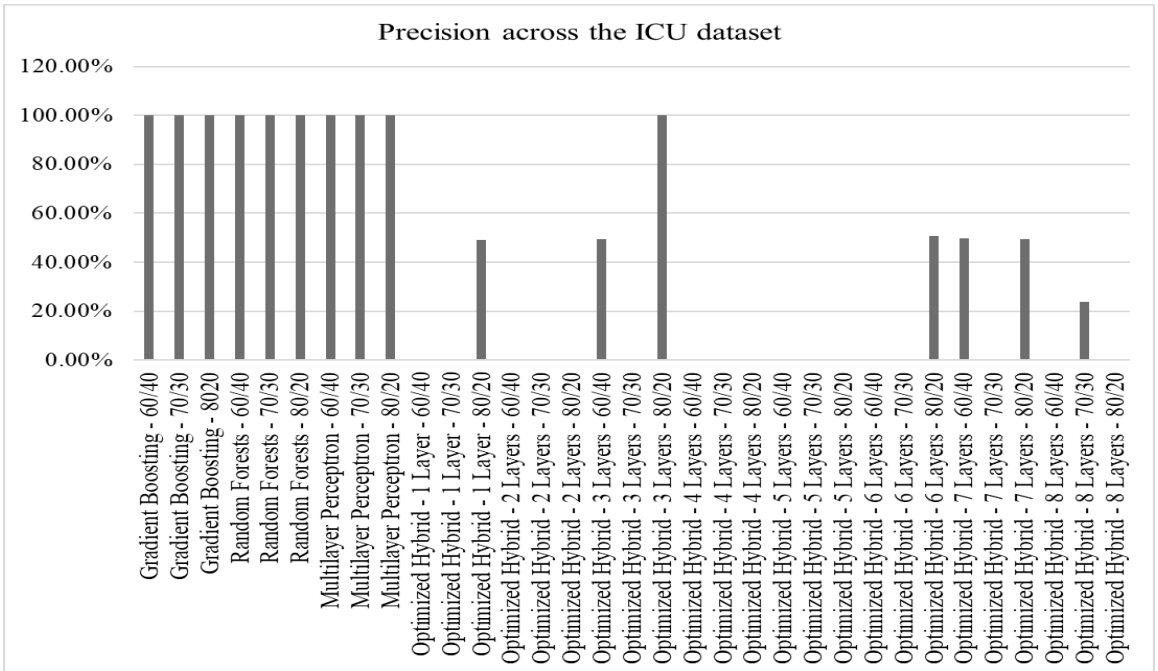


Figure 4.111: Precision scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on ICU dataset

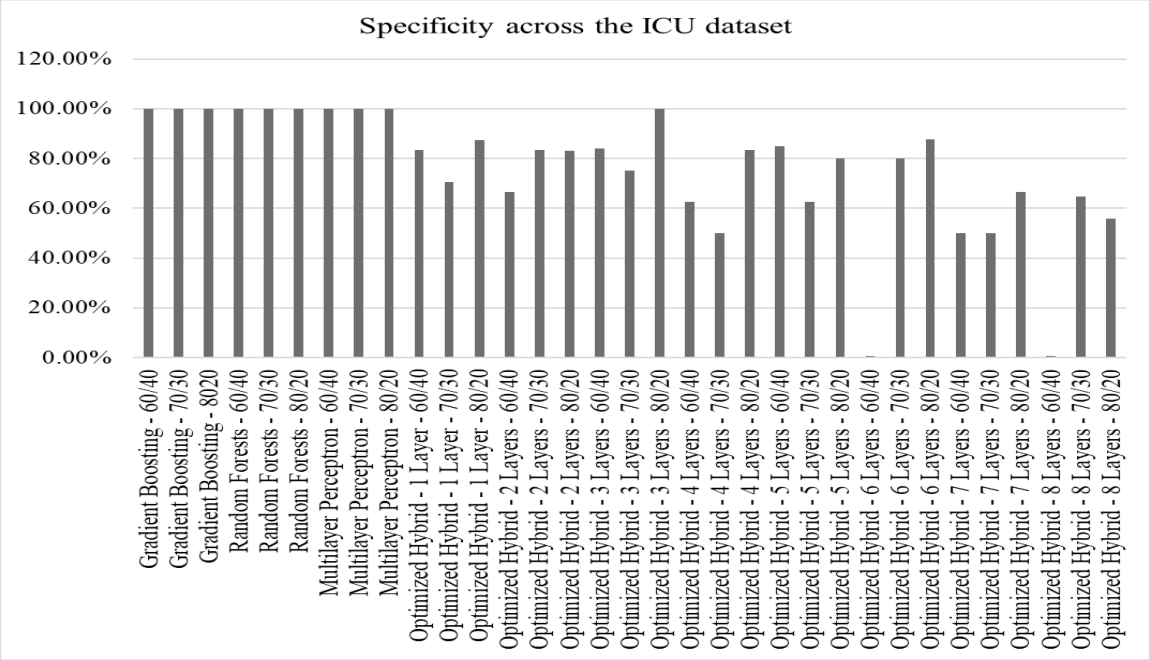


Figure 4.112: Specificity scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on ICU dataset

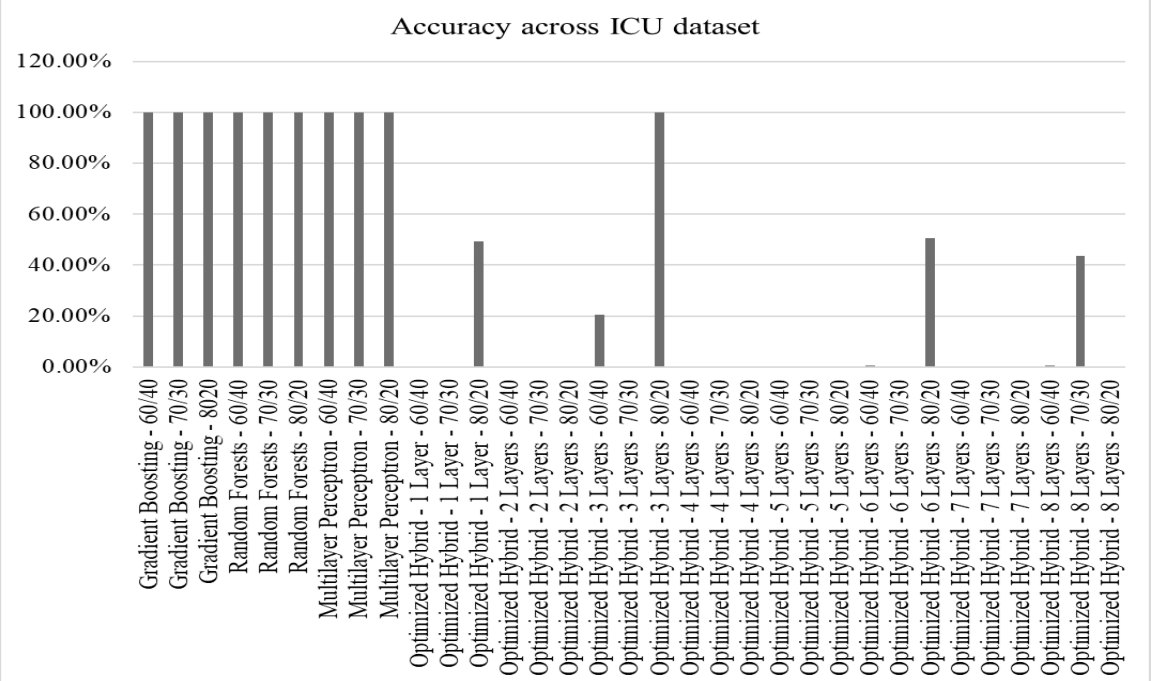


Figure 4.113: Accuracy scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on ICU dataset

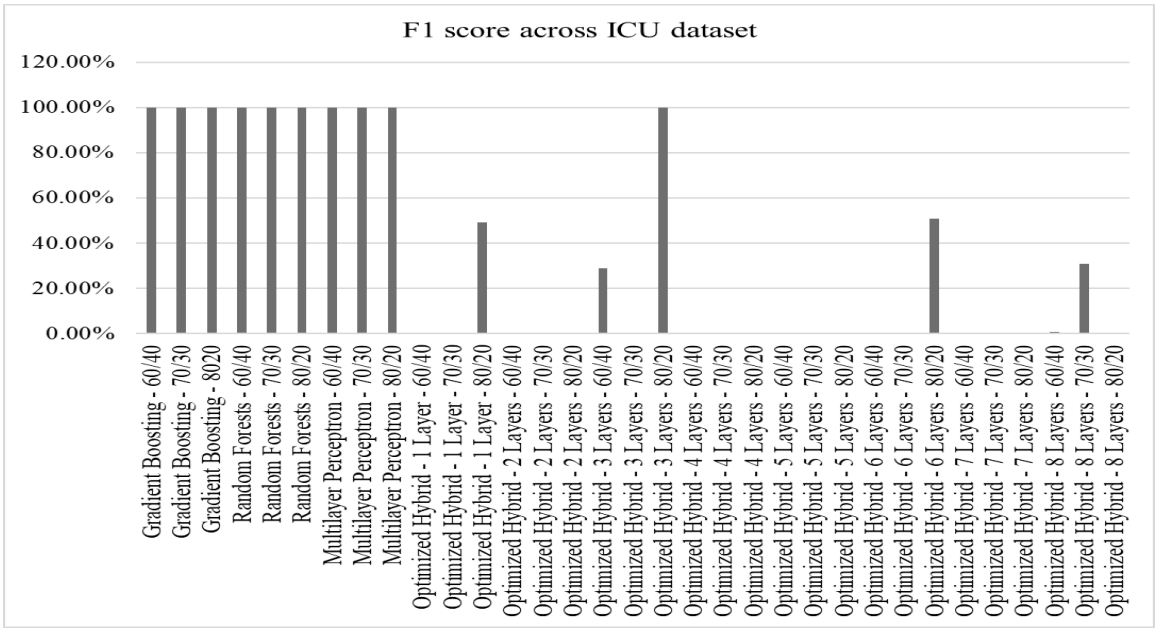


Figure 4.114: F1 scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on ICU dataset

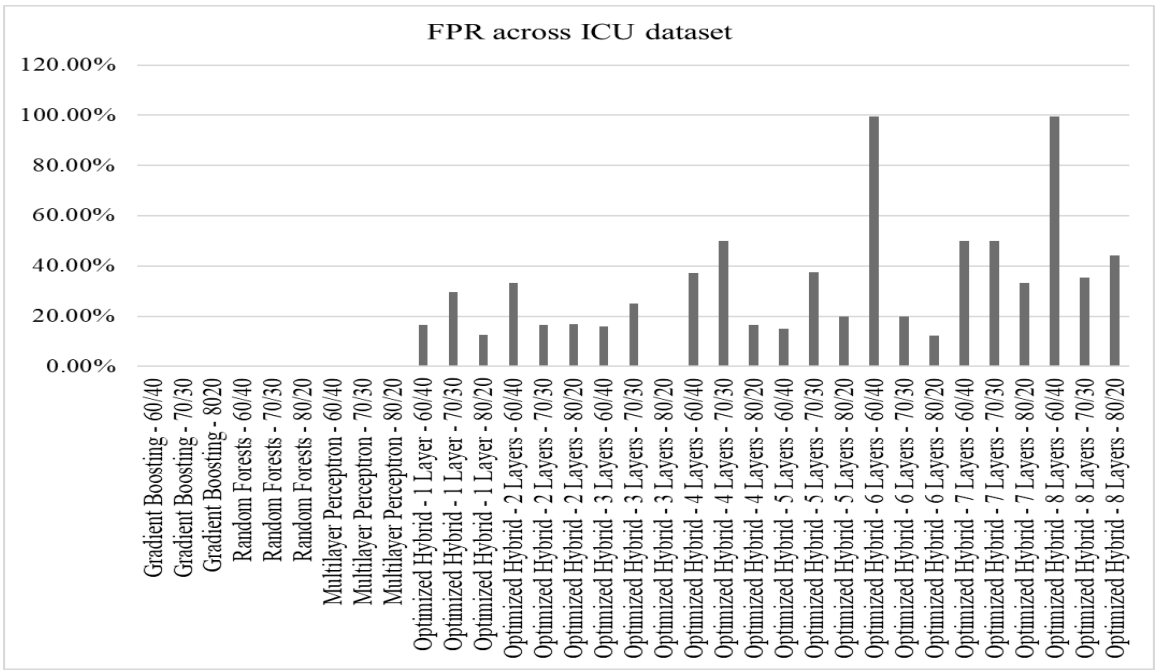


Figure 4.115: False positive rate for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on ICU dataset

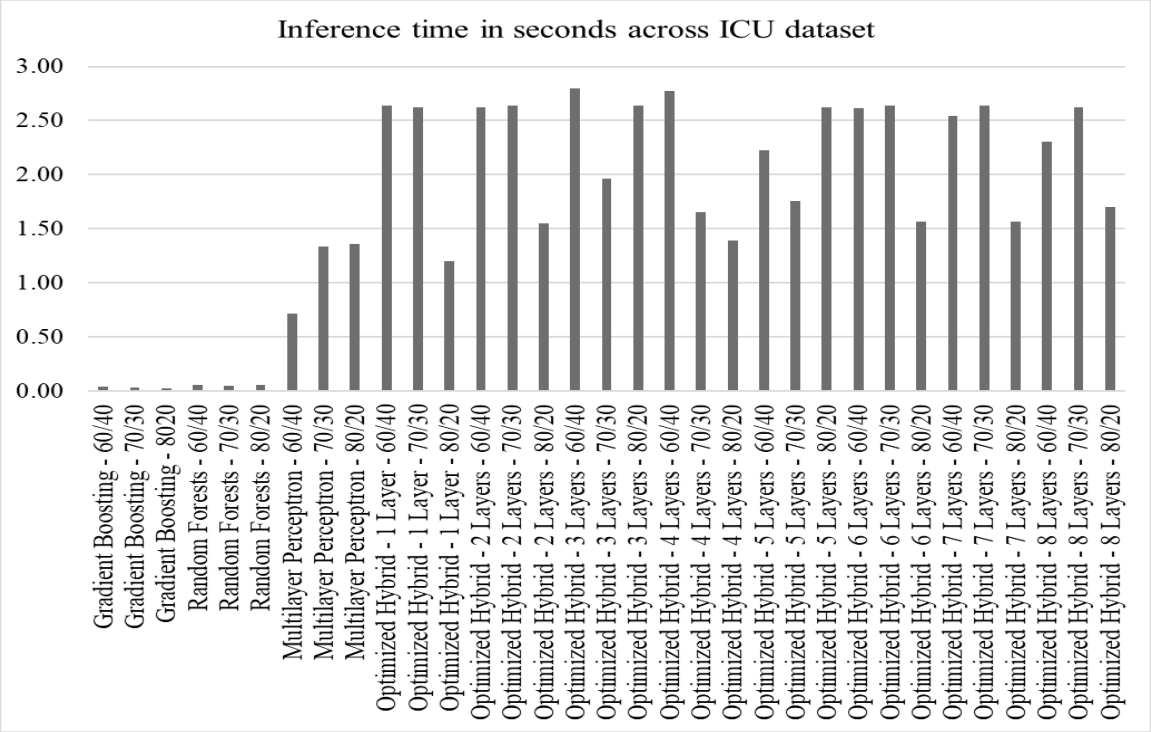


Figure 4.116: Inference time for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on ICU dataset

4.5.2 Machine Learning Models on WUSTL Dataset

The comparative analysis of machine learning models and Nadam-optimized hybrid models provides insights into their performance relative to machine learning models. The machine learning models dominated recall, accuracy, F1, and inference scores, while the Nadam-optimized hybrid models emerged as the best in precision, specificity, and false positive rates. For instance, the 60/40 variant of one hidden layer and the 70/30 variants of three hidden layers outperformed the machine learning models in terms of precision, scoring 88.68% and 87.21%, respectively. The five-hidden-layers' 80/20, three-hidden-layers' 70/30, and other hybrid models dominated the key positions, scoring over 87% in specificity. In comparison, the best machine learning model was the Random Forests' 80/20 variant at 86.60%. This phenomenon demonstrates that Nadam optimization outperformed machine learning strategies in terms of specificity. Similar behavior was observed in false-positive rates, where hybrid models scored as low as 7.85%. The best machine learning model achieved 3.40%, demonstrating weak performance compared to the hybrid models. These results are shown in Figures 4.117, 4.118, 4.119, 4.120, 4.121, 4.122 and 4.123.

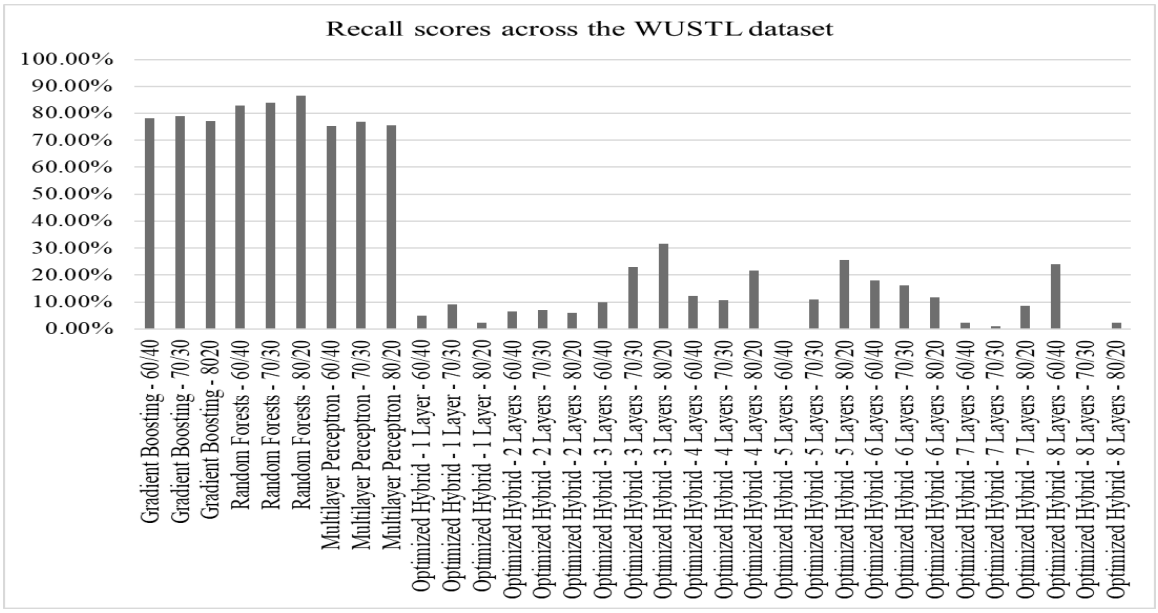


Figure 4.117: Recall scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on WUSTL dataset

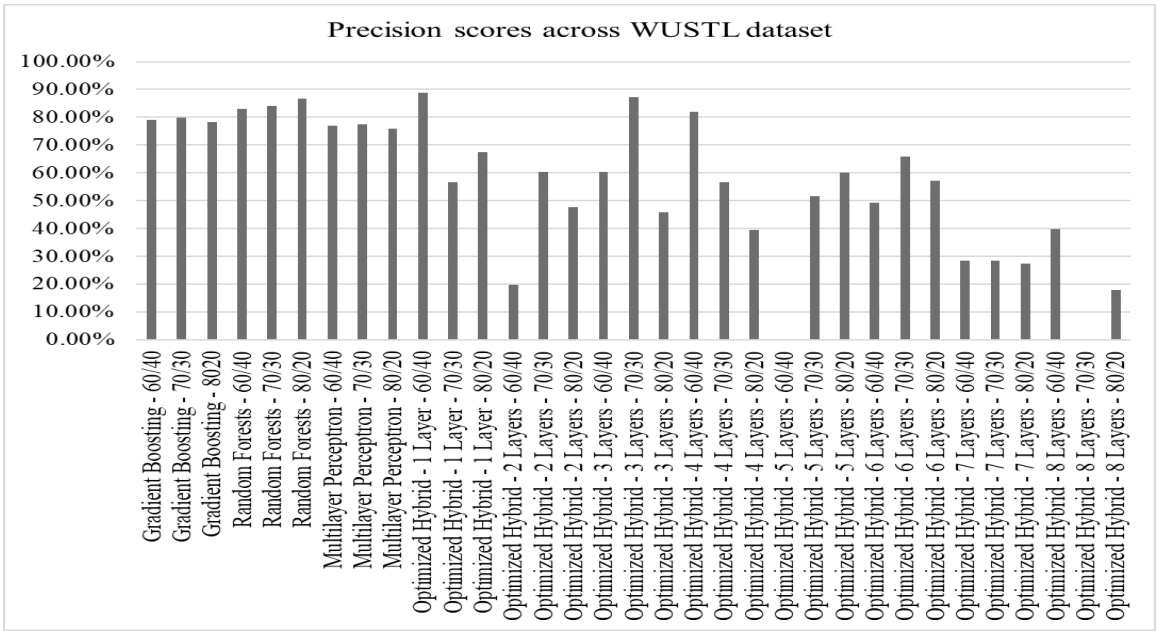


Figure 4.118: Precision scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on WUSTL dataset

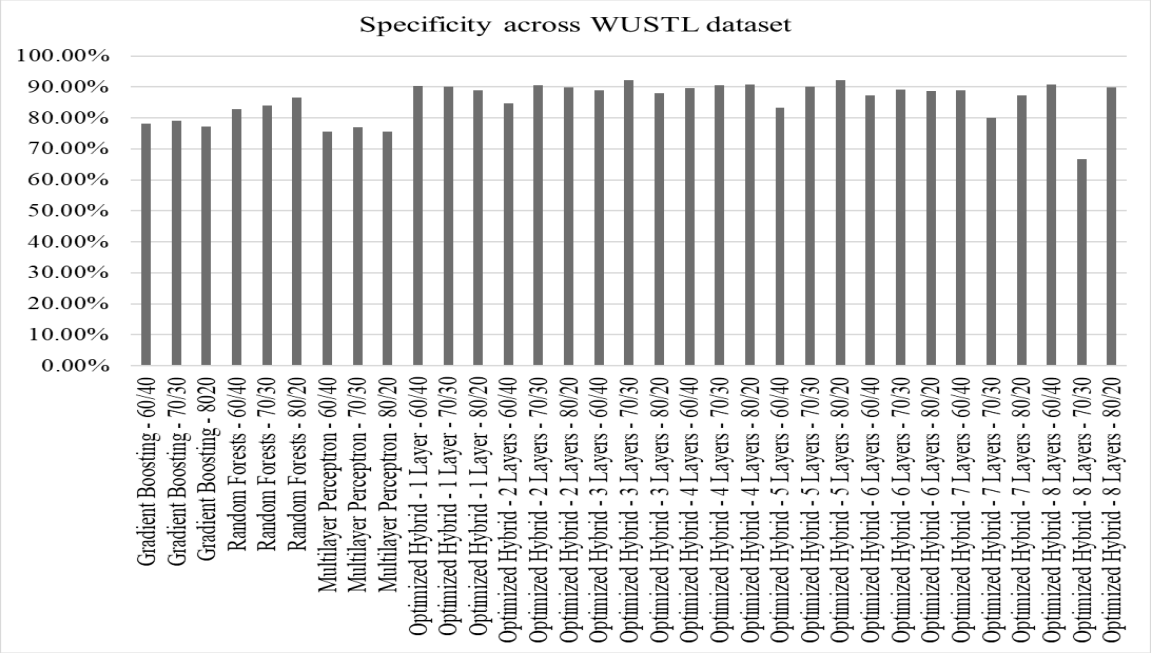


Figure 4.119: Specificity scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on WUSTL dataset

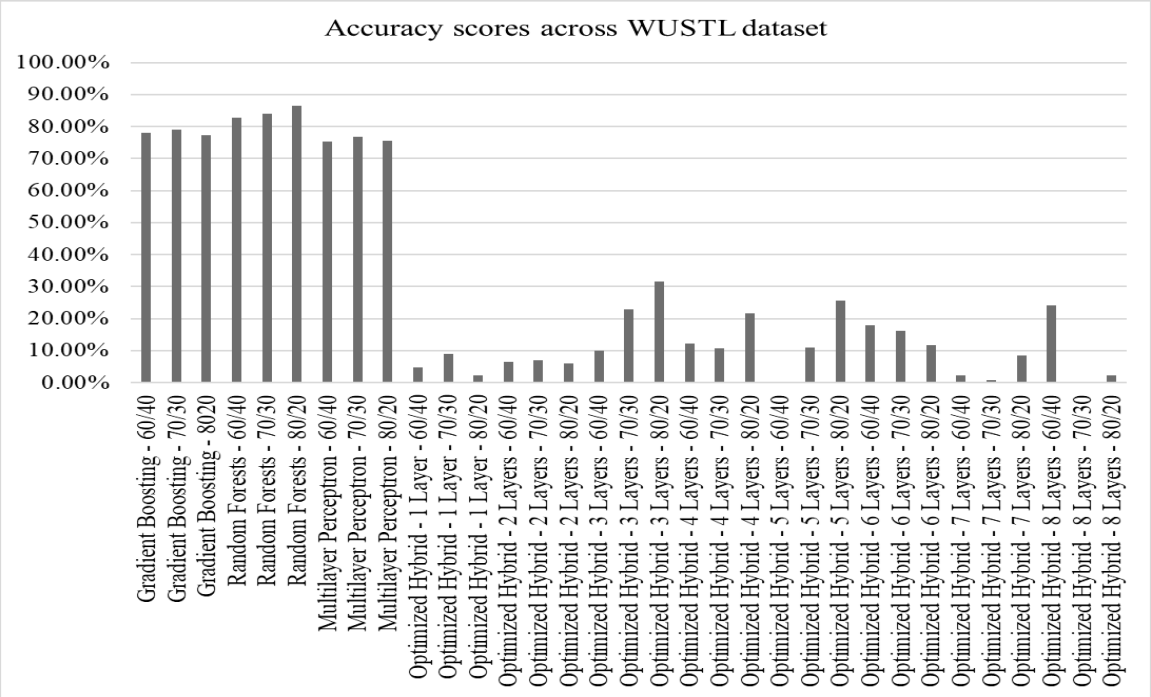


Figure 4.120: Accuracy scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on WUSTL dataset

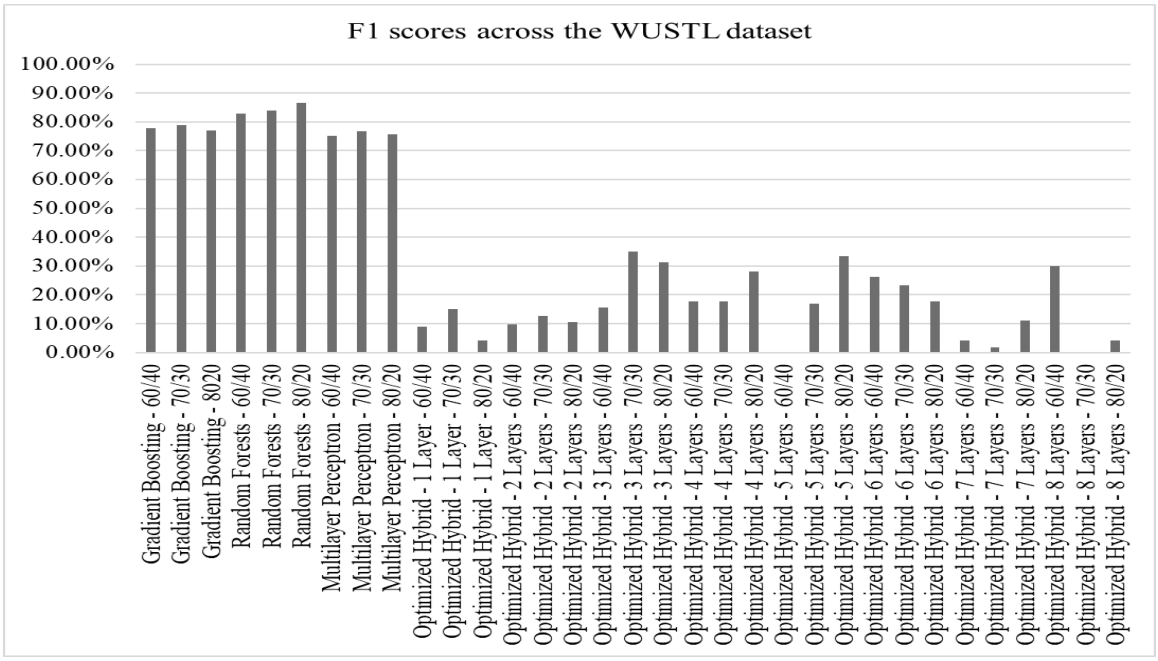


Figure 4.121: F1 scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on WUSTL dataset

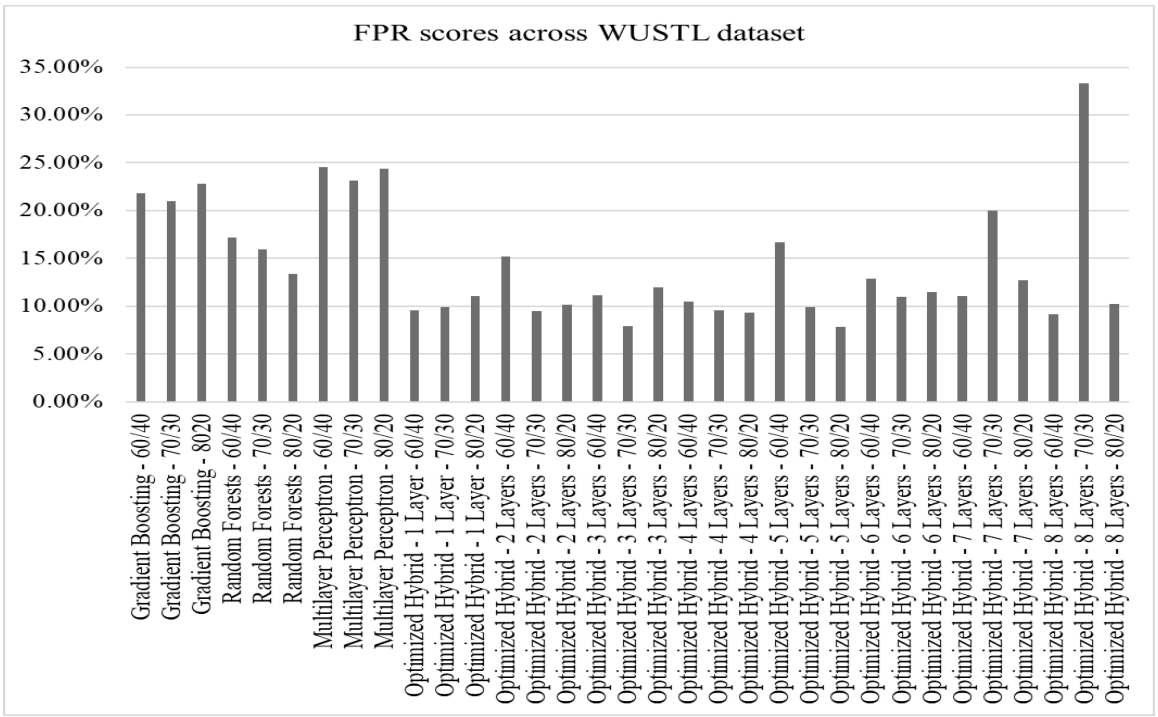


Figure 4.122: False positive rate for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on WUSTL dataset

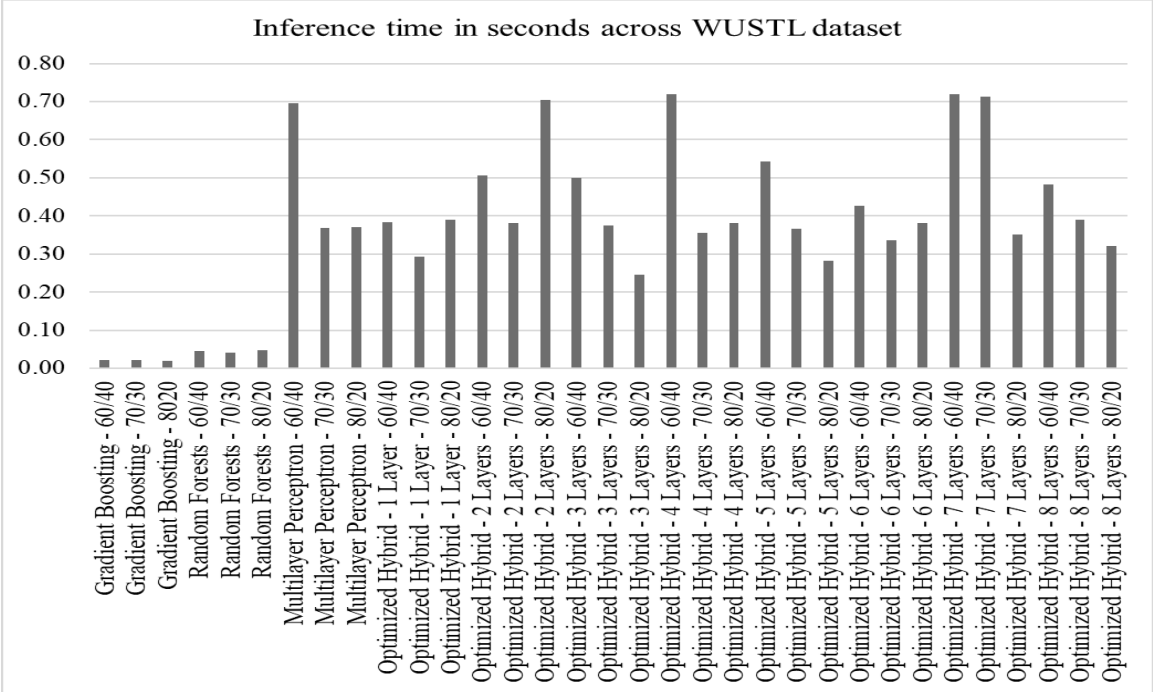


Figure 4.123: Inference time for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on WUSTL dataset

4.5.3 Machine Learning Models on Edge IIoT Dataset

The comparative assessment of machine learning models against Nadam-optimized hybrid models provides insights into their performance relative to these machine learning approaches. Machine learning models dominate recall, accuracy, precision, specificity, F1, false-positive rates, and inference time among the compared models. For instance, the random forests' 80/20 variant emerged as the best model for the dataset. This variant had recall, accuracy, and F1 scores of 96.92%, precision of 96.98%, specificity of 96.93%, and a false positive rate of 3.07%. The gradient boosting model performed worst, scoring 92.66% for recall and accuracy, 92.65% for F1, 92.69% for specificity, and a false-positive rate of 7.31%. The lowest inference time was obtained from gradient boosting's 80/20 variant at 0.05s, while the highest was obtained from three hidden layers' 6040 variant at 10.34 seconds. These results are shown in Figures 4.124, 4.125, 4.126, 4.127, 4.128, 4.129 and 4.130.

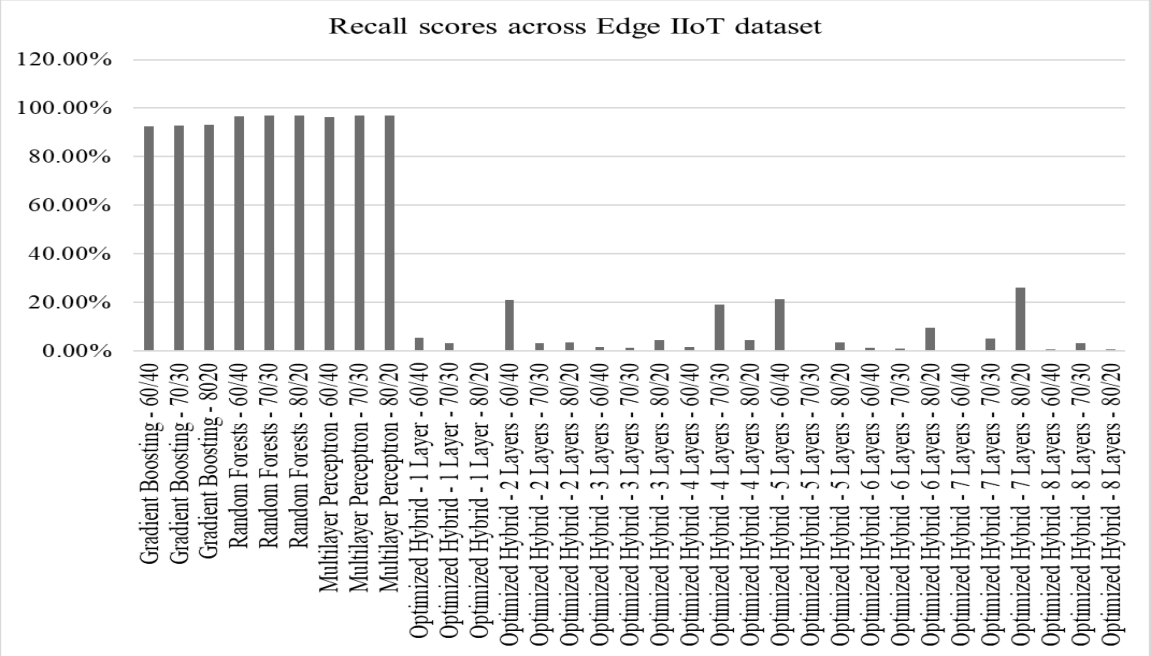


Figure 4.124: Recall scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on Edge IIoT dataset

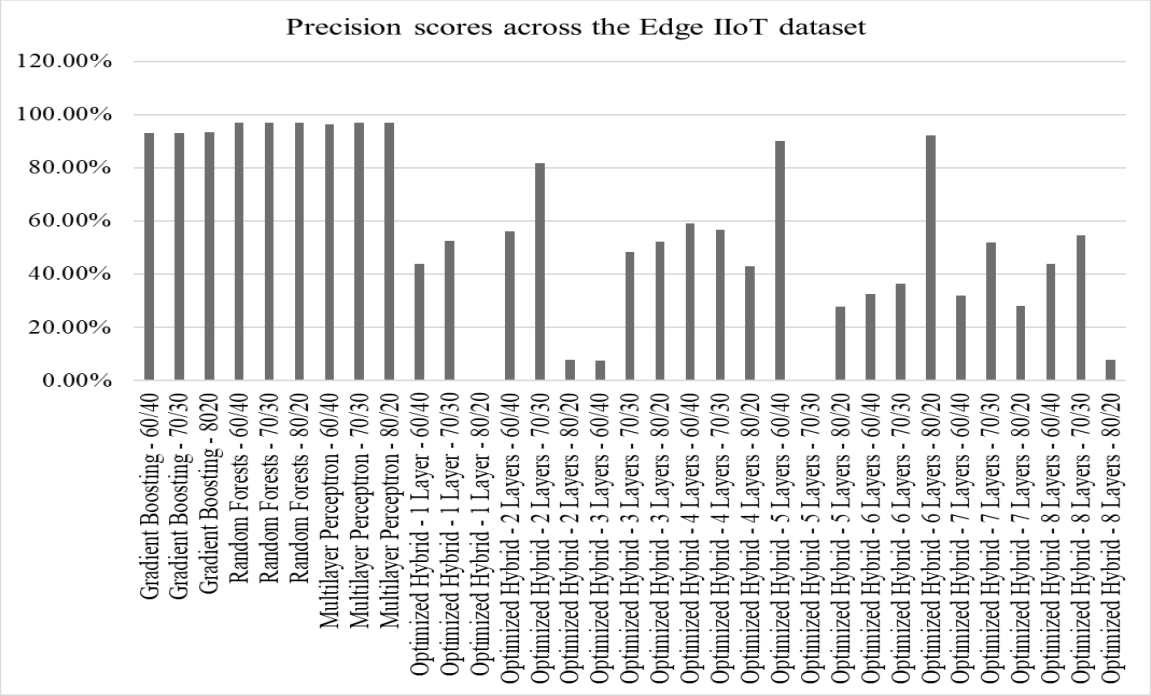


Figure 4.125: Precision scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on Edge IIoT dataset

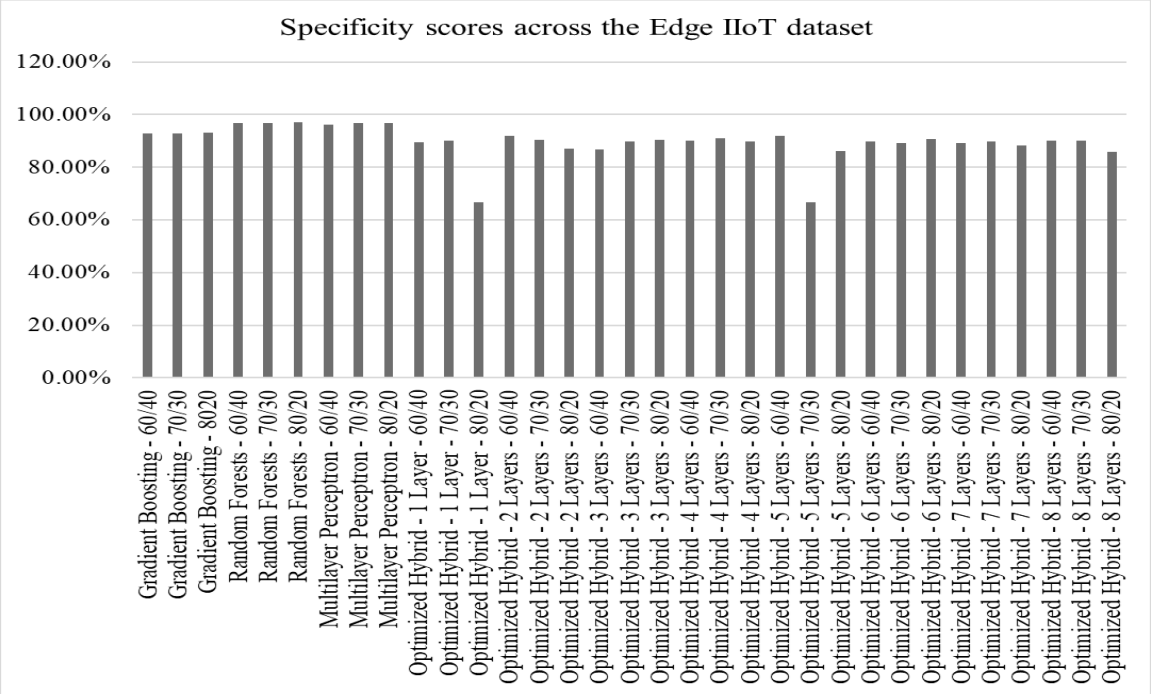


Figure 4.126: Specificity scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on Edge IIoT dataset

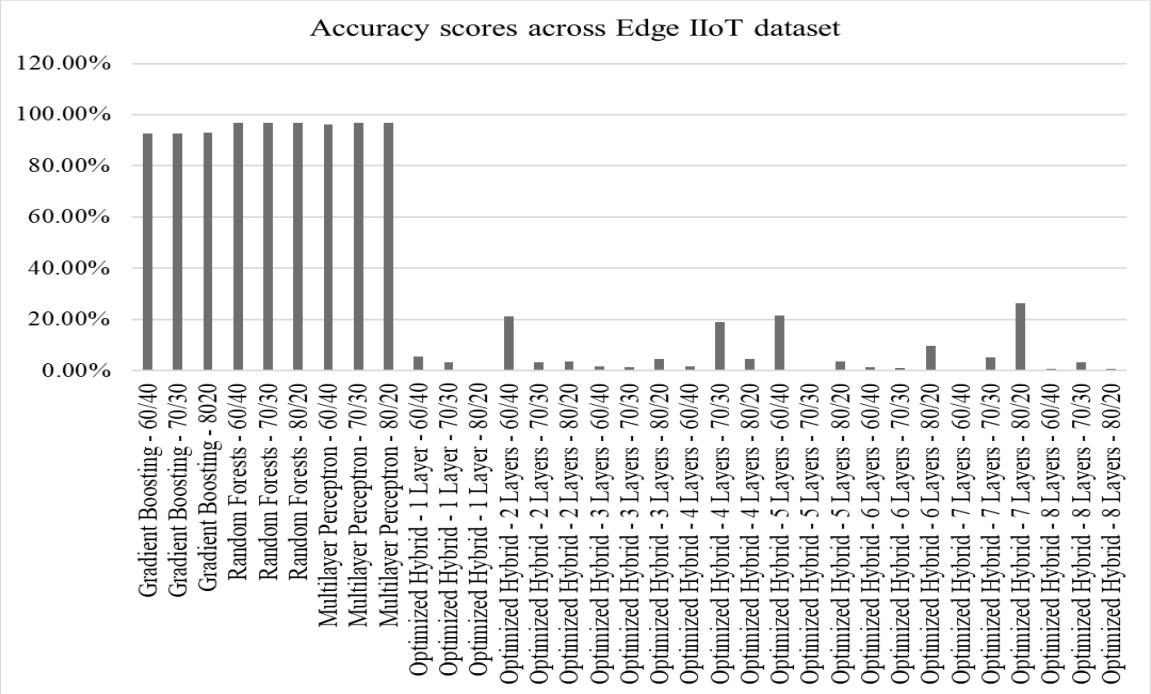


Figure 4.127: Accuracy scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on Edge IIoT dataset

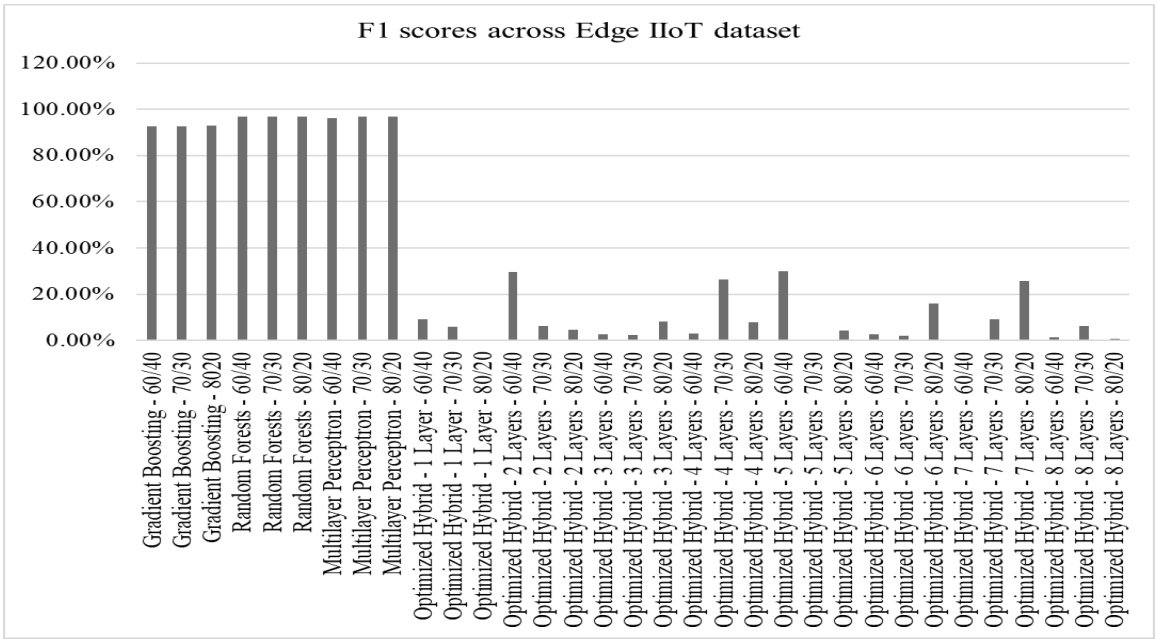


Figure 4.128: F1 scores for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on Edge IIoT dataset

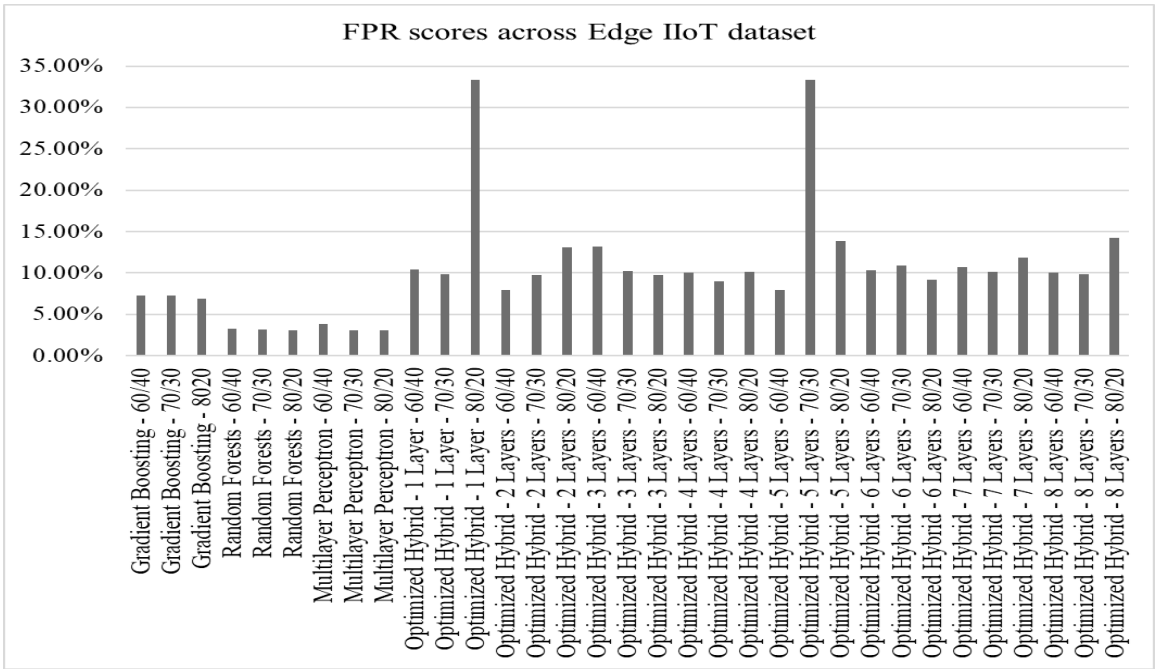


Figure 4.129: False positive rate for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on Edge IIoT dataset

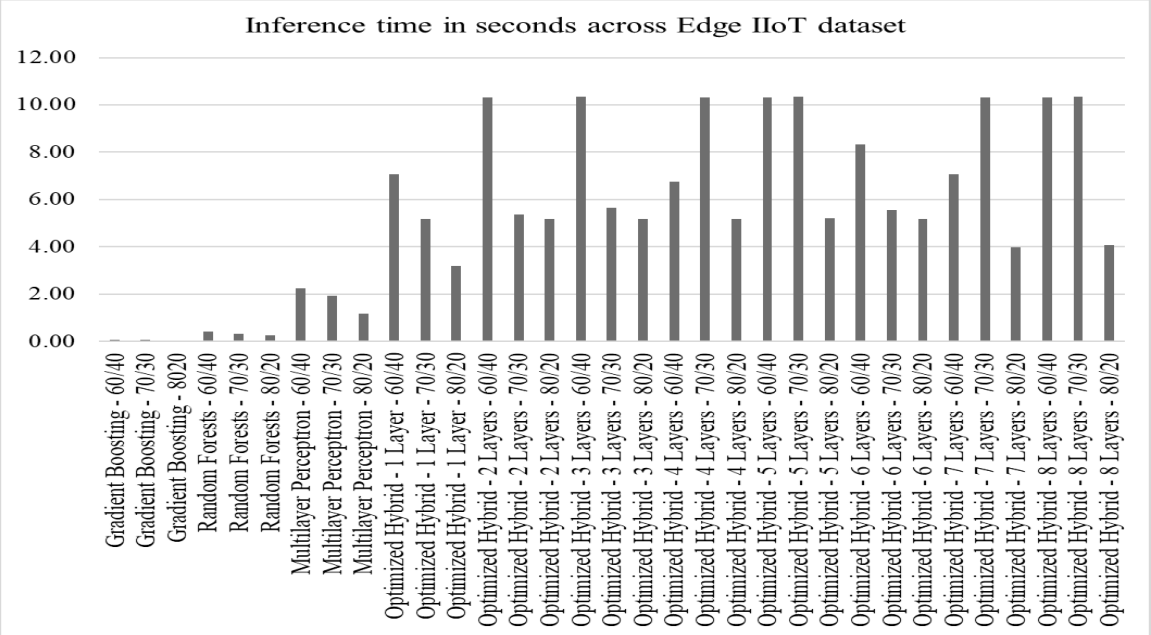


Figure 4.130: Inference time for Gradient Boosting, Random Forests, Multilayer Perceptron, and Nadam-optimized hybrid deep autoencoder model on Edge IIoT dataset

4.6 Enhanced Hybrid Model

The hybrid model was further refined by incorporating batch normalization, regularization, and a Multilayer Perceptron layer serving as the classification head. The resulting model was trained and tested across the three datasets, with each dataset split into three training/testing ratios. There were seven training/testing sessions that involved increasing the number of hidden layers in the encoder/decoder from 1 to 7. Unlike the Nadam-optimized and standard models, the eight-hidden-layer model could not be trained due to repeated crashes in the cloud environment. This phenomenon was attributed to the high number of hidden layers, which increased the model's complexity and led to increased resource consumption on the platform. Nevertheless, a reasonable number of hidden layers were used to provide an informed analysis of the model.

4.6.1 Enhanced Hybrid Model on the ICU Dataset

The analysis of the Multilayer Perceptron-enhanced Nadam-optimized hybrid model against the baseline and machine learning models provides insights into the implications of MLP enhancement. The MLP’s integration into the hybrid model offered excellent results, enabling the model to achieve high scores. For instance, the 80/20 variant of one hidden layer, the 60/40 variants of four hidden layers, the 70/30

variants of five hidden layers, and the 70/30 variant of six hidden layers outperformed the baseline models and rivaled the machine learning models. These models scored 100% in recall, precision, specificity, accuracy, and F1 scores, and had 0% false-positive rates. This phenomenon enabled them to outperform the MLP's 80/20 variant, which scored 99.97% in recall, precision, specificity, accuracy, and F1, and 0.03% in false positive rate. However, machine learning models dominated the inference scores, recording as low as 0.03 seconds for detection. The best hybrid model in terms of detection speed was a 60/40 variant with one hidden layer, which took 1.20 seconds. This situation marked a slight improvement from Nadam optimization, where the best hybrid model had recorded 1.39 seconds for the ICU dataset. The slowest detection speed for the MLP-enhanced hybrid model was obtained from the four-hidden-layers' 80/20 variant at 5.20 seconds. These results are shown in Figures 4.131, 4.132, 4.133, 4.134, 4.135, 4.136 and 4.137.

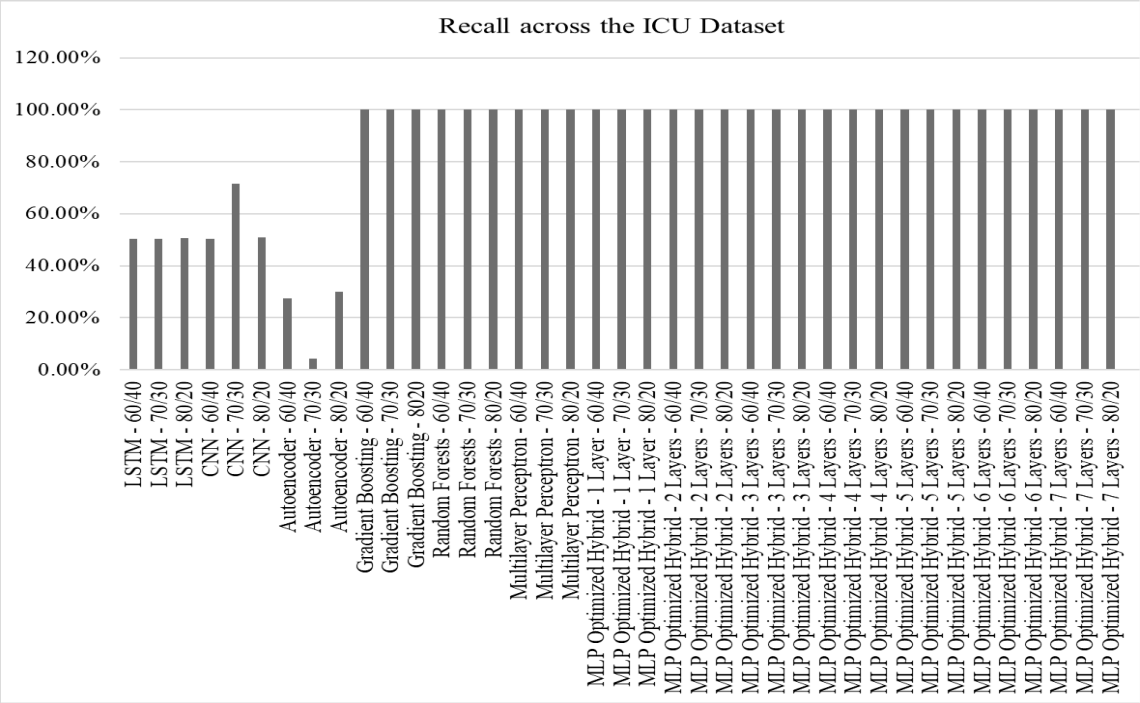


Figure 4.131: Recall scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on ICU dataset

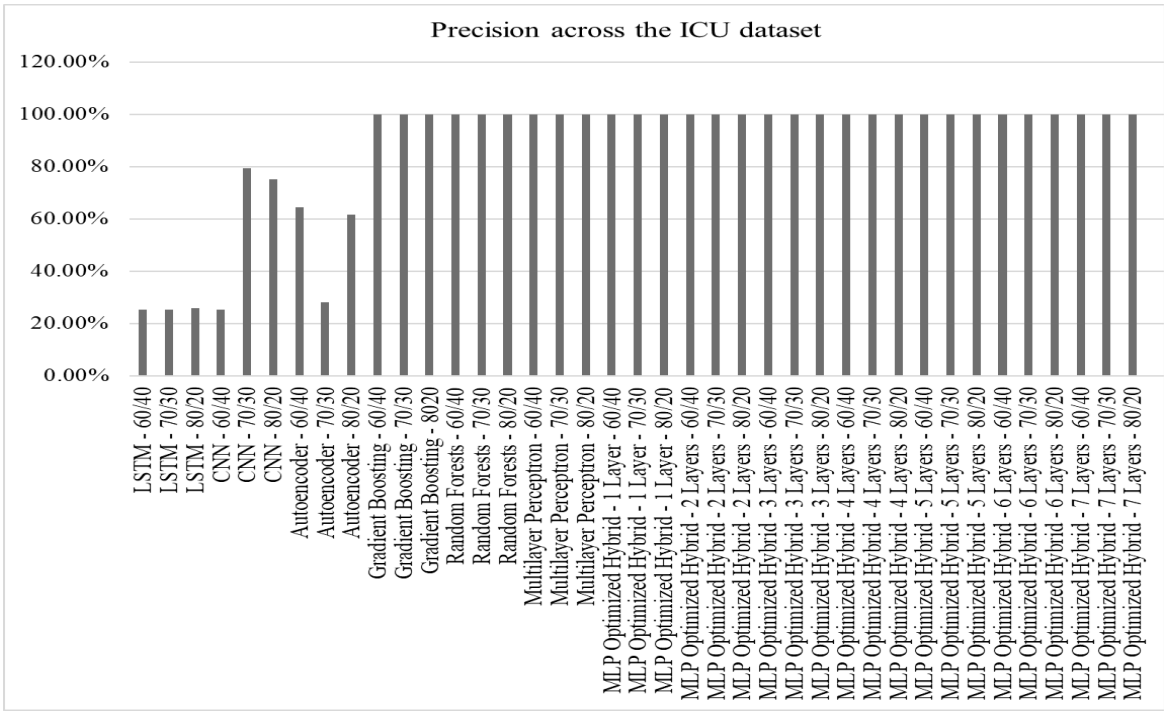


Figure 4.132: Precision scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on ICU dataset

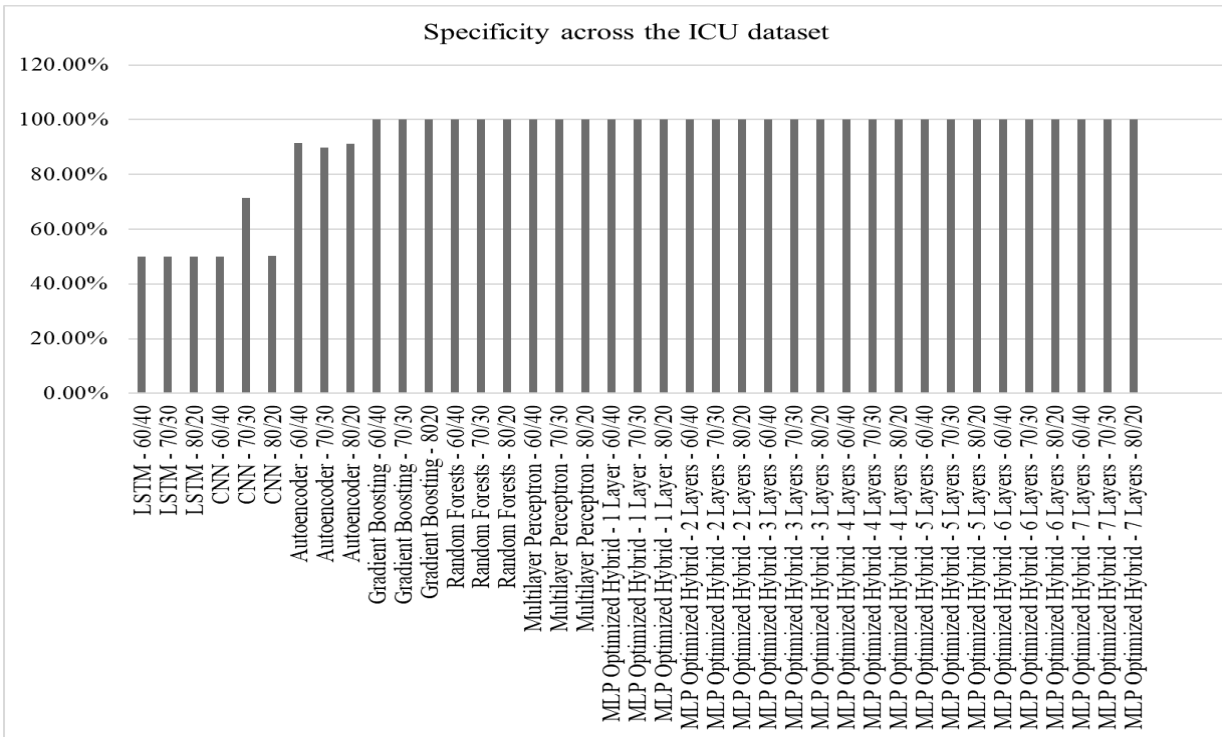


Figure 4.133: Specificity scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on ICU dataset

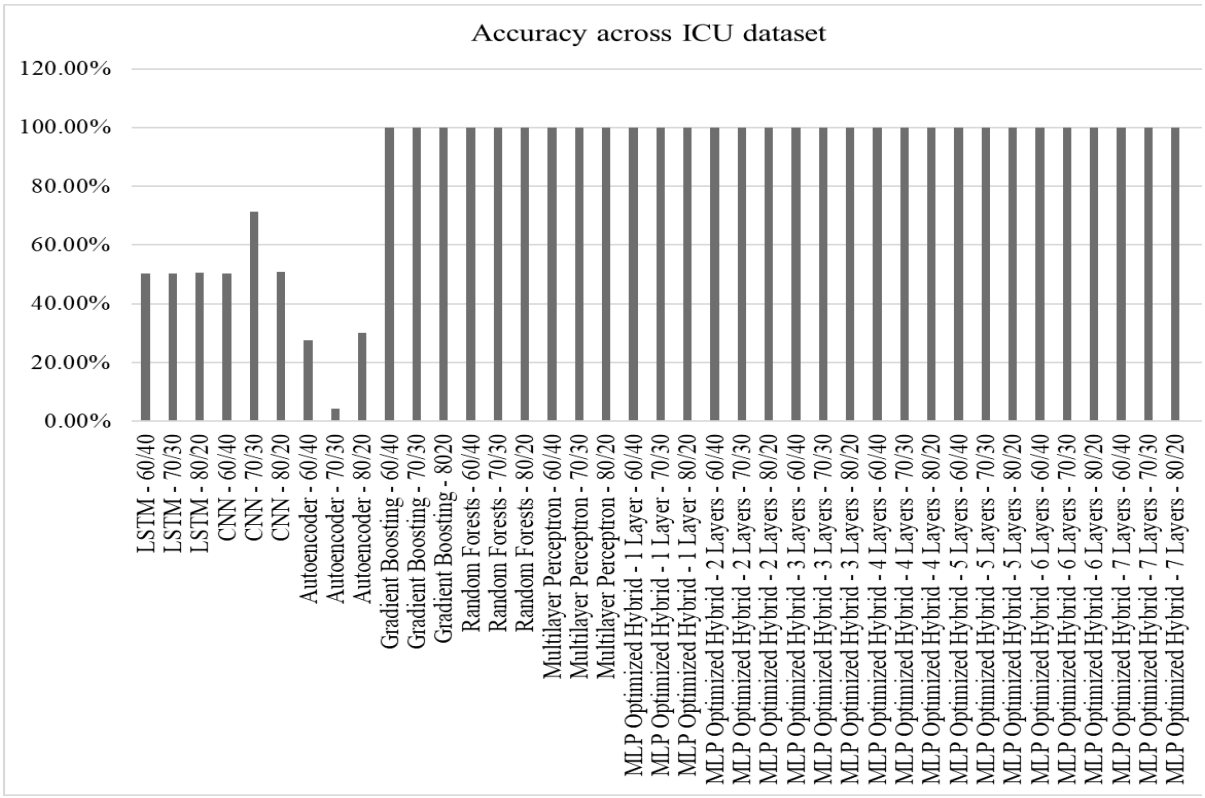


Figure 4.134: Accuracy scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on ICU dataset

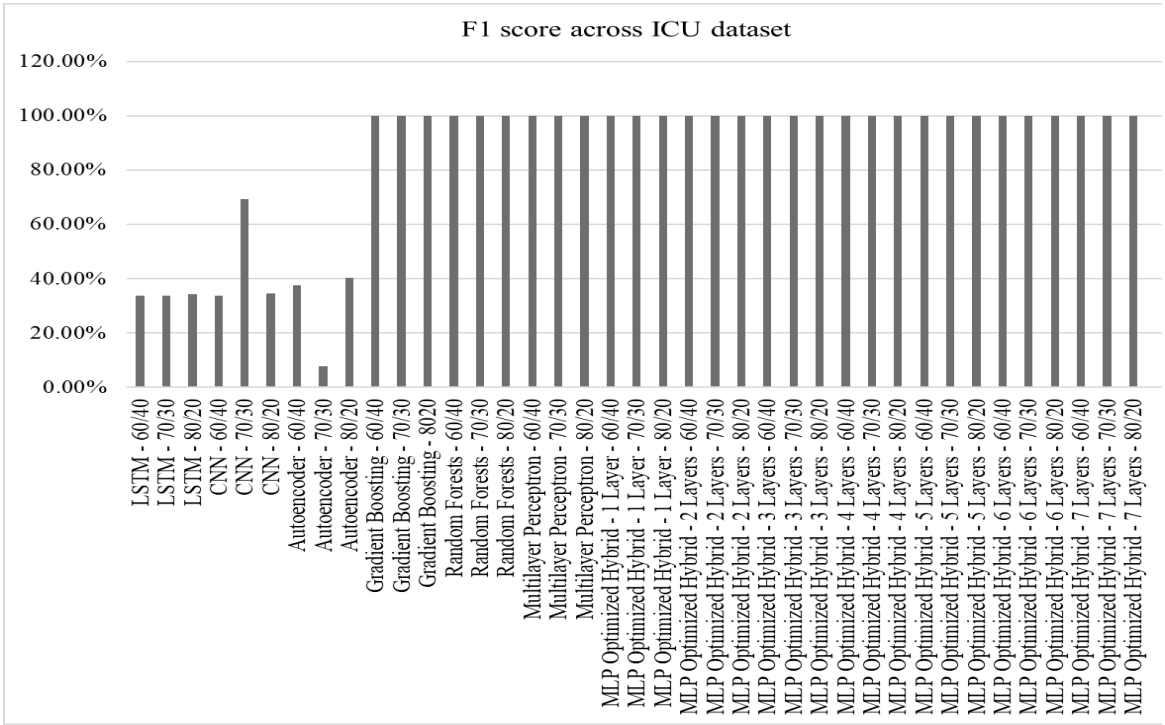


Figure 4.135: F1 scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on ICU dataset

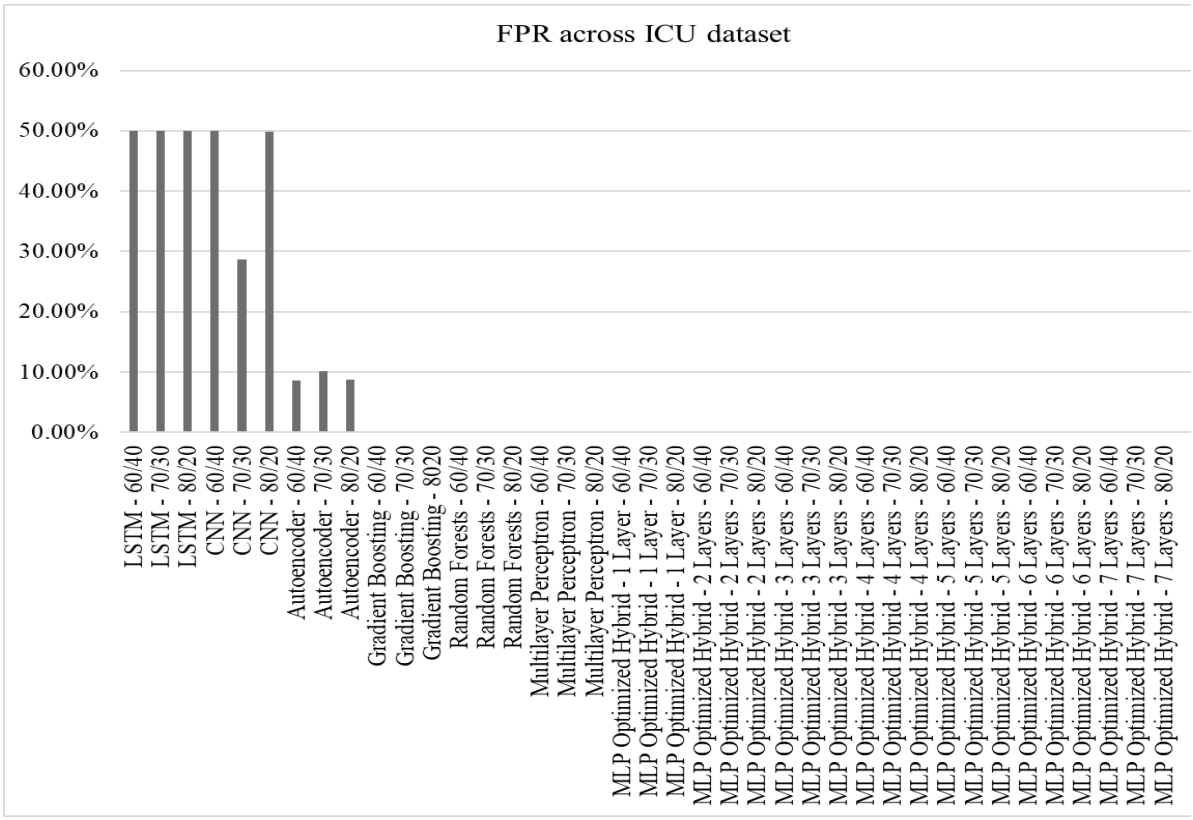


Figure 4.136: False positive rates for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on ICU dataset

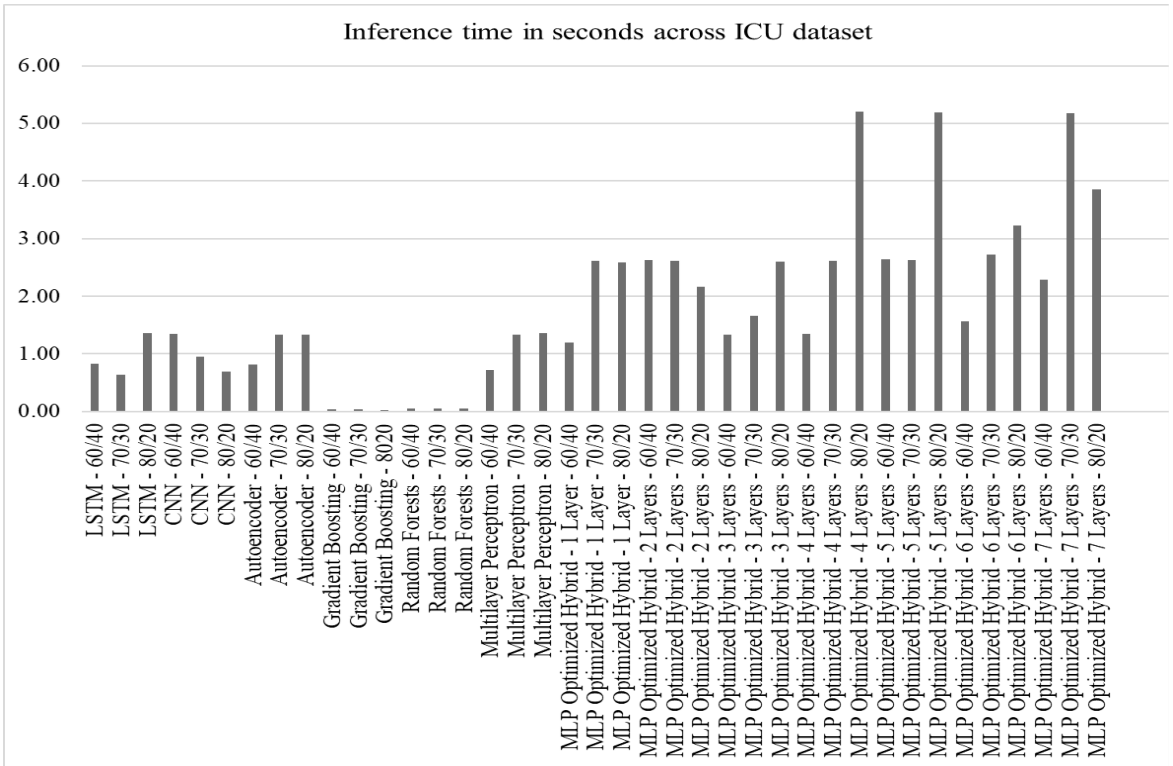


Figure 4.137: Inference time for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on ICU dataset

A comparative analysis of the four dominant models in the ICU dataset revealed their performance across different numbers of hidden layers and training/testing ratios. For instance, the six-hidden-layer 70/30 variant had the highest inference time, followed by the five-hidden-layer 70/30 configuration. One hidden layer's 80/20 variation was at the third position, while the four-hidden-layers' 60/40 variant emerged as the least inference time. The accuracy, recall, F1, precision, and FPR scores for these models were similar. Additionally, regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) provided insights into the model's performance across variations. These regression metrics had a score of zero across the variations, demonstrating a perfect fit for the prediction. This score coincided with the model's accuracy, recall, precision, specificity, FPR, and F1 scores. These comparative results are shown in Figure 4.138 and 4.139.

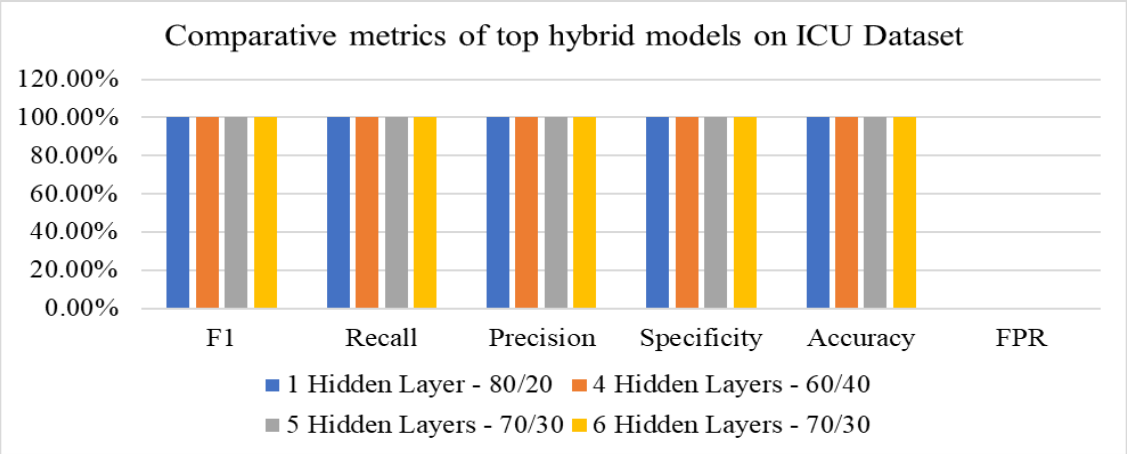


Figure 4.138: Comparative view of performance scores for selected hybrid models on the ICU dataset

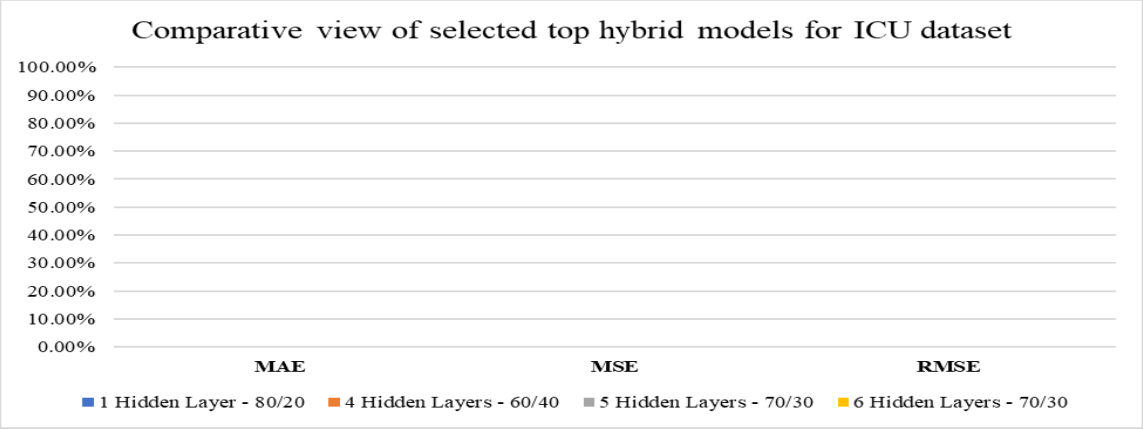


Figure 4.139: Comparative view of regression metrics for selected hybrid models on the ICU dataset

Analysis of the confusion matrix and receiver operating characteristics (ROC) curves enhanced understanding of these selected models. For instance, the ROC curve for one hidden layer's 80/20 split showed a perfect fit, with an Area under the ROC Curve (AUC) of 1.00. The confusion matrix for the model showed zero scores for the false positives and false negatives. These scores from the confusion matrix and ROC curve showed ideal performance of the developed model. This behavior was replicated in four-hidden-layers' 60/40 variation, five-hidden-layers' 70/30 configuration, and six-hidden-layer 70/30 variant. The differences in the confusion matrices arose only from the numbers of true negatives and true positives. This difference was attributed to the test subset of the dataset, determined by the splitting ratio. For instance, the 80/20 ratio uses 6,000 samples for testing, while the 70/30 ratio uses 9,001 samples. The 60/40 ratio uses 12,000 samples for model testing. These confusion matrices and ROC curves are shown in Figures 4.140, 4.141, 4.142, 4.143, 4.144, 4.145, 4.146 and 4.147.

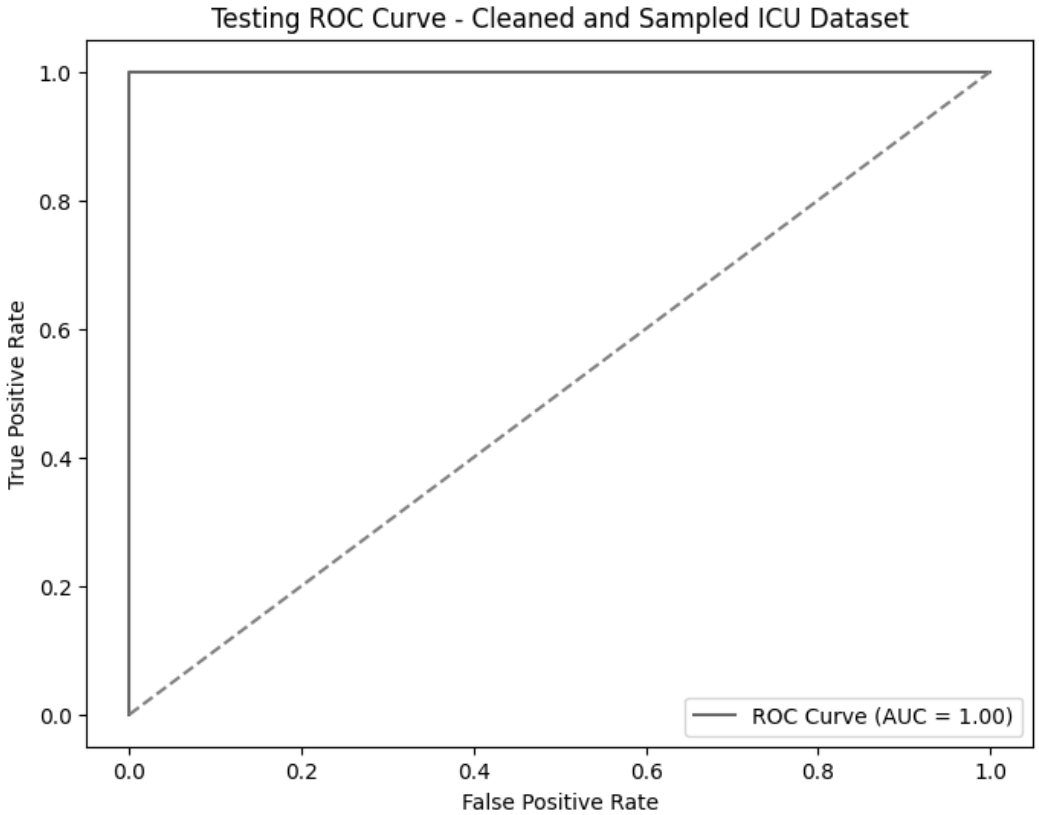


Figure 4.140: Receiver Operating Characteristics curve for one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on the ICU Dataset

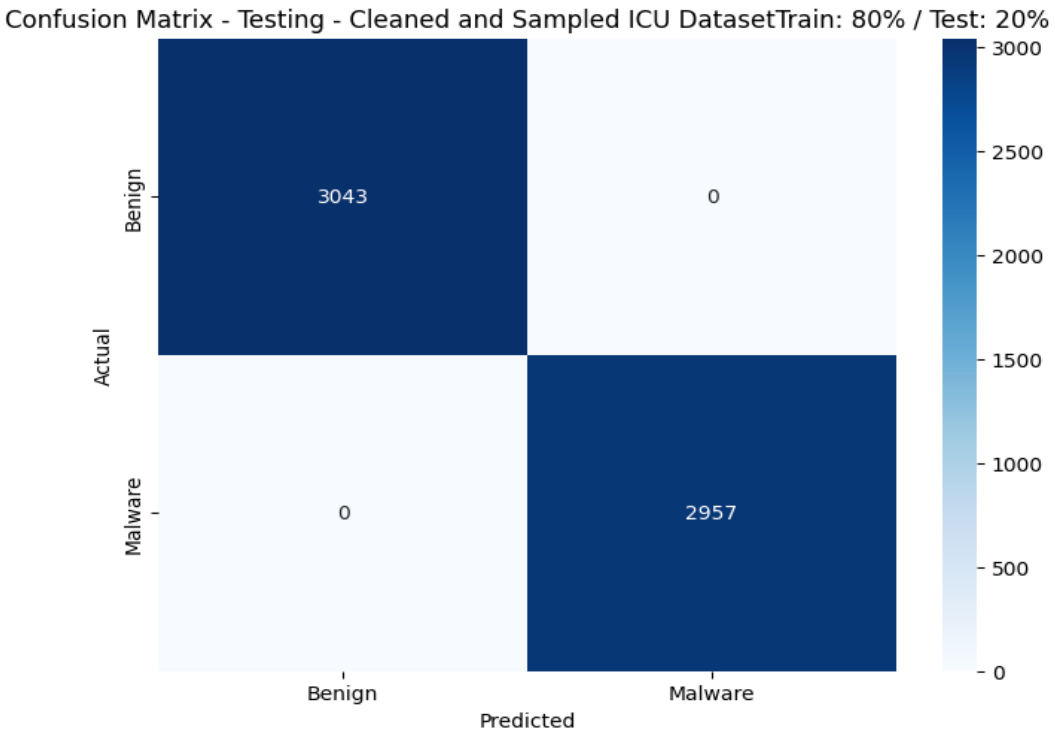


Figure 4.141: Confusion matrix for one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on the ICU Dataset

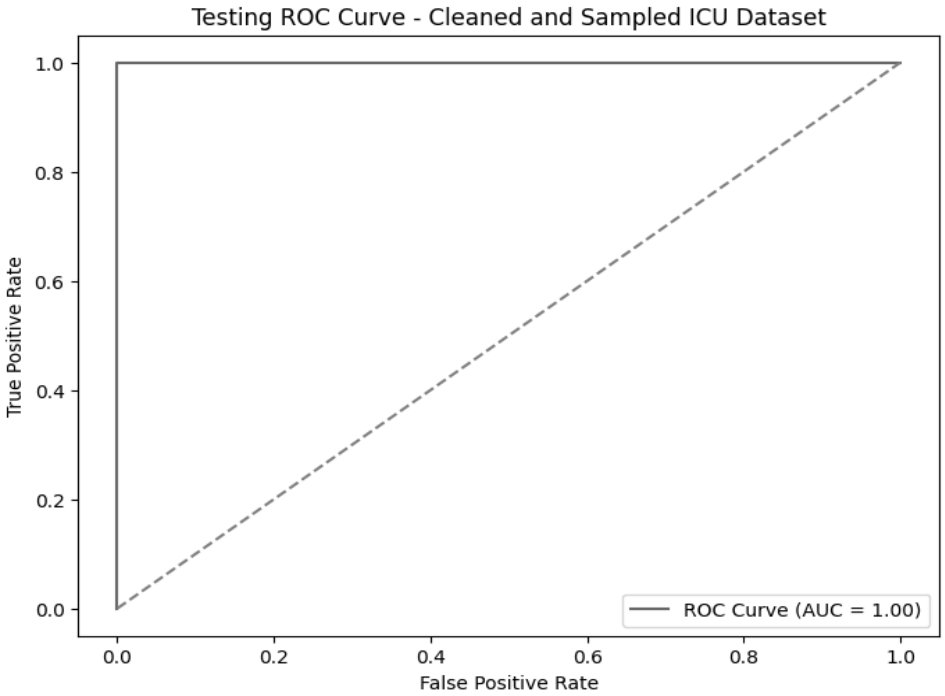


Figure 4.142: Receiver Operating Characteristics curve for a four-hidden-layer' 60/40 MLP-Enhanced and Nadam-optimized model on the ICU dataset

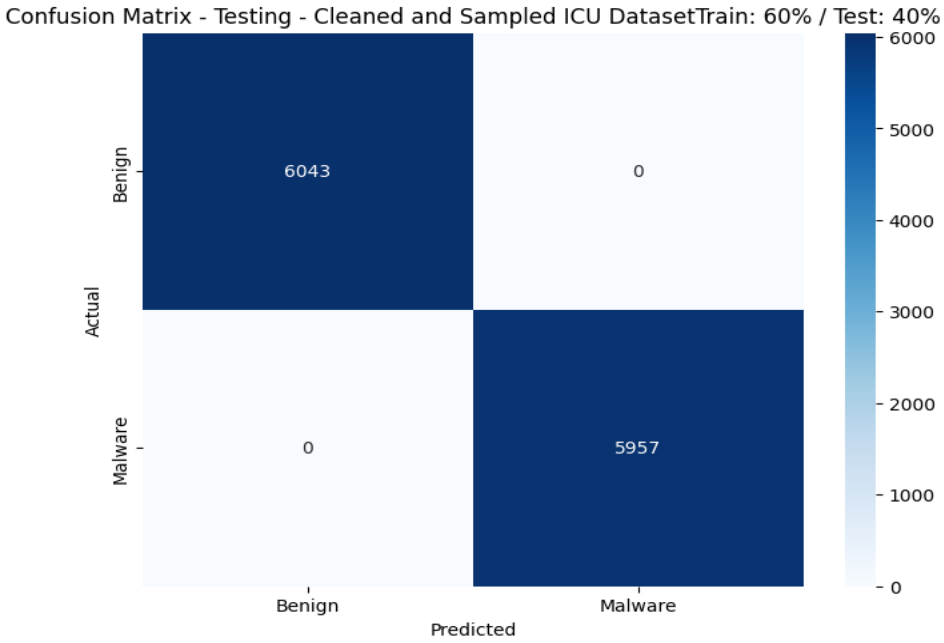


Figure 4.143: Confusion matrix for four-hidden-layers' 60/40 MLP-Enhanced and Nadam-optimized model on ICU dataset

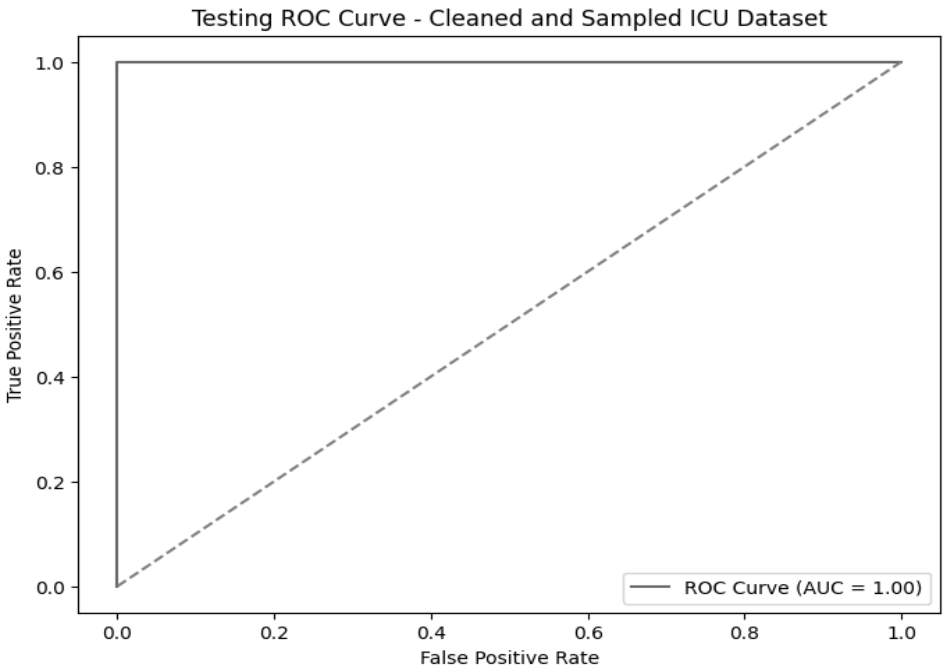


Figure 4.144: Receiver Operating Characteristics curve for a five-hidden-layer's 70/30 MLP-Enhanced and Nadam-optimized model on the ICU dataset

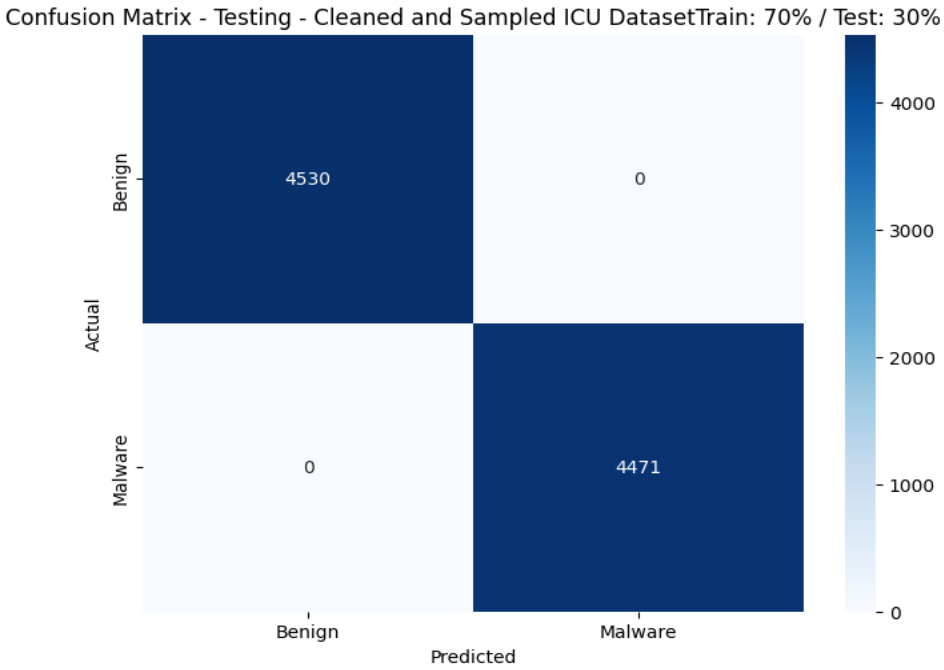


Figure 4.145: Confusion matrix for five-hidden-layers' 70/30 MLP-Enhanced and Nadam-optimized model on ICU dataset

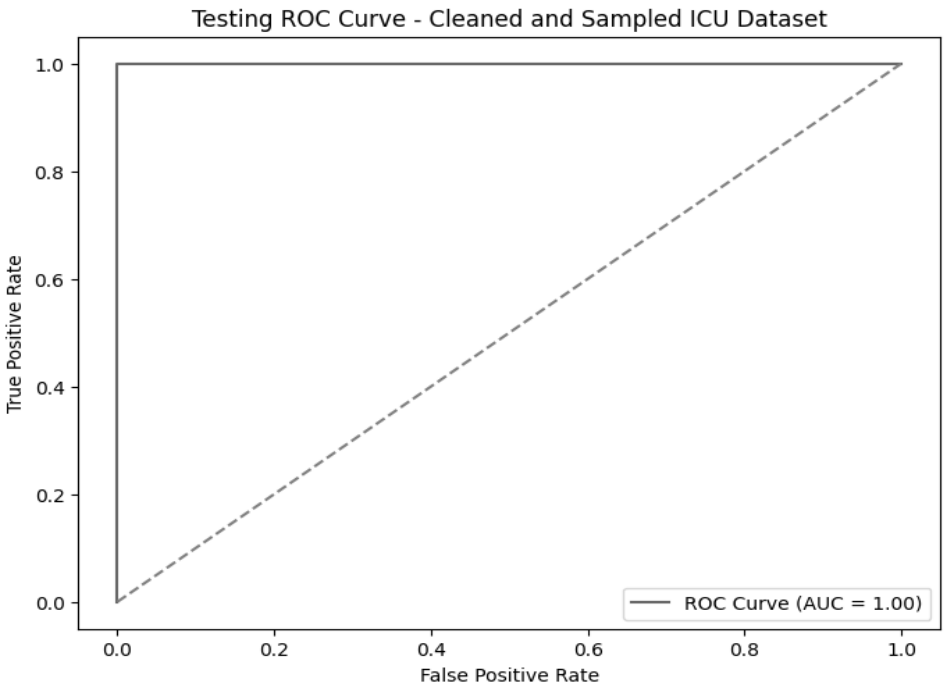


Figure 4.146: Receiver Operating Characteristics curve for the six-hidden-layer 70/30 MLP-Enhanced and Nadam-optimized model on the ICU dataset

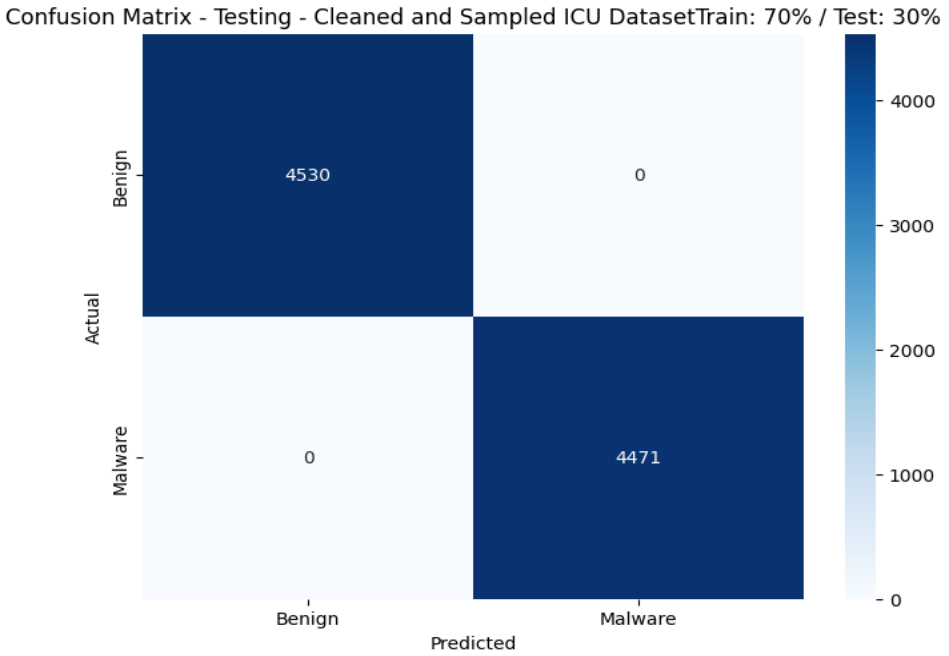


Figure 4.147: Confusion matrix for the six-hidden-layer 70/30 MLP-Enhanced and Nadam-optimized model on the ICU dataset

Analysis of the loss curves for a one-hidden-layer 80/20 split demonstrated the model's performance during threat classification. The loss for the first three epochs was 0, then increased towards 0.1 and slightly declined between the fifth and sixth epochs. This behavior in the first three epochs demonstrated the model's ability to predict the test samples perfectly. This loss spiked to 0.5 in the seventh epoch, then fell to slightly above 0.2 in the eighth. This spike was attributed to an adjustment in the model's learning rate, leading to improved model performance. The validation loss started at about 0.35 and steadily declined to 0.1 in the seventh epoch, then slightly increased to about 0.15 in the eighth epoch. The low validation loss demonstrated the model's ability to generalize to an unseen dataset. Figure 4.148 shows these observations:

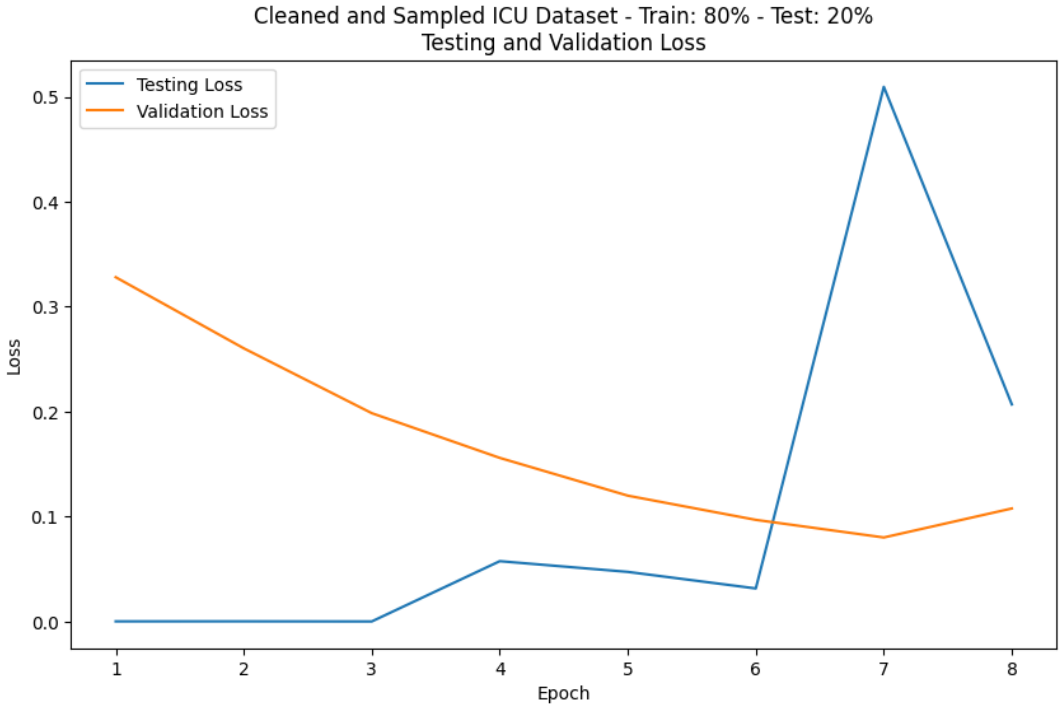


Figure 4.148: Testing and validation loss curves for one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on the ICU dataset

Assessment of the 60/40 four-hidden-layer model showed that the test loss remained at zero until the third epoch, then increased slightly to about 0.1 in the fourth epoch. It then maintained this score through the sixth epoch, then increased to about 0.5 in the seventh and eighth epochs, before declining to 0.5 in the ninth epoch and maintaining this score until the tenth epoch. The sudden jump in the loss curve was attributed to an adjustment in the learning rate, which introduced dynamism to the model. The validation loss started at about 3.4 and fell to 0.5 in the second epoch, then declined slightly to about 0.4 in the tenth epoch. While increasing the number of epochs for the model might have improved its performance, the overall loss scores were higher than those of the 80/20 one-hidden-layer model. As a result, one hidden layer's 80/20 variant offered a better learning environment than this model with the current number of epochs. This behavior is shown in Figure 4.149.

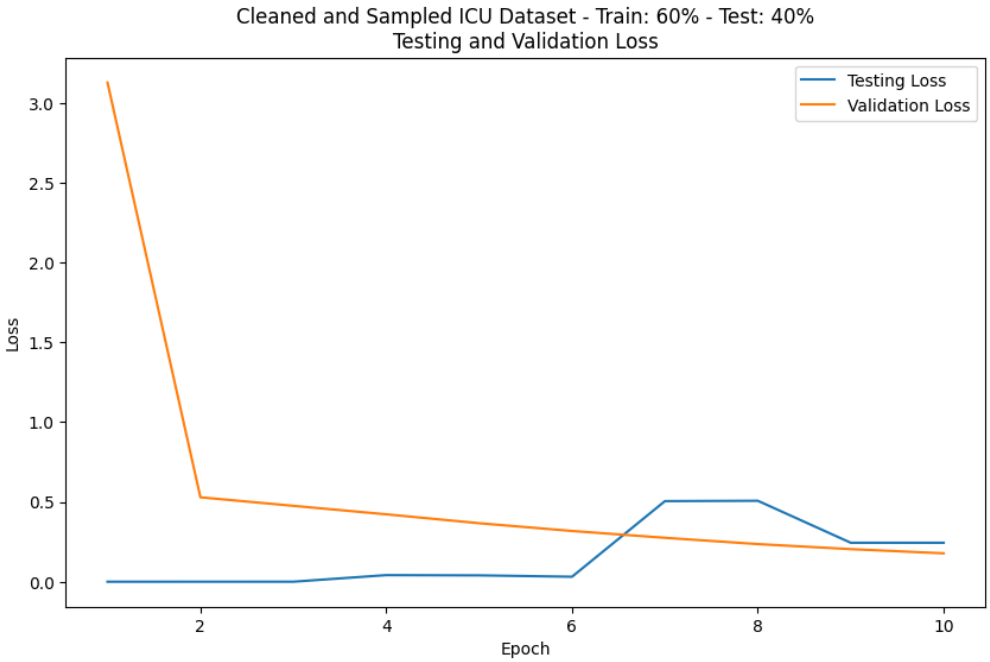


Figure 4.149: Testing and validation loss curves for a four-hidden-layer 60/40 MLP-Enhanced and Nadam-optimized model on the ICU dataset

A review of 70/30 variants with five and six hidden layers revealed differences in the model's performance across training/testing as the number of hidden layers increased. The testing loss for the five-hidden-layer model started at zero and remained at zero until the third epoch. The value increased slightly to about 0.2 in the fourth epoch and decreased slightly in the fifth before reverting to 0.2 in the sixth epoch. The value then increased to 0.5 in the seventh epoch and stabilized at this level until the thirteenth epoch. This increase in the seventh epoch was attributed to a change in the model's learning rate. The validation loss started high, at about 2.4, and fell to about 0.6 in the second epoch. From this phase, it declined steadily until it reached about 0.1 in the thirteenth epoch. The model's performance, as indicated by the loss curves, was still lower than that of the 80/20 variant with one hidden layer. These results are shown in Figure 4.150.

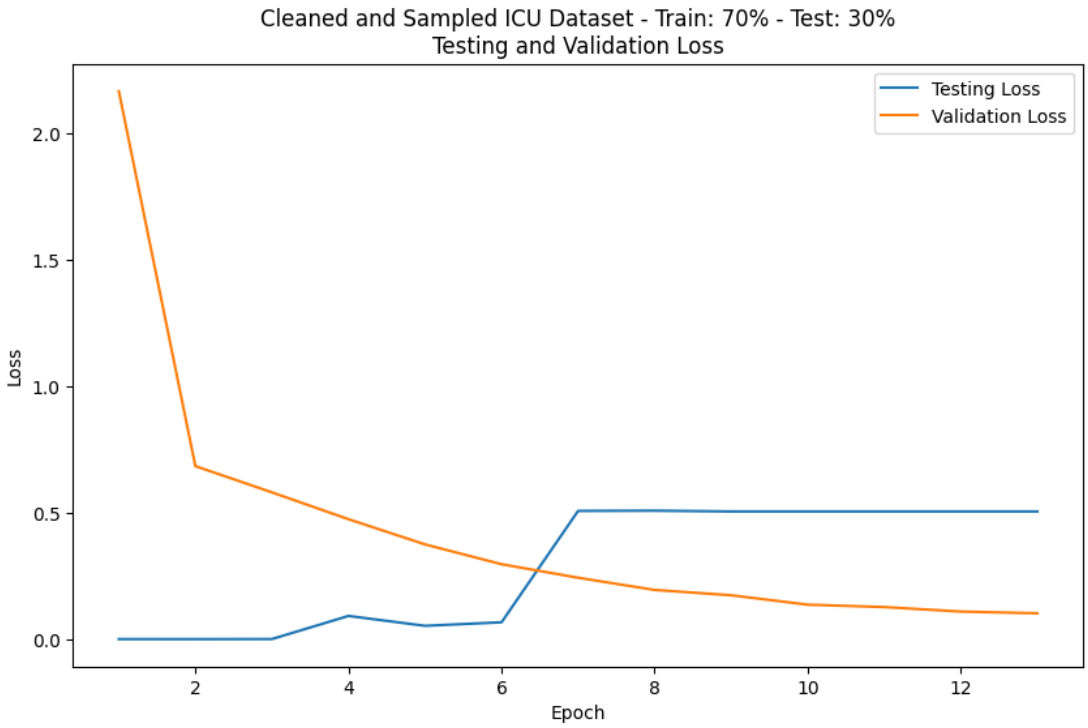


Figure 4.150: Testing and validation loss curves for a four-hidden-layer 60/40 MLP-Enhanced and Nadam-optimized model on the ICU dataset

The testing loss for the six-hidden-layer 70/30 split started at zero and remained at zero until the third epoch. It then slightly increased to about 0.5 in the fourth epoch, maintained the score in the sixth epoch, and then rose to about one and stabilized at this value until the thirteenth epoch. The validation loss started at about 3.75, fell to about 1.0 in the second epoch, and then steadily declined to about 0.5 in the eighth epoch. It then rises to 10 in the ninth epoch, before returning to 0.5 in the tenth epoch and slightly decreasing to about 0.4 in the thirteenth epoch. These loss scores were higher than those in the five-hidden-layers, demonstrating a substantial decline in the model's generalization capability as the number of hidden layers was increased. As a result, one hidden layer's 80/20 variant was selected for adoption. The loss curves for the six-hidden-layer 70/30 variant are shown in Figure 4.151.

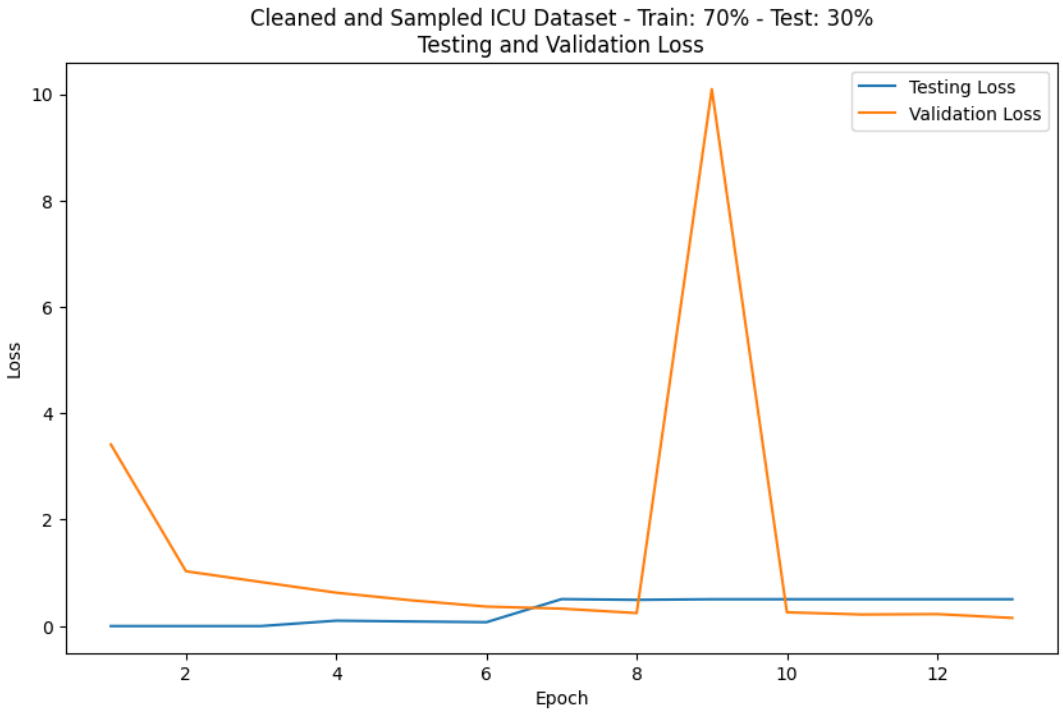


Figure 4.151: Testing and validation loss curves for a five-hidden-layer 60/40 MLP-Enhanced and Nadam-optimized model on the ICU dataset

The analysis of loss curves for the selected models offers insights into their testing performance. For instance, the training and validation loss curves for one hidden layer's 80/20 variant show even greater improvement. The training loss started at 0.40 while the validation loss started at 0.345. The two curves declined, with the training loss curve reaching 0.05 in the eighth epoch and validation loss achieving 0.13 in the same epoch. Since the curves did not stabilize, additional training might have yielded slightly better results. The training loss curve from the hidden layers 60/40 model started at 0.55 and slowly declined to 0.45 in the fourth epoch, alongside the validation loss, which had started at 0.35. This situation implies that the model would not have significantly improved its performance even with additional training epochs.

The training loss curve from the five-hidden-layer' 70/30 model started at 0.9 and declined to about 0.2 by the 12th epoch, while the validation loss started at 2.5 but quickly fell to 0.55 in the second epoch. This validation loss then followed the training loss's behavior, with both achieving nearly similar values in the thirteenth epoch. This behavior demonstrates that the model could have been slightly improved with additional training epochs. The loss curve for the six-hidden-layer 70/30 variant started at about 1.75, while the validation loss started at about 3.75. The two models declined,

reaching about 0.8 in the eighth epoch. While the training loss evened out until the 13th epoch, the validation loss increased to 10 in the 9th epoch before falling again to about 0.75 in the 10th, and then evened out until the 13th. These results demonstrate that the model would have slightly improved with additional training epochs, as it would have quickly stabilized. These results are shown in Figures 4.152, 4.153, 4.154 and 4.155.

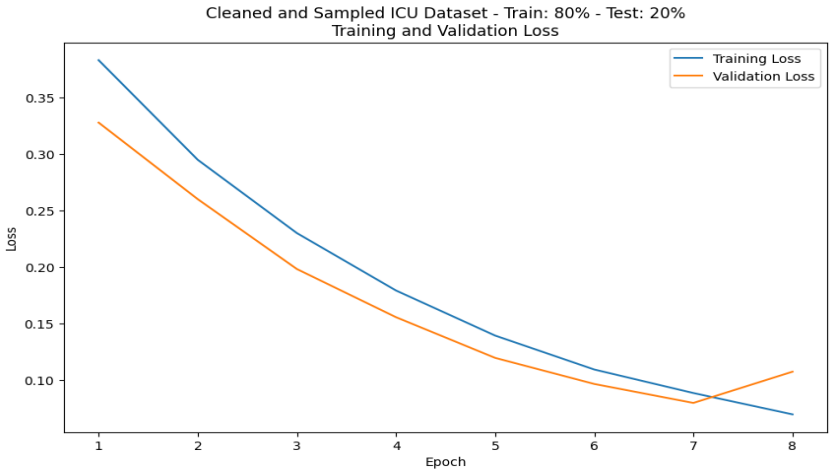


Figure 4.152: Training and validation loss curves for one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on the ICU dataset

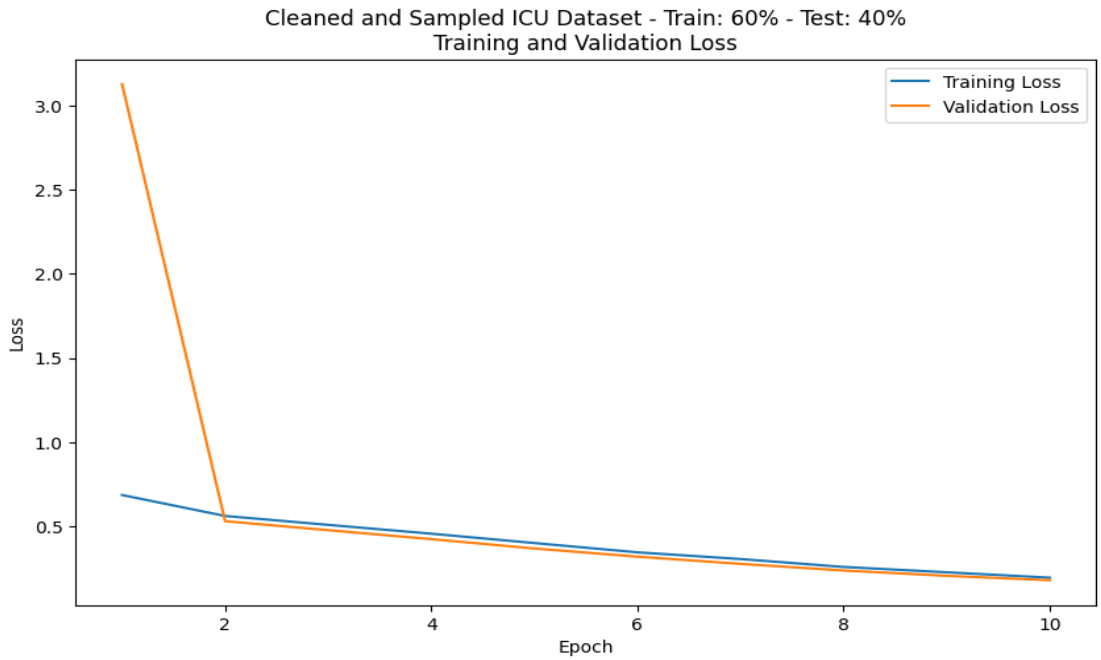


Figure 4.153: Training and validation loss curves for a four-hidden-layer 60/40 MLP-Enhanced and Nadam-optimized model on the ICU dataset

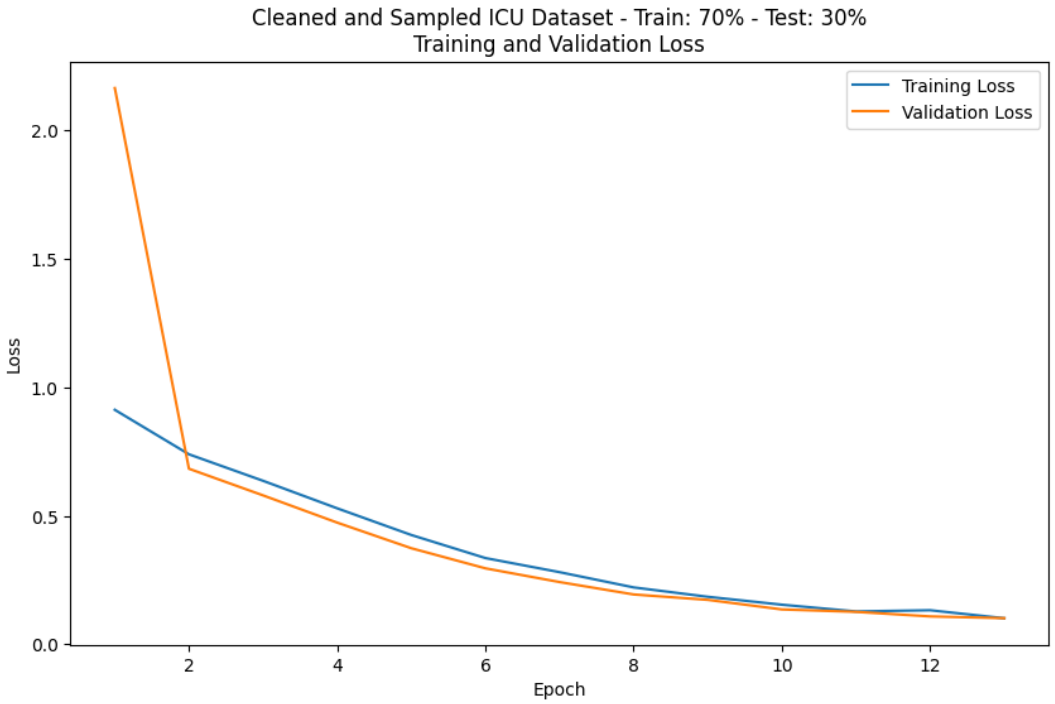


Figure 4.154: Training and validation loss curves for a five-hidden-layer' 70/30 MLP-Enhanced and Nadam-optimized model on the ICU dataset

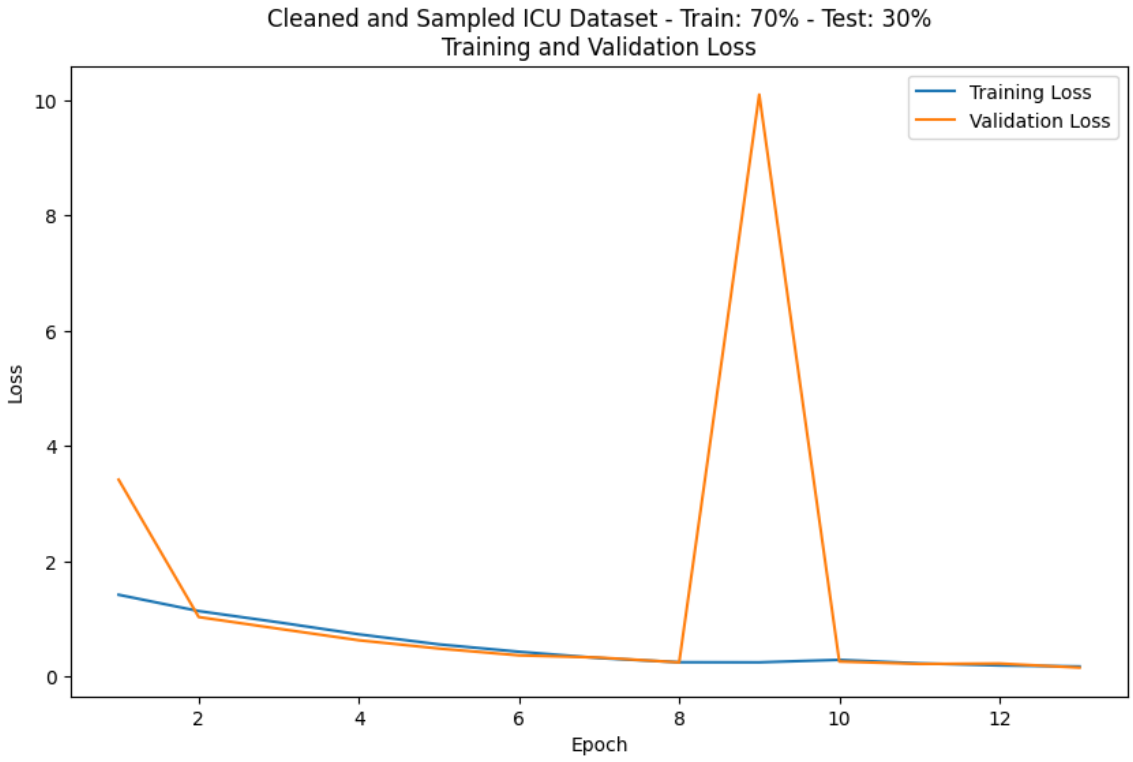


Figure 4.155: Training and validation loss curves for a six-hidden-layer 70/30 MLP-Enhanced and Nadam-optimized model on the ICU dataset

4.6.2 Enhanced Hybrid Model on the WUSTL Dataset

The analysis of the Multilayer Perceptron-enhanced Nadam-optimized hybrid model on the WUSTL dataset provides insights into its comparative performance with machine learning and baseline models. The Random Forest models dominate the recall and accuracy, recording scores of 86.57% for 80/20, 84.04% for 70/30, and 82.77% for 60/40. Furthermore, these models achieved high specificity and precision. Although the enhanced hybrid models outperform some baseline models, only three models reach 75% for accuracy, recall, precision, specificity, and F1. These models are the 70/30 variant with one hidden layer and the 80/20 variants with four and two hidden layers. Further, these models showed relatively low false-positive rates. The machine learning and baseline models dominated the detection speeds, recording low inference speeds. The 80/20 variant of the three-hidden-layer MLP-enhanced Nadam-optimized model had the best detection speed among the hybrid models, at 1.33 seconds. These results are shown in Figures 4.156, 4.157, 4.158, 4.159, 4.160, and 4.161.

Three models emerge as the dominant variants of the hybrid model in the WUSTL dataset, warranting further analysis to determine the optimal model for the dataset. These models were the one-hidden-layer's 70/30, four-hidden-layers' 80/20, and two-hidden-layer's 80/20 variants. A comparative assessment of the model's recall, precision, specificity, accuracy, and F1 scores showed that a 70/30 split with one hidden layer was the leading approach, followed by an 80/20 split with four hidden layers. The 80/20 variant of the one-hidden-layer model had a recall and accuracy of 79.32%, precision of 79.92%, specificity of 79.74%, an F1 of 79.24%, and an FPR of 20.59%. The 80/20 four-hidden-layer model achieved recall and accuracy of 75.58%, precision of 76.38%, specificity of 75.65%, F1 of 75.42%, and FPR of 24.35%. This behavior was confirmed by the model's regression metrics, which showed that a 70/30 configuration with one hidden layer had a mean absolute error and mean squared error of 0.21, and a root mean squared error of 0.45. Four-hidden-layers' 80/20 had an MAE and MSE of 0.244, and an RMSE of 0.49. The 80/20 two-hidden-layer configuration had an MAE and MSE of 0.25 and an RMSE of 0.50. These scores are shown in Figures 4.162 and 4.163.

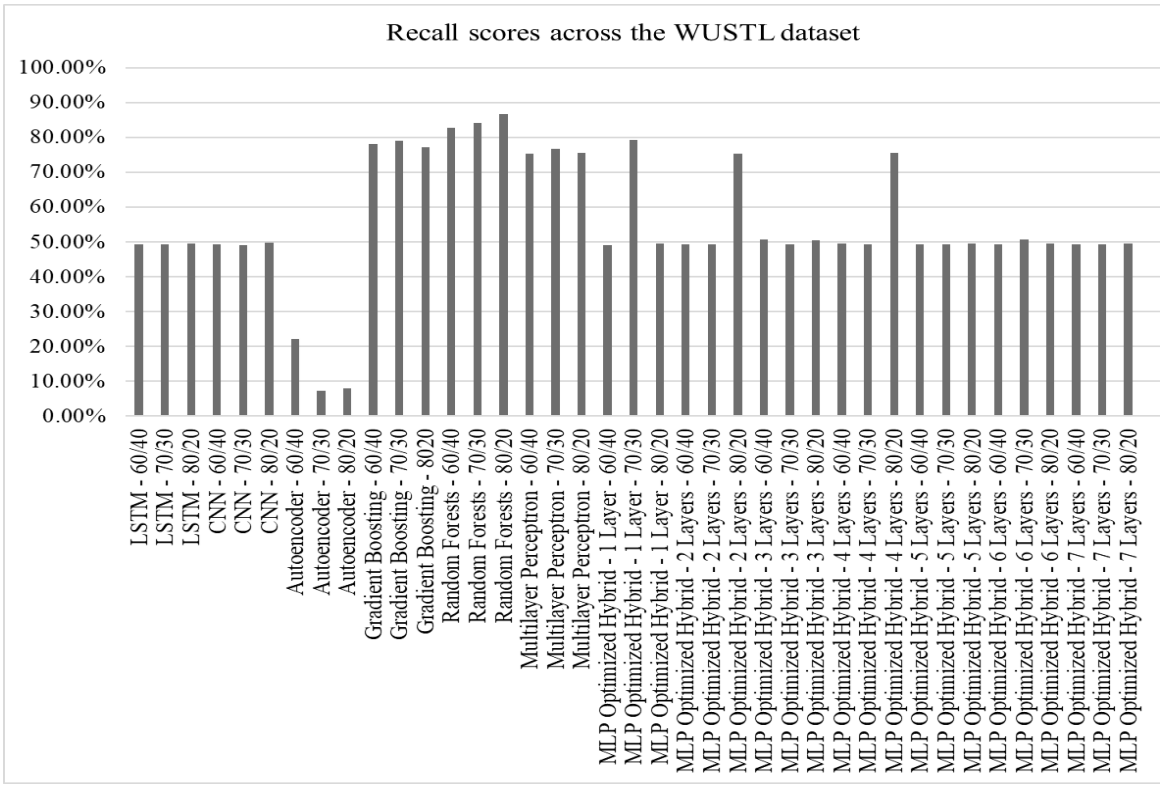


Figure 4.156: Recall scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on WUSTL dataset

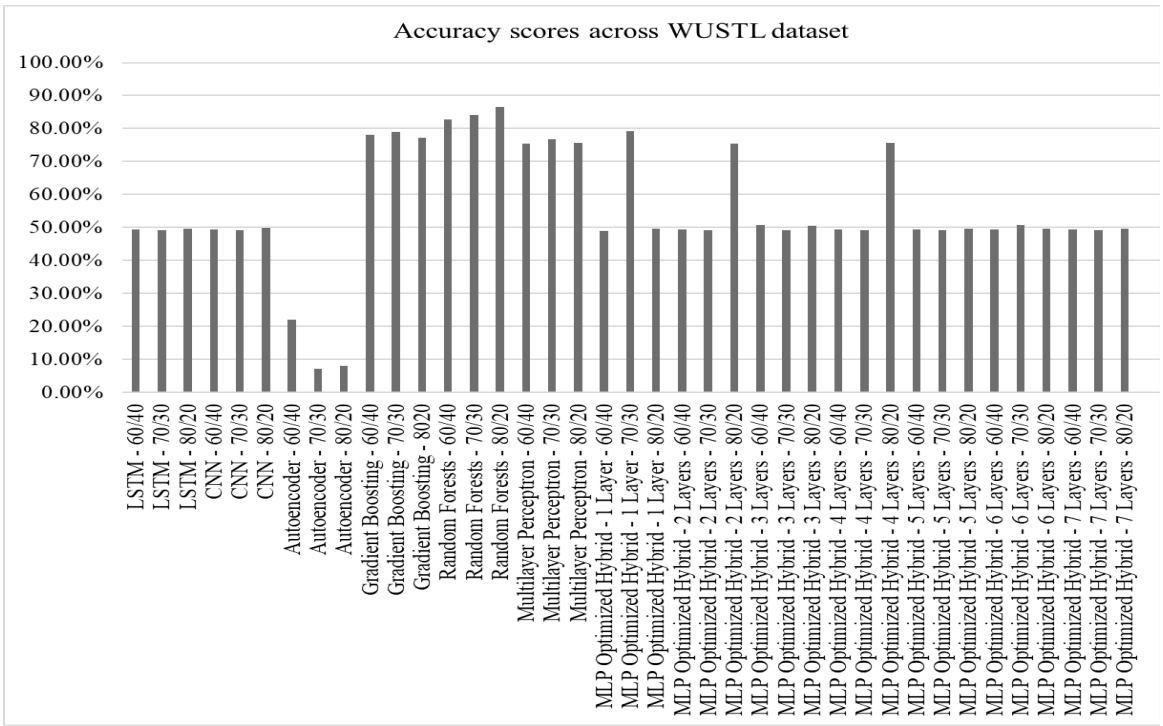


Figure 4.157: Accuracy scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on WUSTL dataset

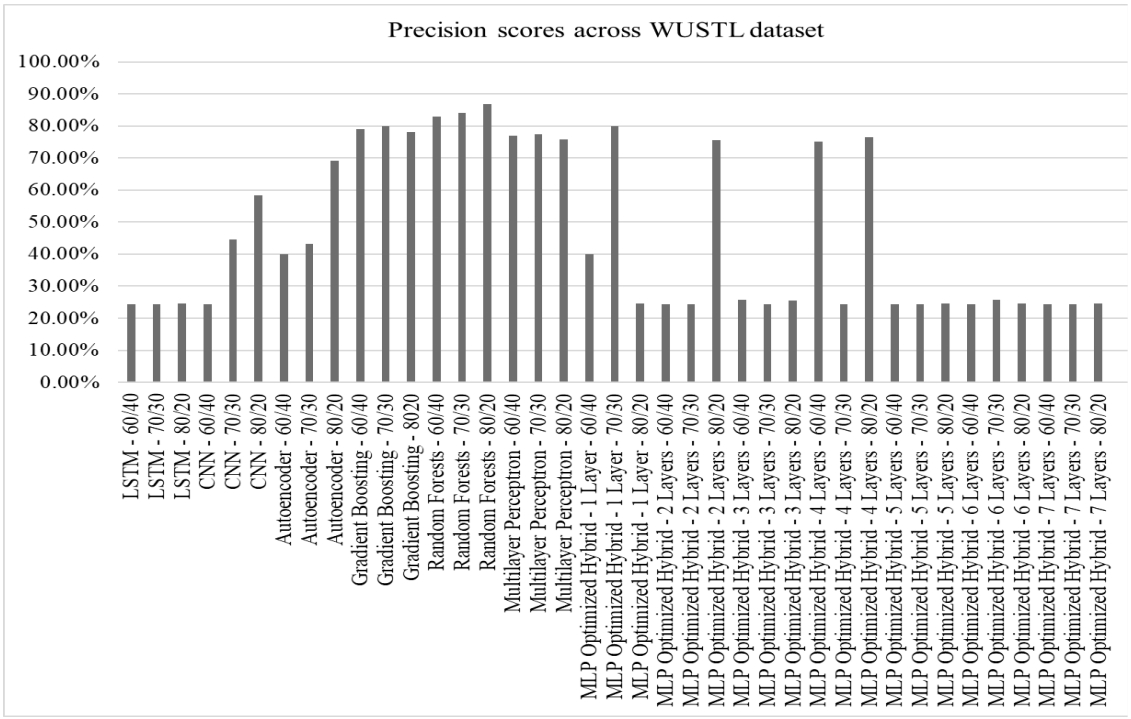


Figure 4.158: Precision scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on WUSTL dataset

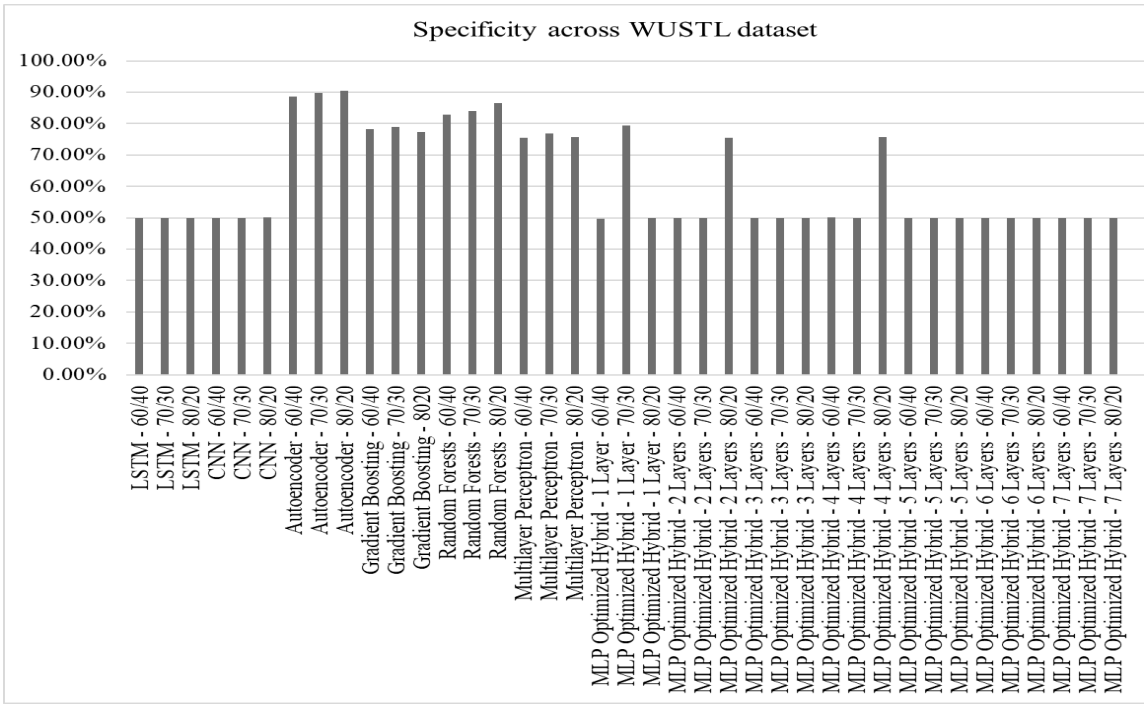


Figure 4.159: Specificity scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on WUSTL dataset

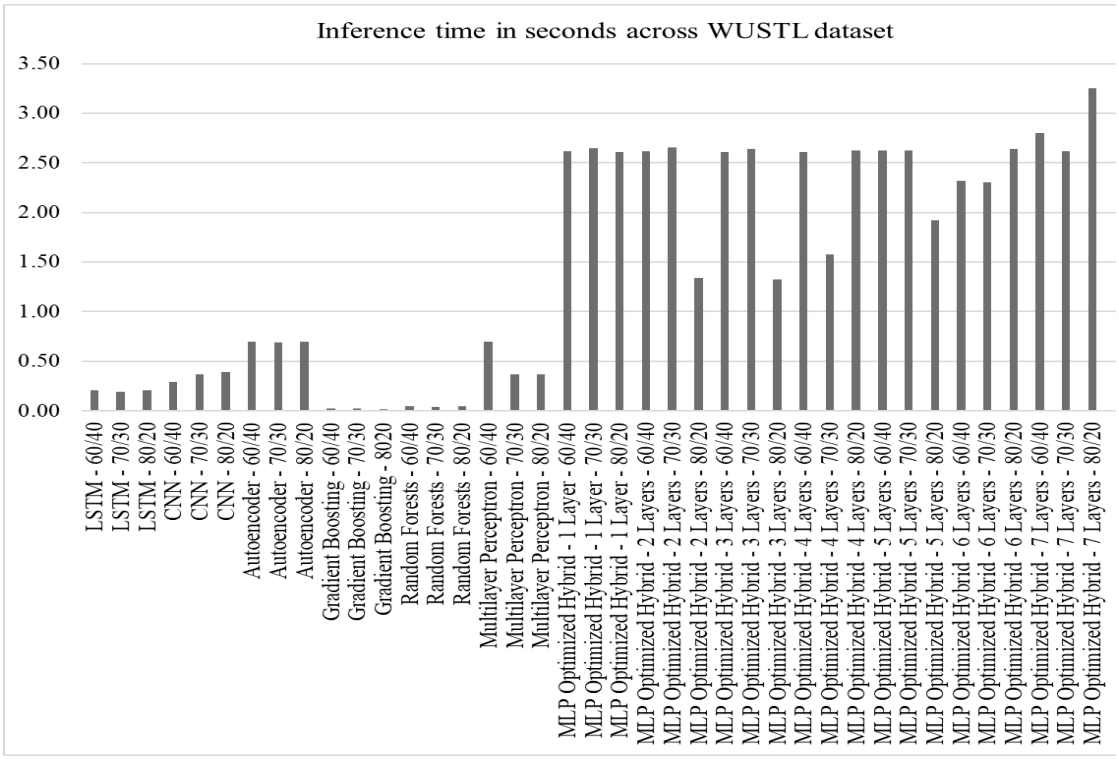


Figure 4.160: Inference time for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on WUSTL dataset

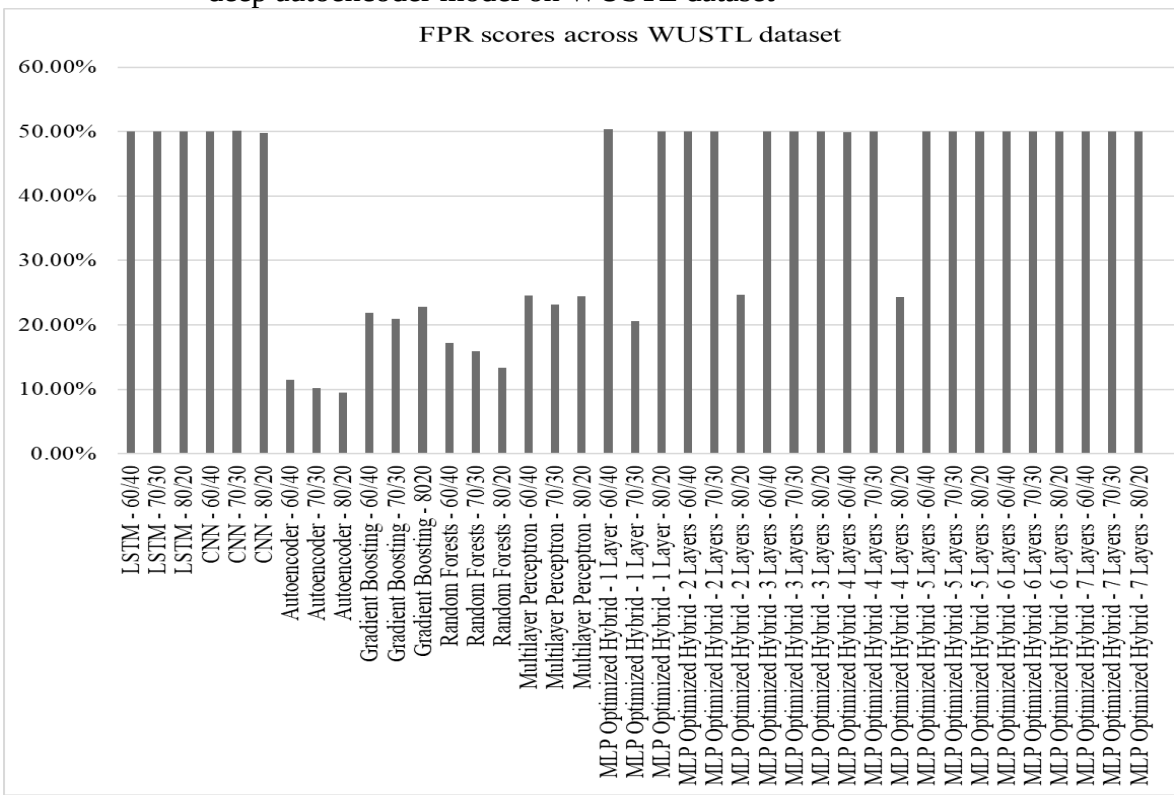


Figure 4.161: False positive rate for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on WUSTL dataset

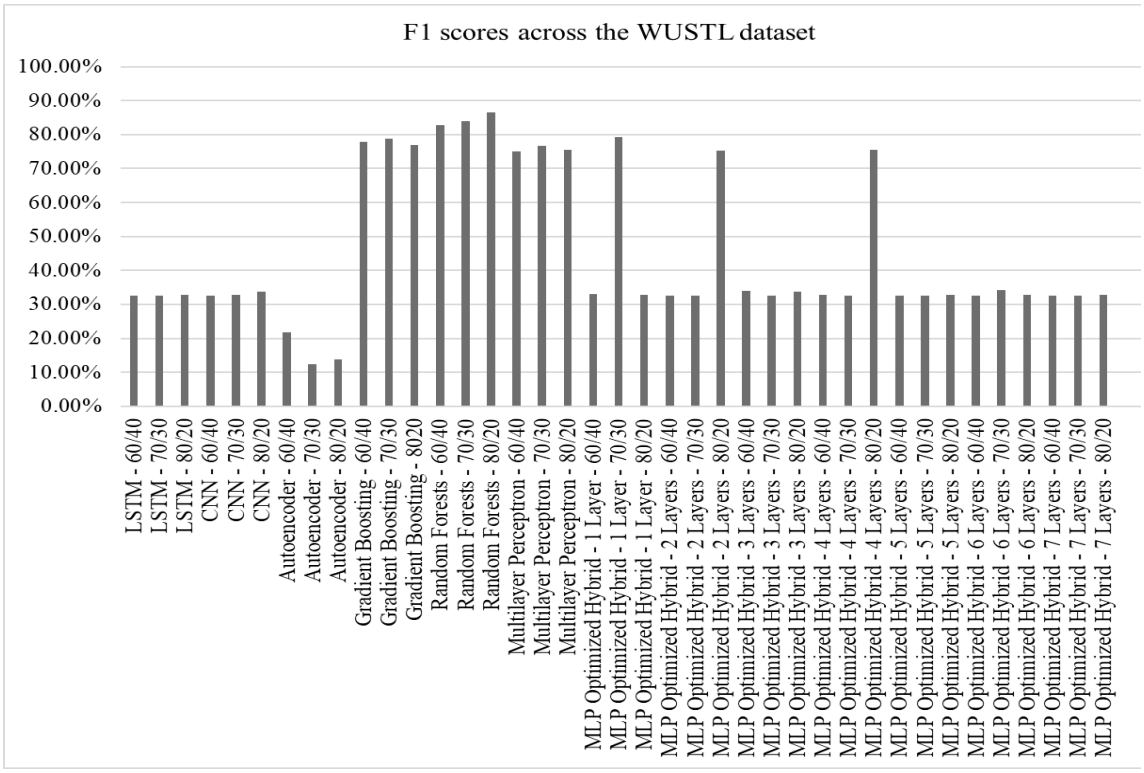


Figure 4.162: F1 scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on WUSTL dataset

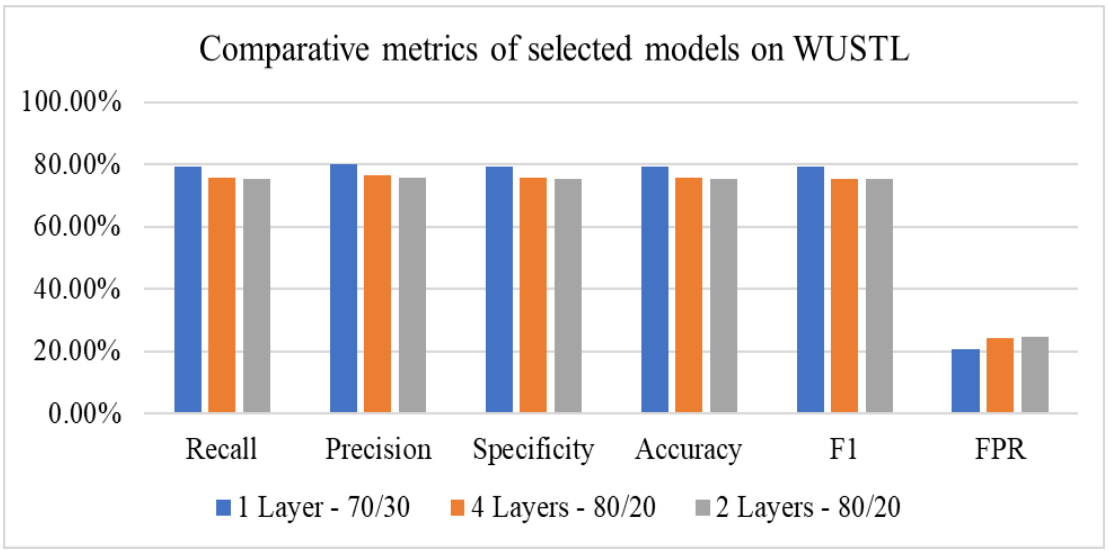


Figure 4.163: Comparative view of performance metrics for selected hybrid models on the WUSTL dataset

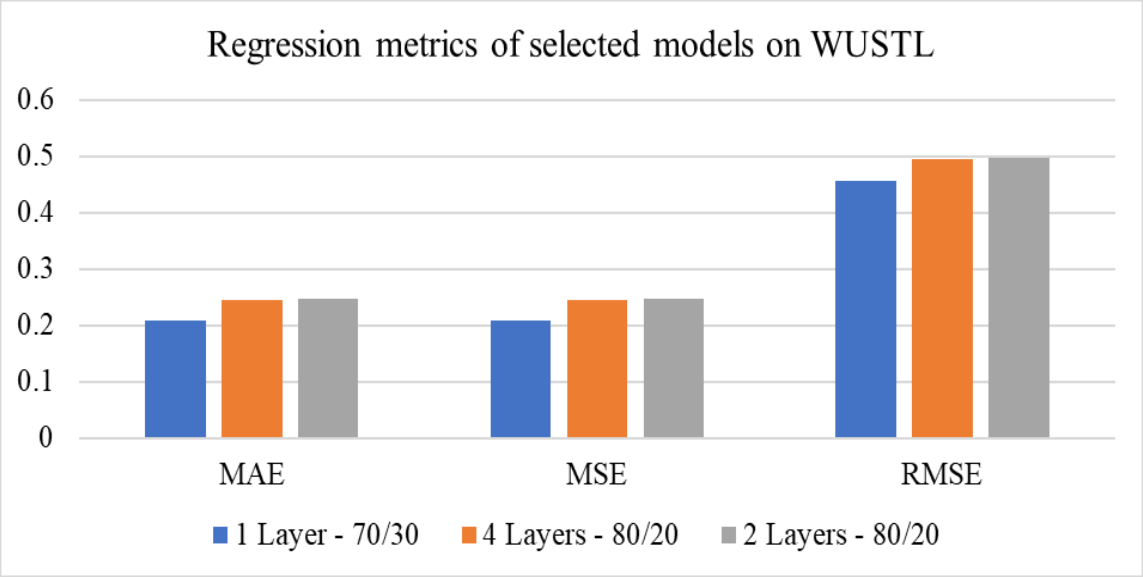


Figure 4.164: Comparative view of regression metrics for selected hybrid models on the WUSTL dataset

Analysis of the confusion matrices and ROC curves for the selected models revealed their performance in classifying threats in the WUSTL dataset. For example, in one hidden layer's 70/30 variant, the false-positive cases were 85, while the false negatives were 169. This phenomenon implies that 85 connections were identified as benign, while they were malicious, potentially exposing the infrastructure to attacks. While this accounts for 6.92% of the classification results, it demonstrates the model's limitations in achieving optimal efficiency. This behavior was further supported by the ROC curve, which showed an AUC of 0.79. While the model was reliable at classifying threats, it was less efficient, falling short of the 0.99 threshold for security models. Nevertheless, the relatively high score demonstrated its suitability in detecting threats for the dataset. These scores are shown in Figures 4.165 and 4.166.

The analysis of the confusion matrix and ROC curve for the 80/20 four-hidden-layer variation provided insights into its detection performance. The model had 65 false positives and 135 false negatives. This phenomenon implies that the false incident cases accounted for 7.94% of the overall detection in the WUSTL dataset. This score was lower than that of the 80/20 variant with one hidden layer. Further, the model had an AUC score of 0.76, slightly lower than that of the 70/30 variant with one hidden layer. This situation implies that the model did not learn better than the latter, but provided a

reliable detection environment when compared to others. These results are shown in Figures 4.167 and 4.168.

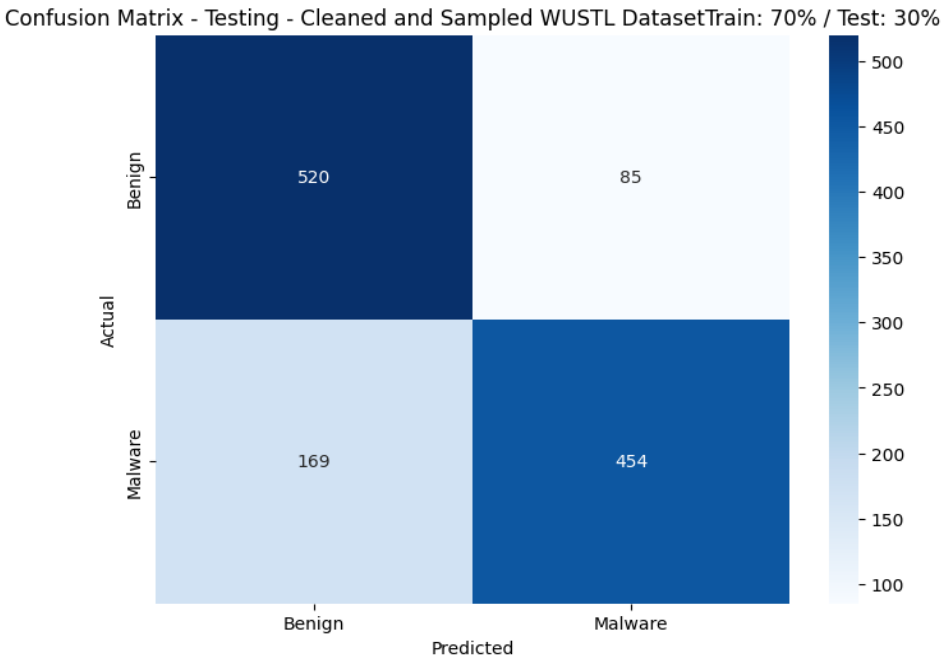


Figure 4.165: Confusion matrix for one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

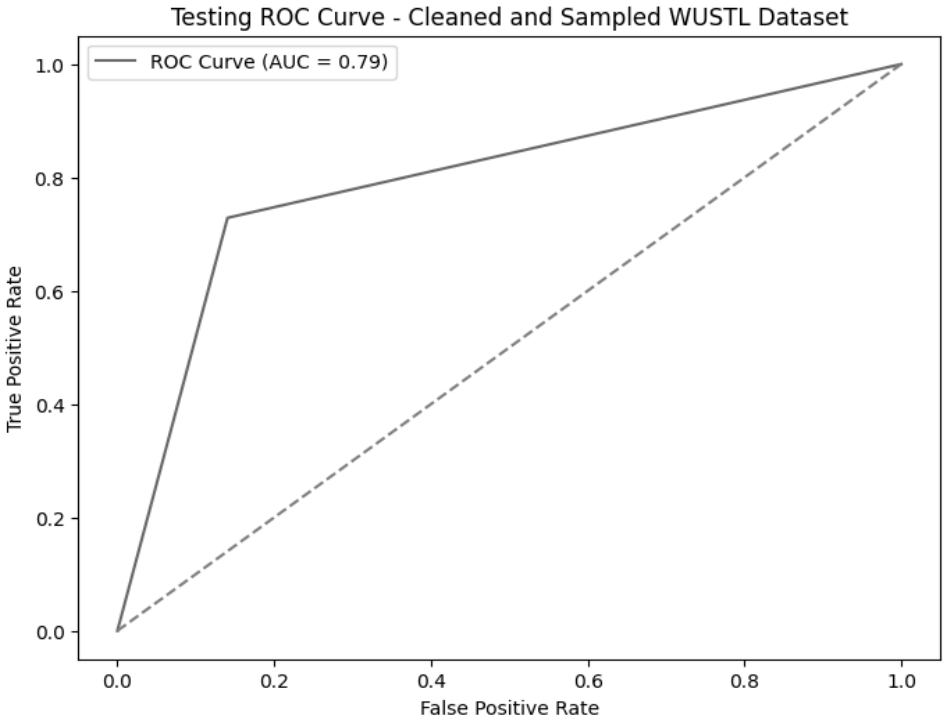


Figure 4.166: Receiver Operating Characteristics curve for one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

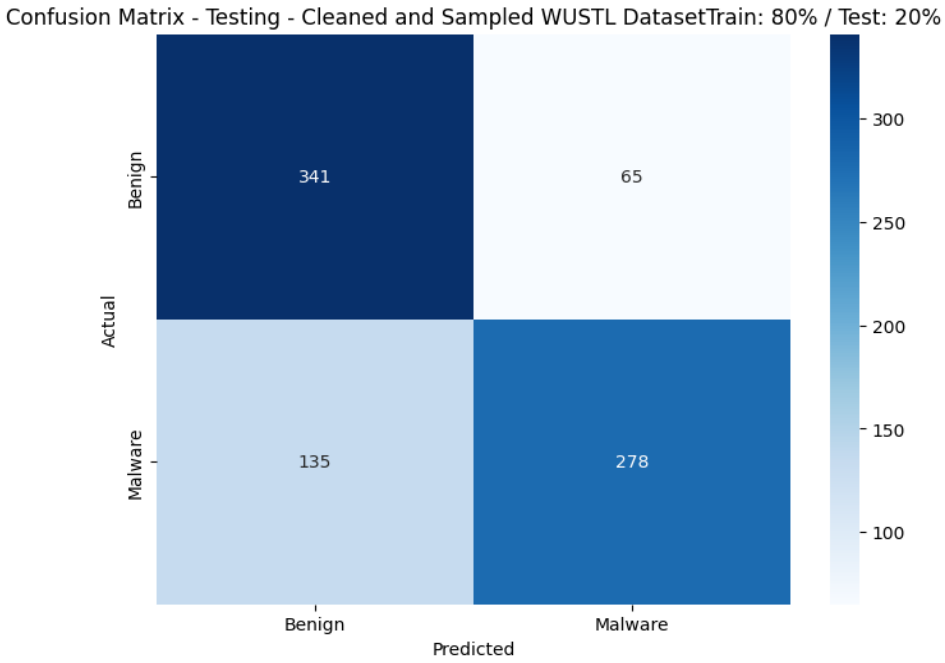


Figure 4.167: Confusion matrix for four-hidden-layers' 80/20 MLP-Enhanced and Nadam-optimized model on WUSTL dataset

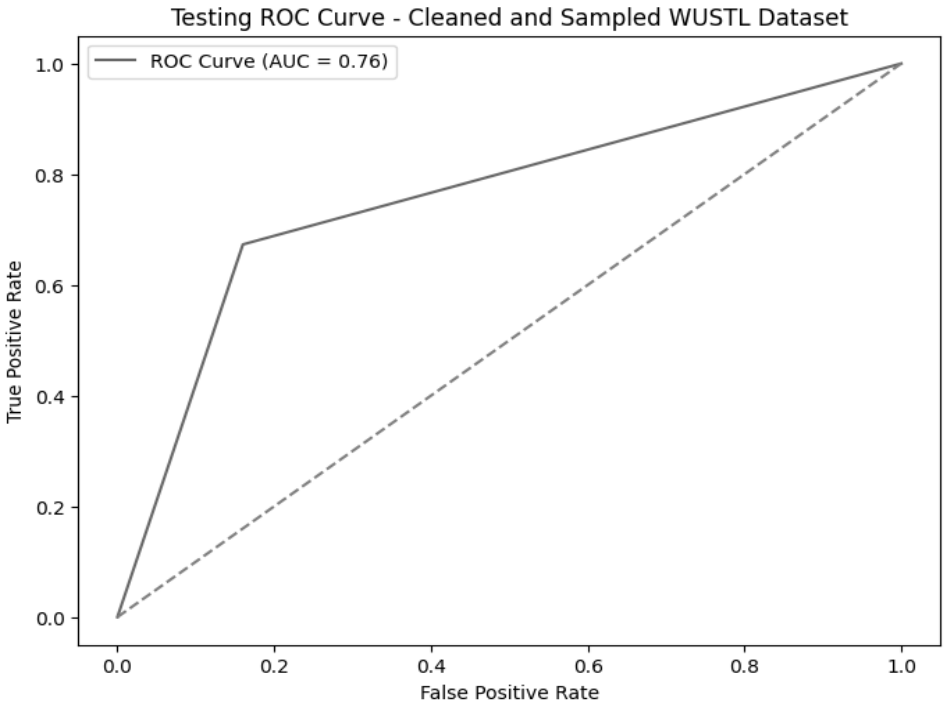


Figure 4.168: Receiver Operating Characteristics curve for a four-layer 80/20 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

The review of the confusion matrix and ROC curve for the two-hidden-layers' 80/20 variant offered insights into the model's performance as the number of hidden layers was reduced from four in the preceding scenario. This model had 85 false positives, an

increase from the 65 recorded in the four-hidden-layers model with the same training/testing ratio. This suggests that increasing the number of hidden layers from 2 to 4 improves overall model accuracy. The false negative cases for the model were 117, which was an improvement from the 135 cases reported in the four-hidden-layer'. Further review of the confusion matrix indicates that the number of true negative cases decreases from 341 to 321, while the number of true positive cases increases from 278 to 296. This phenomenon implies that the model improved its ability to identify positive cases more than negative ones. The analysis of the ROC curve indicated that the model achieved an AUC of 0.75, indicating reliable detection capability. While this score was slightly lower than that obtained with four hidden layers' 80/20, it still showed significant detection capability. The confusion matrix and ROC curve for the two-hidden-layers' 80/20 variant are shown in Figure 4.169 and 170.

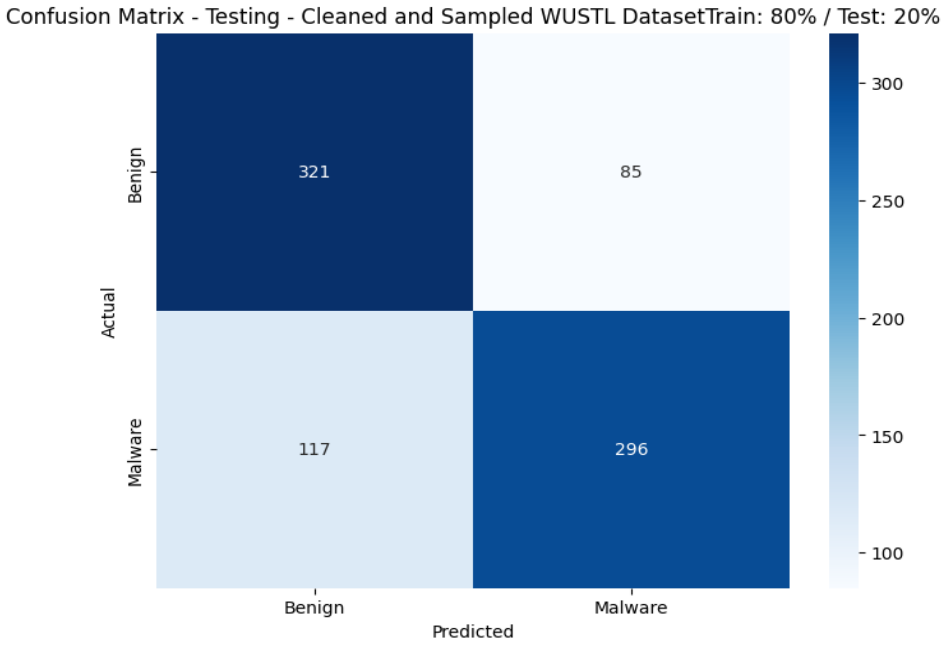


Figure 4.169: Confusion matrix for two-hidden-layers' 80/20 MLP-Enhanced and Nadam-optimized model on WUSTL dataset

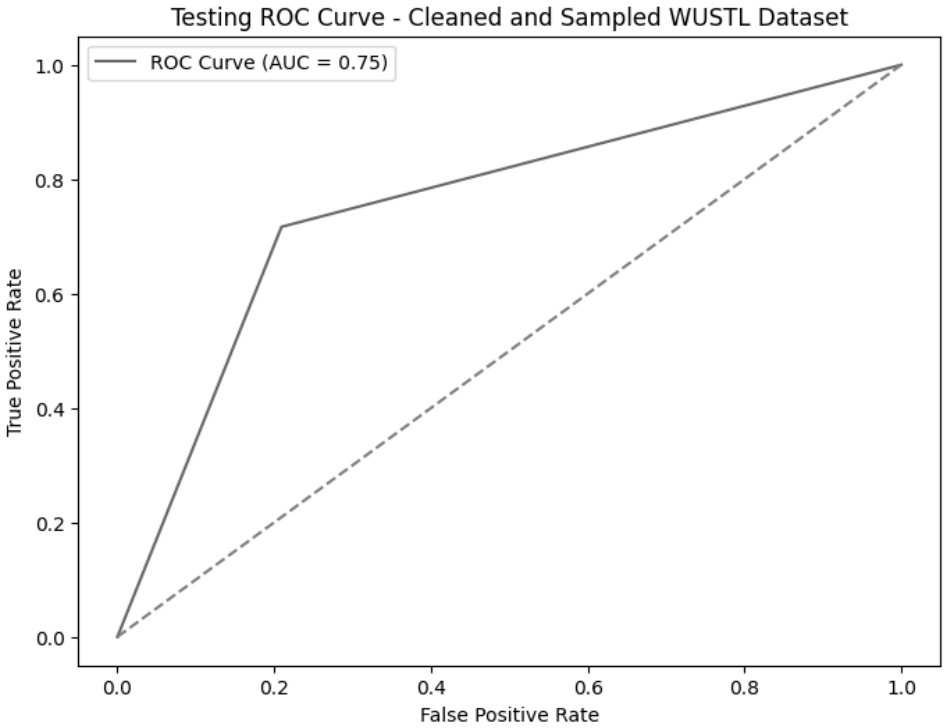


Figure 4.170: Receiver Operating Characteristics curve for a two-hidden-layer 80/20 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

Analysis of the 70/30 variant's loss curves for one hidden layer enhanced understanding of its detection performance, offering insights into the rationale for its accuracy scores. For instance, the testing loss for the model started at zero and remained around 0.1 for about three epochs, then rose slightly to 0.1 in the fourth epoch and fell to 0 in the fifth epoch. It then grew to about 0.6 in the seventh epoch, attributed to a learning rate adjustment. It then fell to 0.2 in the eighth epoch and rose to about 0.5 in the ninth epoch, where it stabilized until the end of the training session. On the other hand, validation loss started at about 1.1, decreased to 1.0 in the third epoch, then to 1.5 in the fourth, and then to 1.3 in the ninth, before falling to about 0.8 in the 12th. This loss curve wobbled for the remaining epochs, indicating its instability. Increasing the model's training epochs may help it stabilize within this range, leading to higher accuracy. This behavior is shown in Figure 4.171.

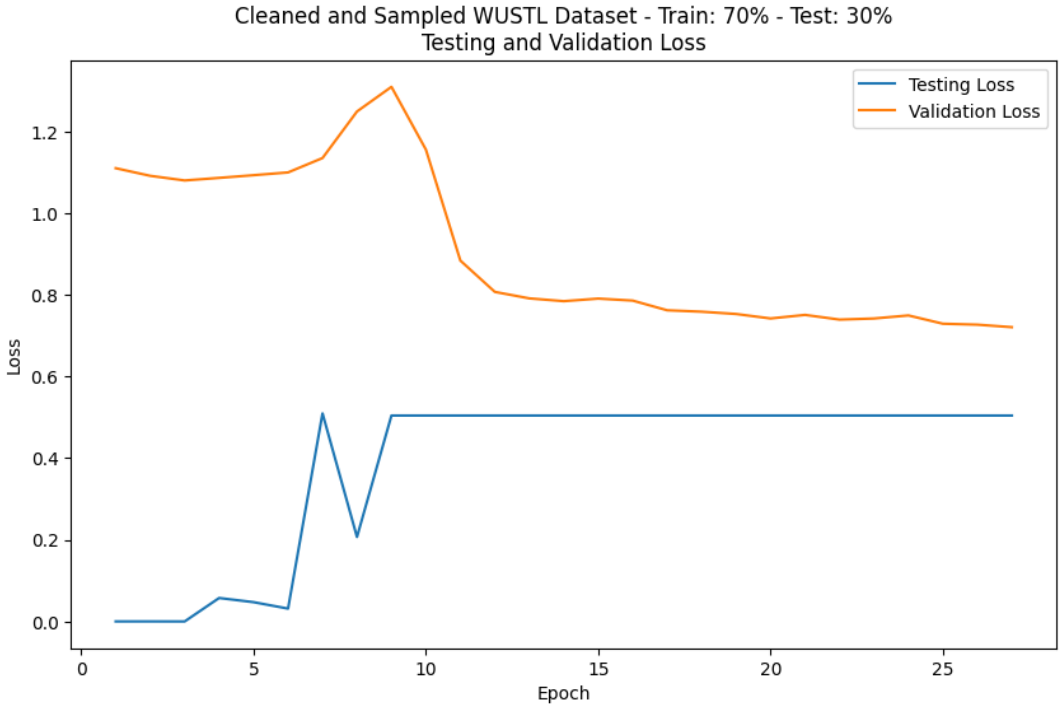


Figure 4.171: Testing and validation loss curves for one hidden layer's 70/30 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

Analysis of the four-hidden-layer' 80/20 variant showed a similar experience for the dataset. The testing loss started at 0, increased slightly to 0.1 in the fourth epoch, and then rose to 0.5 in the seventh epoch. This rise was attributed to the adjustable learning rate, which modifies the learning rate after five epochs if the model fails to stabilize. It evened out slightly until the eighth epoch, then fell to 0.5 in the ninth epoch and stabilized until the end of the training session. The validation loss curve started high at 1.4, slightly declined to 1.3 in the second epoch, then increased to about 1.5 in the seventh epoch, and then fell to 1.0 in the tenth epoch. It then wobbled slightly throughout the training session. This behavior indicated that with additional training epochs, the model may achieve stability, increasing its accuracy and overall performance. These results are shown in Figure 4.172.

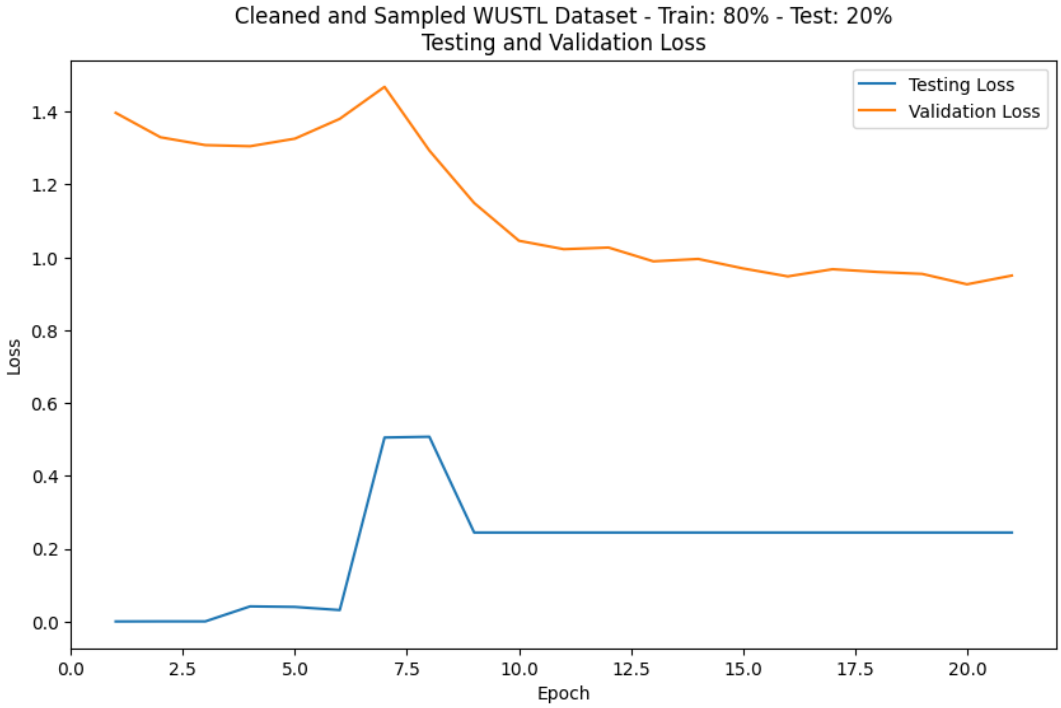


Figure 4.172: Testing and validation loss curves for a four-hidden-layer' 80/20 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

The analysis of the two-hidden-layer 80/20 loss curve showed behavior consistent with the model's other loss curves. The testing loss started at 0, increased to about 0.1 in the fourth epoch, rose to 0.5 in the seventh, leveled off slightly in the eighth, and fell to 0.3 in the ninth. It stabilized at this level until the end of the training. On the other hand, the validation loss started at 1.1, declined to 1.0 in the fourth epoch, rose to 1.1 in the sixth epoch, and then declined unevenly to 0.8 in the ninth epoch. From this point, the model wobbled, with uneven decline until the end of the training session. This phenomenon demonstrated that the model may fail to generalize effectively to an unseen dataset with the current training. However, additional training epochs may enable the model to stabilize and improve its performance. Nevertheless, the 70/30 variant of the one-hidden-layer model was recommended for adoption in this dataset due to its higher scores than those of the other models. This behavior for the two-hidden-layers' 80/20 variant is shown in Figure 4.173.

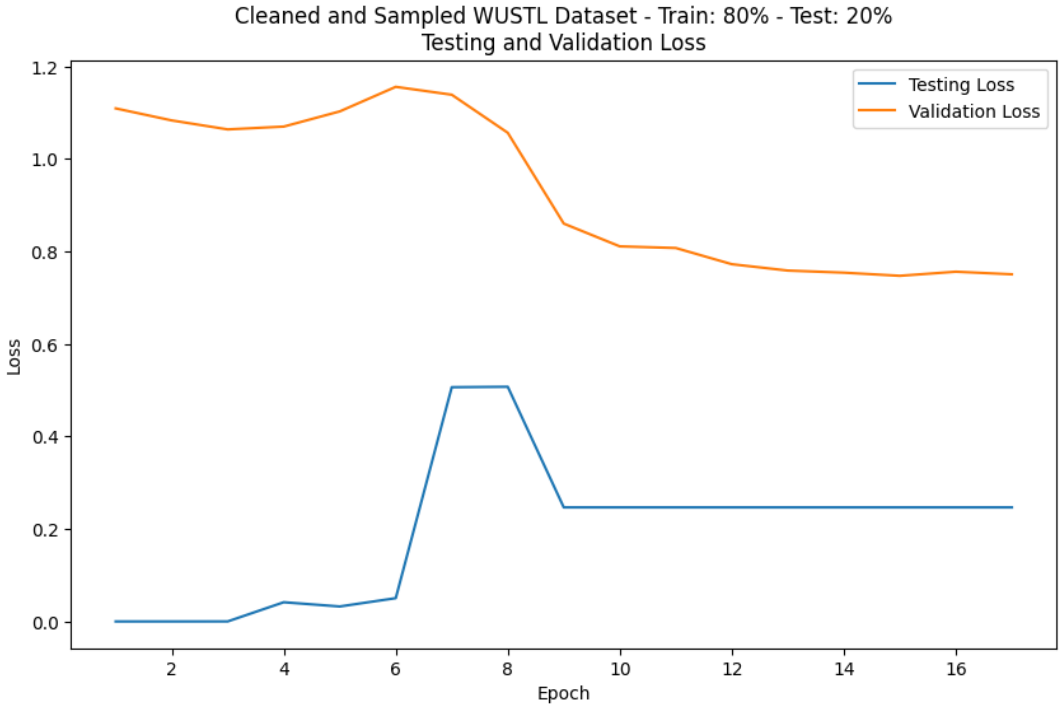


Figure 4.173: Testing and validation loss curves for a two-hidden-layer 80/20 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

The review of training loss curves for the selected models enhances understanding of these models. For instance, the loss curve for one hidden layer's 70/30 model started at 1.15 but wobbled towards 0.7 by the 25th epoch. The validation loss began at 1.12, then rose to 1.3 in the 9th epoch, before declining to about 0.8 in the 12th epoch, and then wobbled until the end of the training session. The training and validation loss curves for the 80/20 variants of four- and two-hidden-layer models demonstrated similar behavior, with few disparities in the initial training and validation loss curves. This behavior demonstrates imbalances in the dataset, leading to substantially low model scores. As a result, there is a need to address these limitations in the dataset to improve the training performance. This approach would yield results that are nearly as good as those in the ICU dataset. These results are shown in Figure 4.174, 4.175 and 4.176.

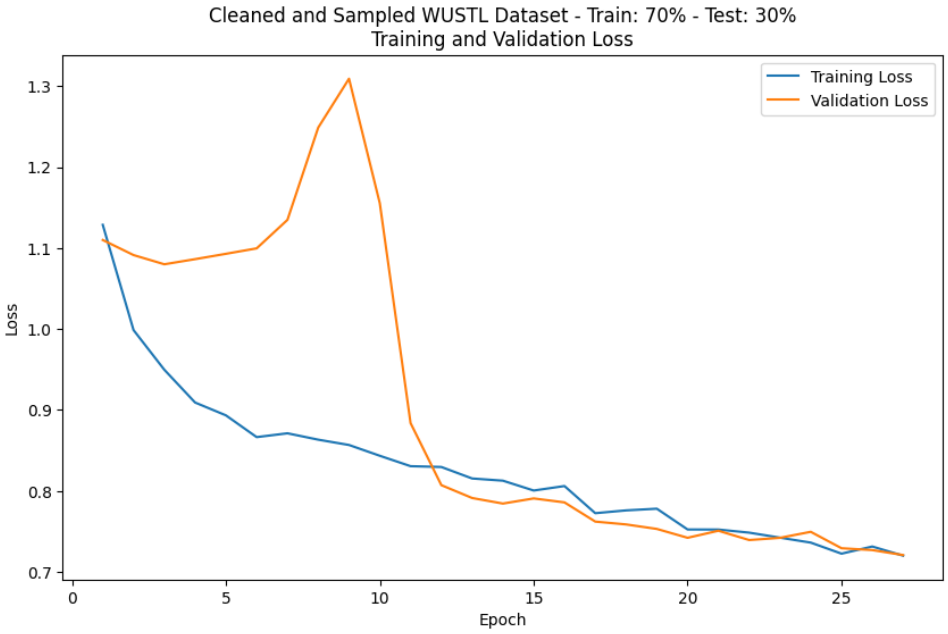


Figure 4.174: Training and validation loss curves for one hidden layer's 70/30 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

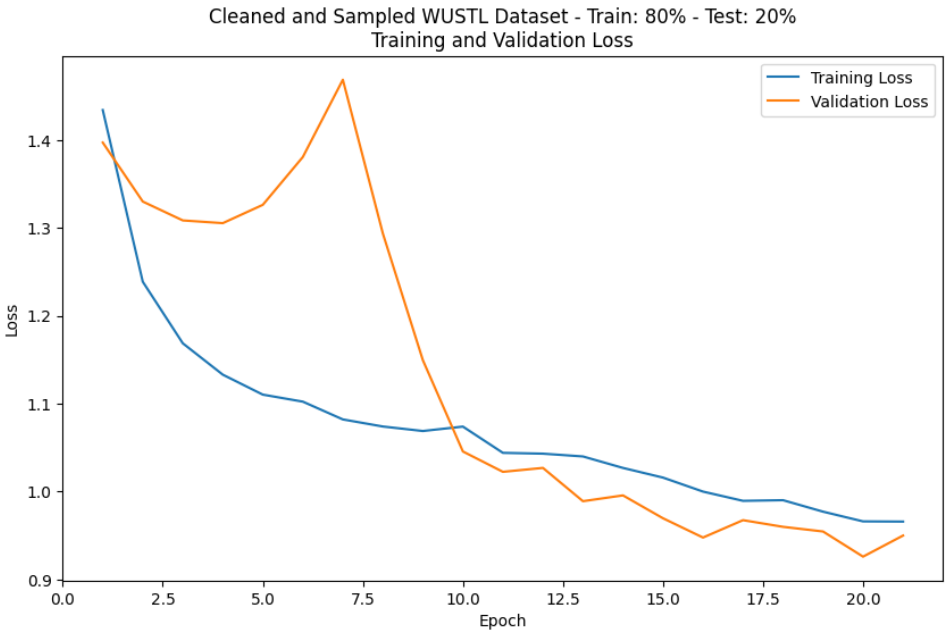


Figure 4.175: Training and validation loss curves for a four-hidden-layer' 80/20 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

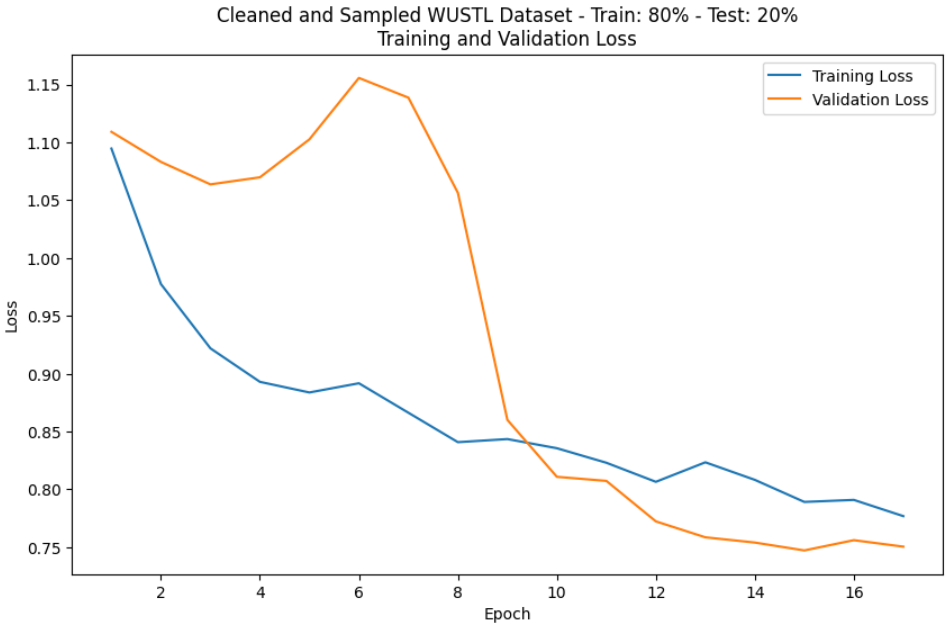


Figure 4.176: Training and validation loss curves for a two-hidden-layer 80/20 MLP-Enhanced and Nadam-optimized model on the WUSTL dataset

4.6.3 Enhanced Hybrid Model on the Edge IIoT Dataset

The analysis of the MLP-enhanced Nadam-optimized hybrid model on the Edge IIoT dataset provides insights into its comparative performance with the baseline and machine learning models. Some machine learning models, such as 80/20 variants of random forests and MLPs, dominated recall, accuracy, precision, specificity, and F1 scores. However, two hybrid models retained at least 95% for recall, specificity, accuracy, precision, and F1 scores. These models demonstrated competitive performance compared with the baseline and machine learning models in terms of false-positive rates. Although the gradient boosting and random forest models had very low inference times, the hybrid model performed reasonably well compared to MLPs, autoencoders, and CNN variants. This comparative performance demonstrates the model's viability for identifying threats, given its other performance aspects. These results are shown in Figures 4.177 to 4.183.

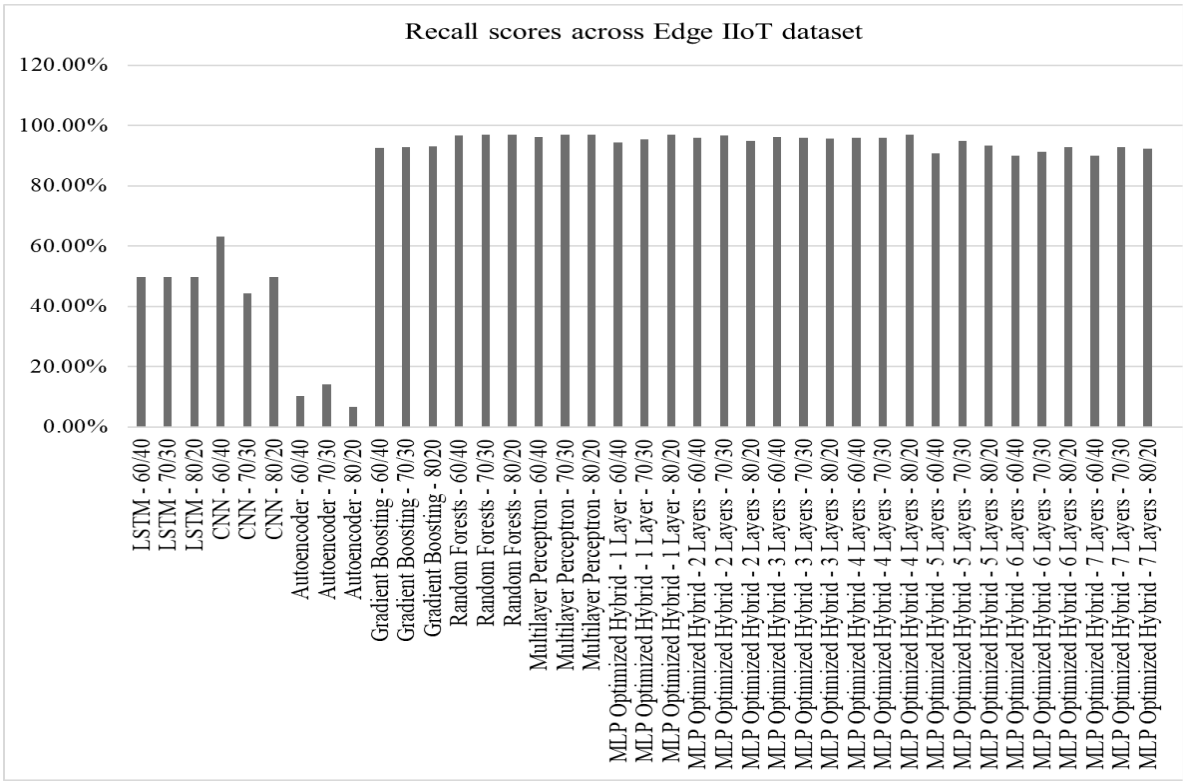


Figure 4.177: Recall scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on Edge IIoT dataset

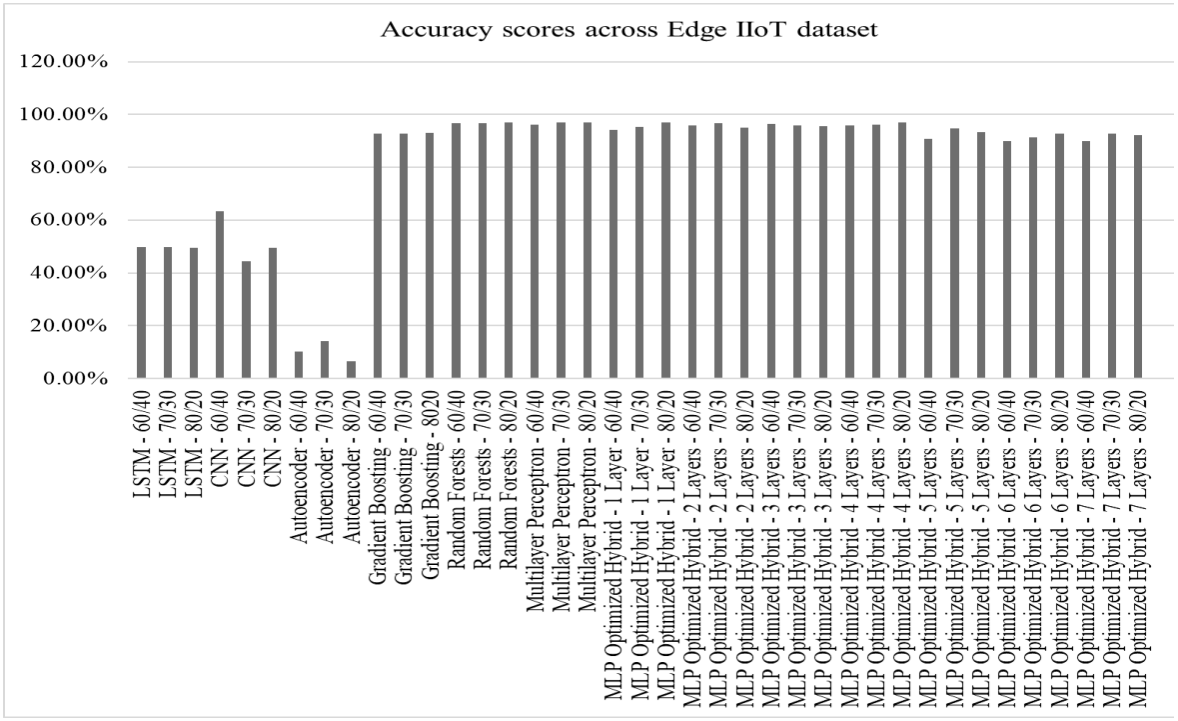


Figure 4.178: Accuracy scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model on Edge IIoT dataset

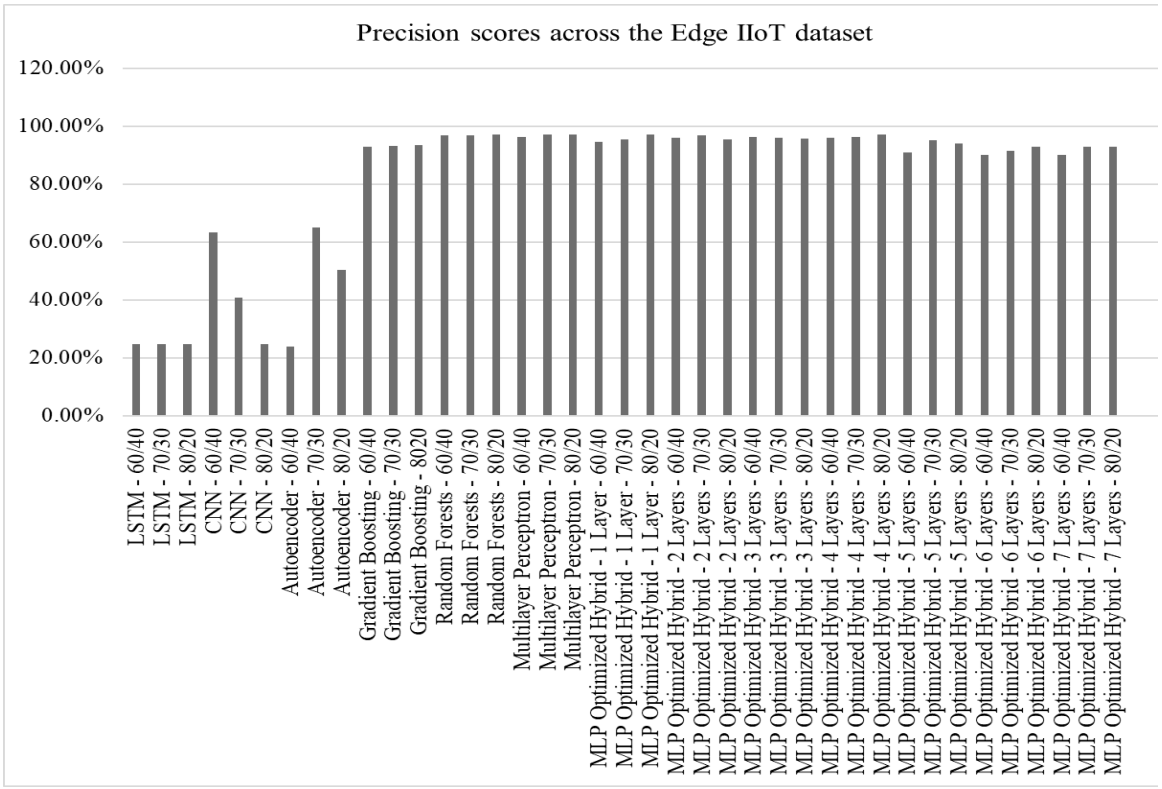


Figure 4.179: Precision scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model variants on Edge IIoT dataset

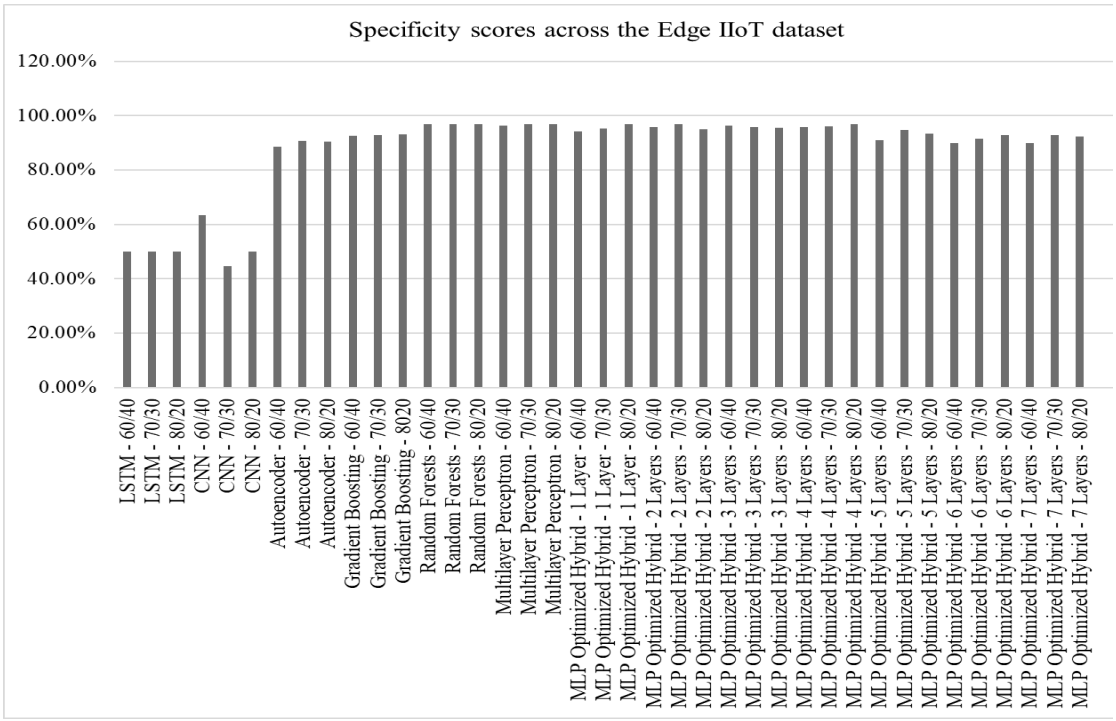


Figure 4.180: Specificity scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model variants on Edge IIoT dataset

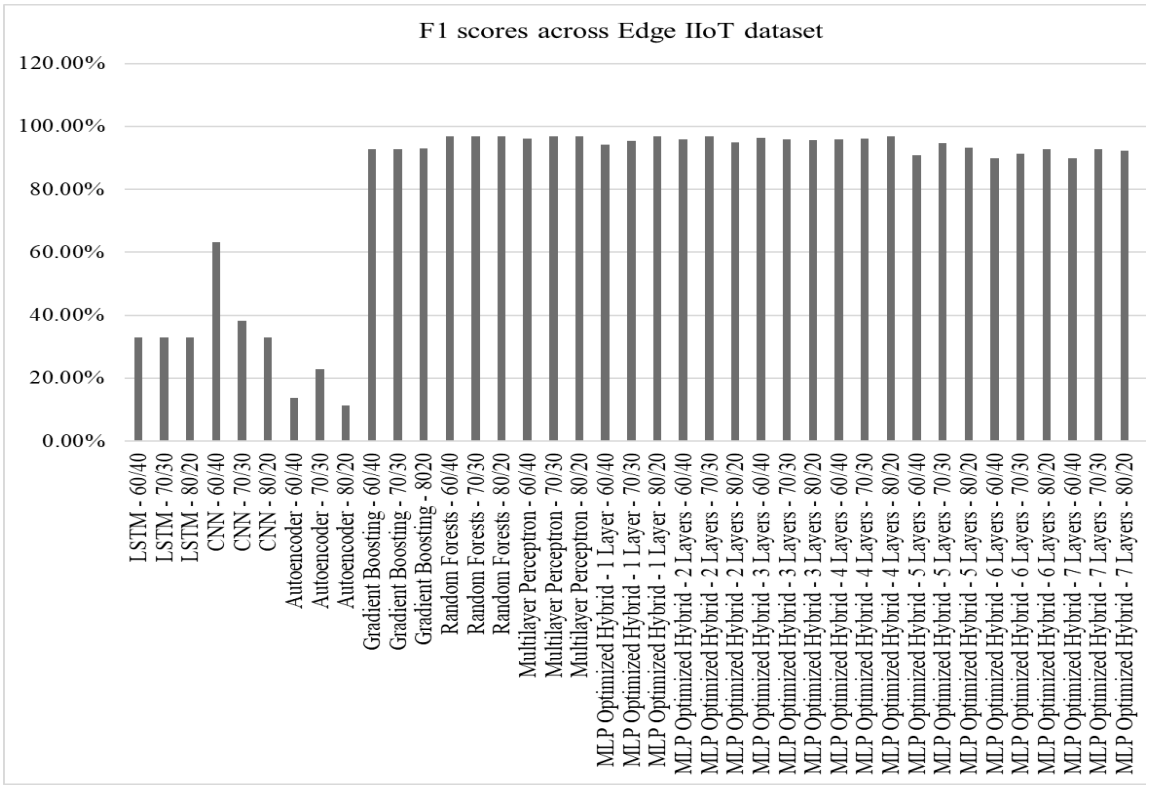


Figure 4.181: F1 scores for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model variants on Edge IIoT dataset

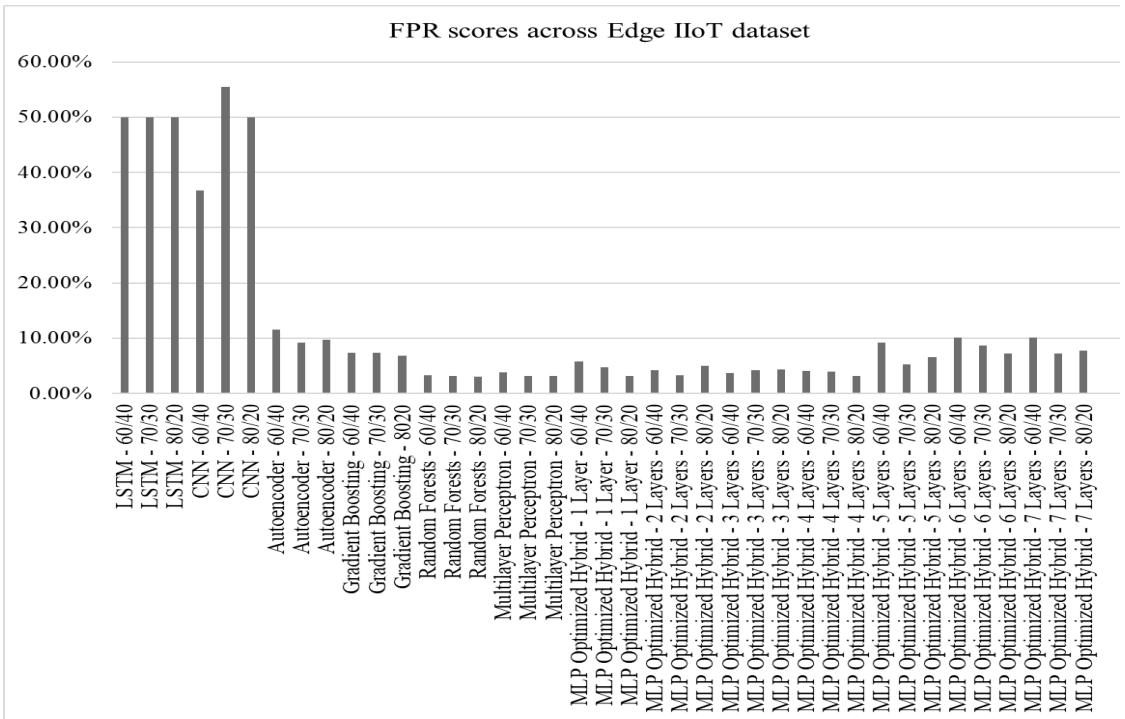


Figure 4.182: False positive rate for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model variants on Edge IIoT dataset

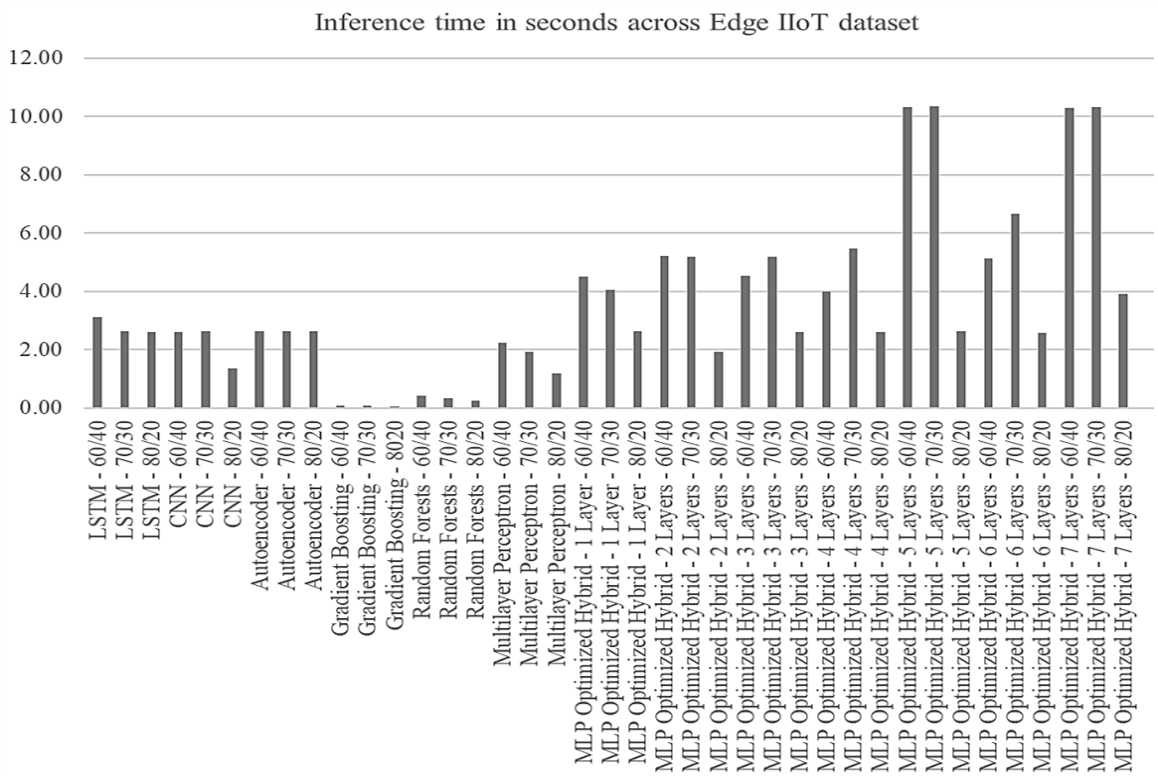


Figure 4.183: Inference time for LSTM, CNN, Autoencoder, Gradient Boosting, Random Forests, Multilayer Perceptron, and MLP-enhanced hybrid deep autoencoder model variants on Edge IIoT dataset

Five models emerge as the dominant variants for the Edge IIoT dataset, providing significant performance metrics. These models were the four-hidden-layers' 80/20, one hidden layer's 80/20, two-hidden-layers' 70/30, three hidden layers' 60/40, and four-hidden-layers' 70/30 variants. The comparative assessment of these models showed their key performance metrics, enabling the selection of the best model for implementation. Four-hidden-layers' 80/20 variant emerged as a top contender for all metrics. This model achieved the highest recall (96.87%), precision (96.94%), specificity (96.88%), accuracy (96.87%), and F1 score (96.87%). The model had a false positive rate of 3.12%, the highest file size of 20.55MB, and an inference time of 2.61 seconds, which was the lowest among the selected models. One hidden layer's 80/20 and two hidden layers' 70/30 had the lowest model sizes at 10.30MB. The 80/20 variant with four hidden layers had the lowest regression metrics, with mean absolute error and mean squared error of 0.03 and a root mean squared error of 0.18. These scores were slightly lower than those of one hidden layer's 80/20 variant, which reported MAE and MSE of 0.03 and RMSE of 0.18. These recall, precision, specificity, accuracy, F1, FPR, and regression metrics for the selected models are shown in Figures 4.184 and 4.185.

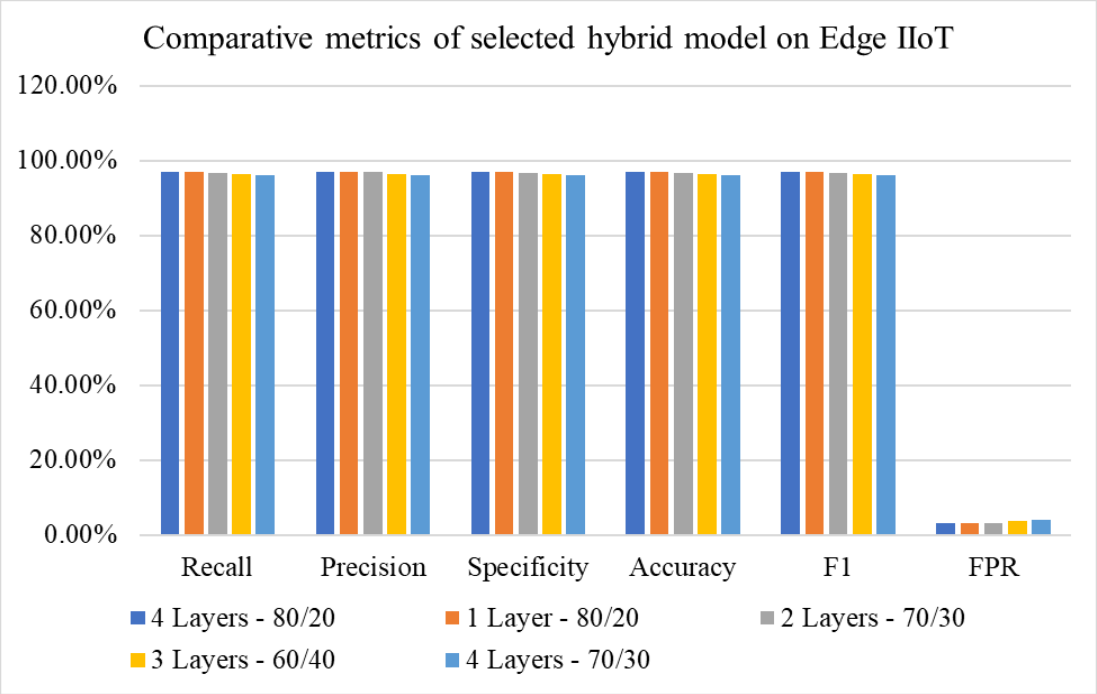


Figure 4.184: Comparative view of performance metrics for selected hybrid models on Edge IIoT dataset

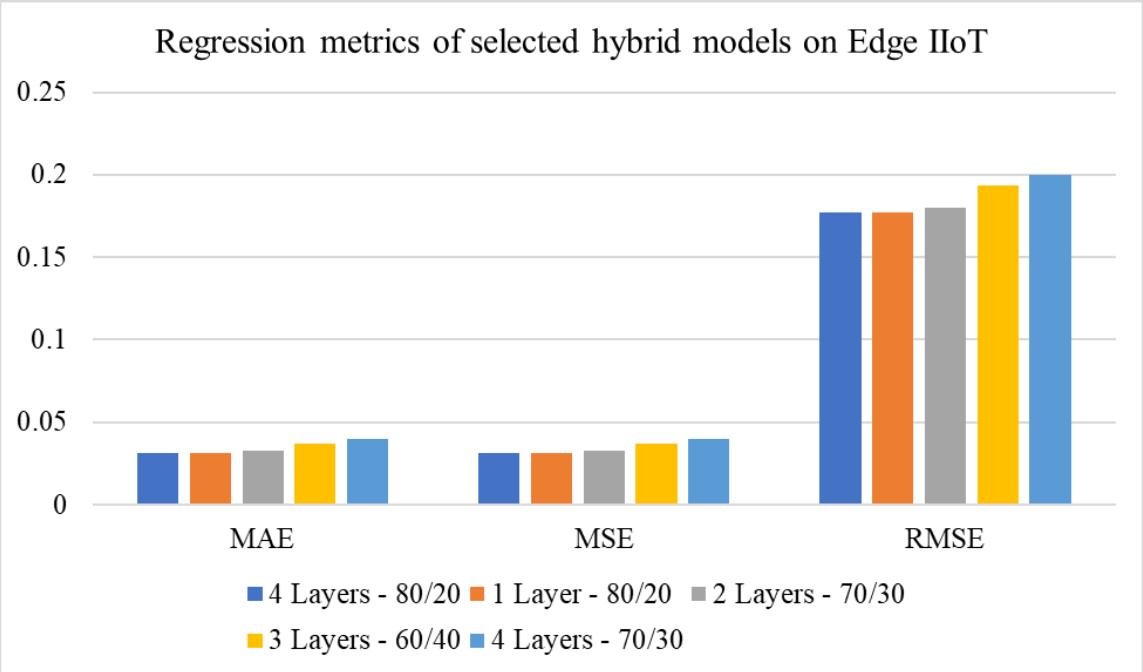


Figure 4.185: Comparative view of regression metrics for selected hybrid models on Edge IIoT dataset

Analysis of the confusion matrices and ROC curves for the selected models provided further insights into their performance. The analysis of the 80/20 confusion matrix for the four-hidden-layers noted 119 false positives and 507 false negatives. The false-positive cases account for 0.60% of the total classified incidents in the dataset. This

situation implies that only 0.6% of the total classification was misidentified as benign when they were malicious. This value was lower than the value reported in the best model for the WUSTL dataset (6.92%). This phenomenon implies that the model offered better false-positive detection concerning the overall classification than the Edge IIoT dataset. The model's ROC curve demonstrated a reliable classification performance, with an AUC of 0.97. This high AUC demonstrated the model's ability to generalize to the dataset, effectively improving its generalization capabilities. These results are shown in Figures 4.186 and 4.187.

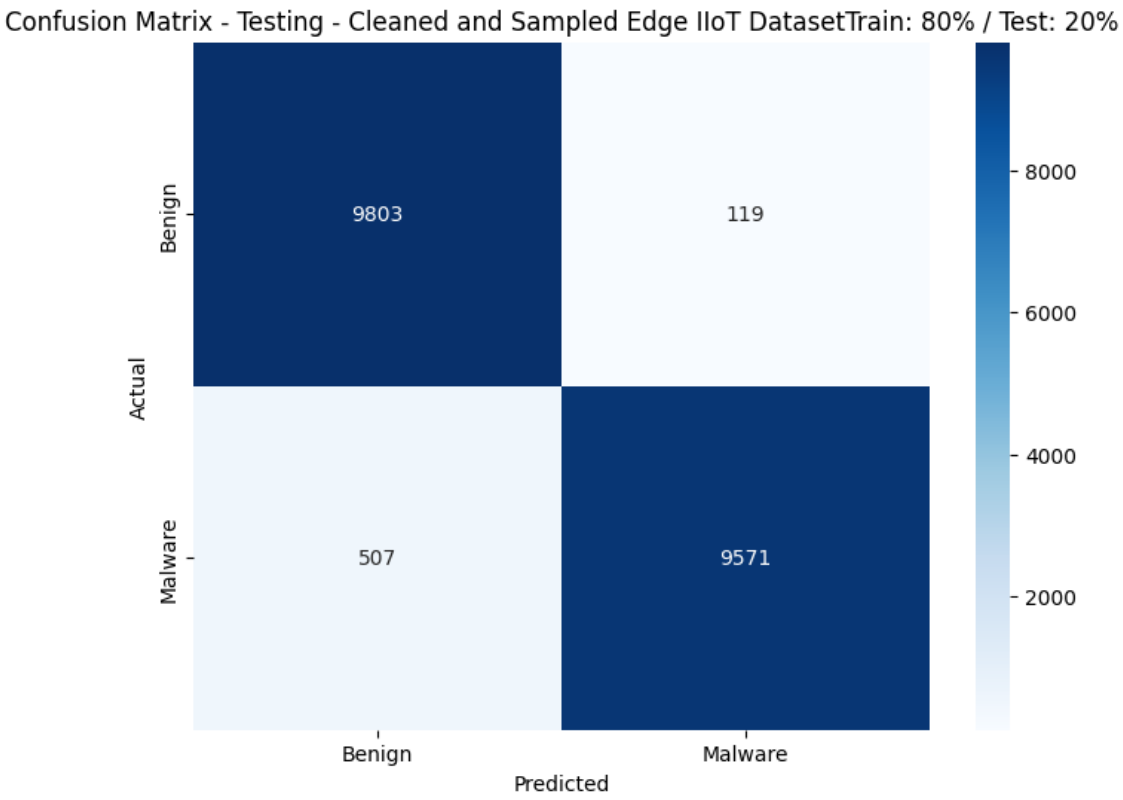


Figure 4.186: Confusion matrix for a four-hidden-layer 80/20 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

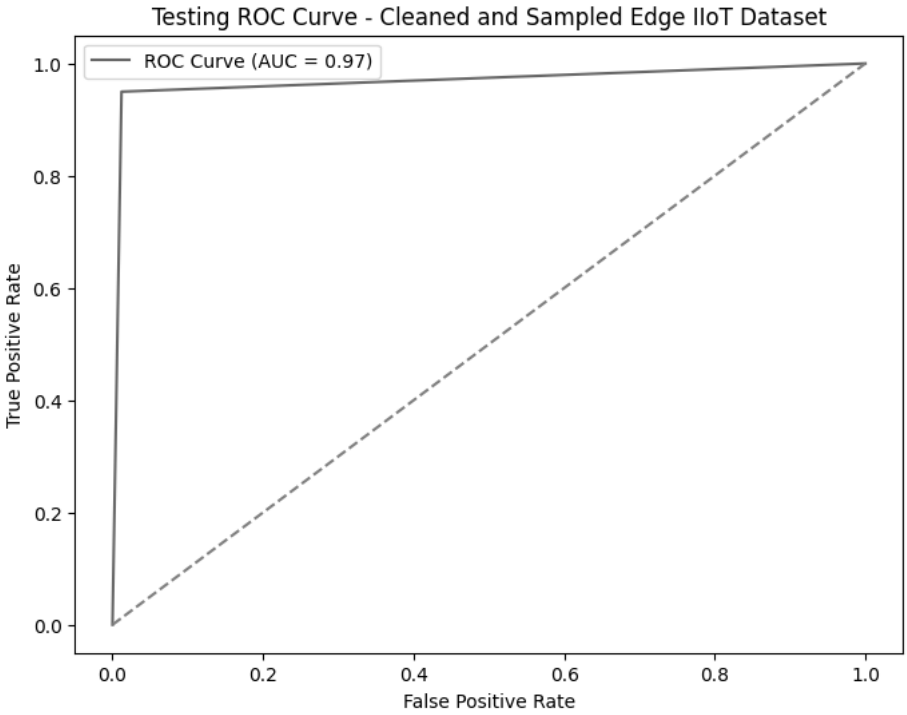


Figure 4.187: Receiver Operating Characteristics curve for four-hidden-layers' 80/20 MLP-Enhanced and Nadam-optimized model on Edge IIoT dataset

Analysis of the confusion matrix and ROC curve for the 80/20 variant with one hidden layer provided insights into the model's classification performance. The confusion matrix showed 125 false positives and 504 false negatives. The false-positive rate was 0.63%, a slight increase from the 80/20 model with four hidden layers. While the false-positive percentage for the model was higher than that of the 80/20 four-hidden-layers variant, it was still lower than that of the best model for the WUSTL dataset. The assessment of the model's ROC curve showed a score consistent with the 80/20 split of the four-hidden-layers. The model had an AUC of 0.97, implying that it predicted reasonably well on an unseen dataset. This performance indicated slight differences compared to the 80/20 variant with four hidden layers in terms of prediction capabilities. The confusion matrix and ROC curve for the model were presented in Figures 4.188 and 4.189.

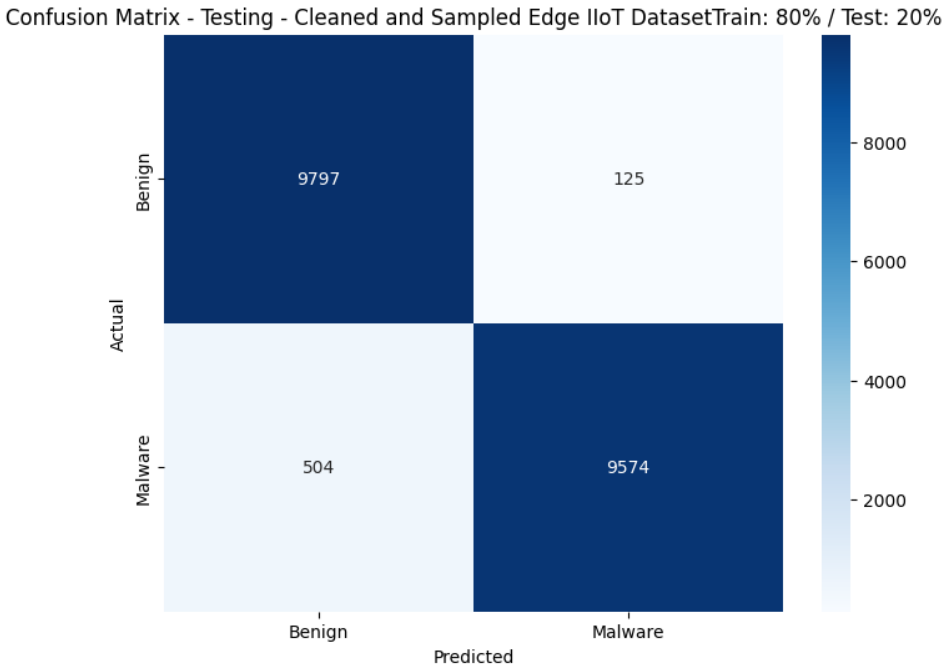


Figure 4.188: Confusion matrix for one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

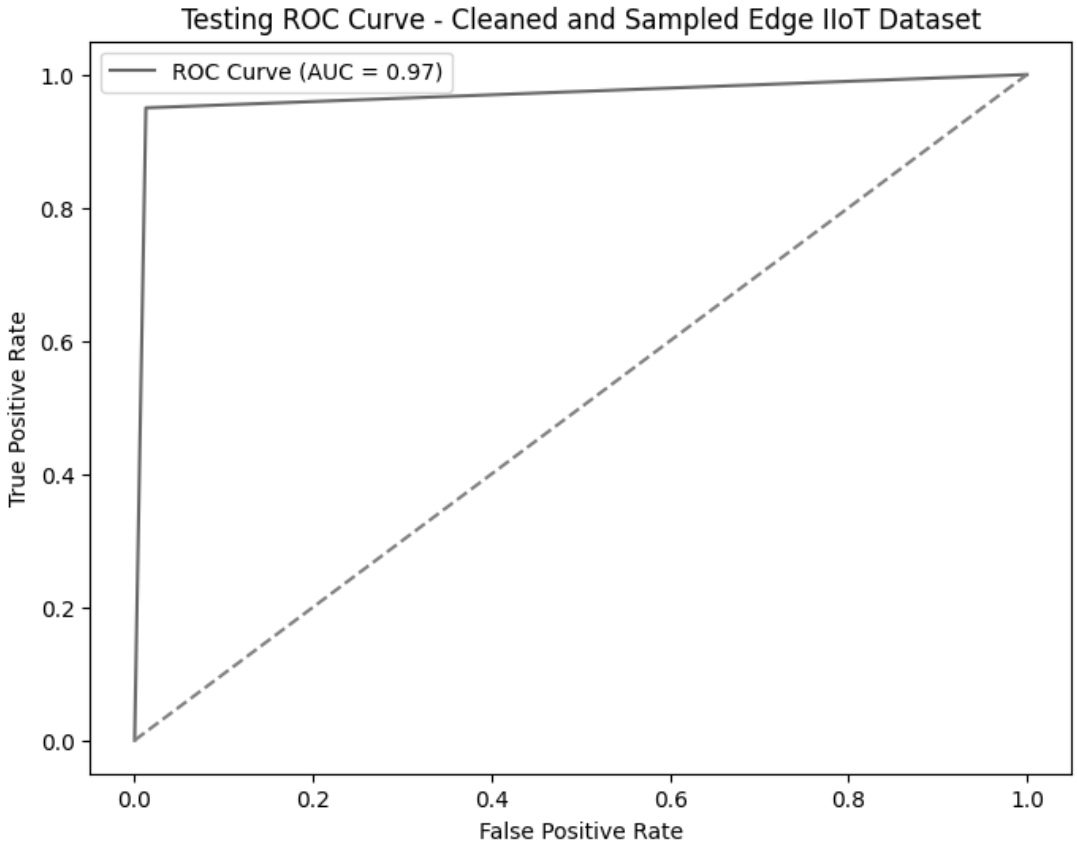


Figure 4.189: Receiver Operating Characteristics curve for a one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

The analysis of the confusion matrix and ROC curve for the two-hidden-layer 70/30 variant provided insights into the performance of a balanced training/testing split on the Edge IIoT dataset. For instance, the confusion matrix showed 178 false positives and 796 false negatives in the model. This phenomenon implies that false positives accounted for 0.59% of the total classification, which was lower than the 4.2% observed in the 80/20 variant with four hidden layers. This significant improvement in overall false-positive classification was attributed to the expanded use of the testing dataset and the relative reduction in the training subset compared to the 80/20 variants. However, the overall false classification rate for the model was 3.25%, which was higher than those for the 4-hidden-layer 80/20 and 1-hidden-layer 80/20 variants, which were 3.13% and 3.15%, respectively. This increase in overall false classification was attributed to the rise in false negative cases and the subsequent reduction in accurate classifications. The ROC curve assessment indicated that the model's AUC was 0.97, consistent with the other models. This phenomenon implies that the model reliably predicts on the dataset, with very minimal differences from the preceding models. These results are shown in Figures 4.190 and 4.191.

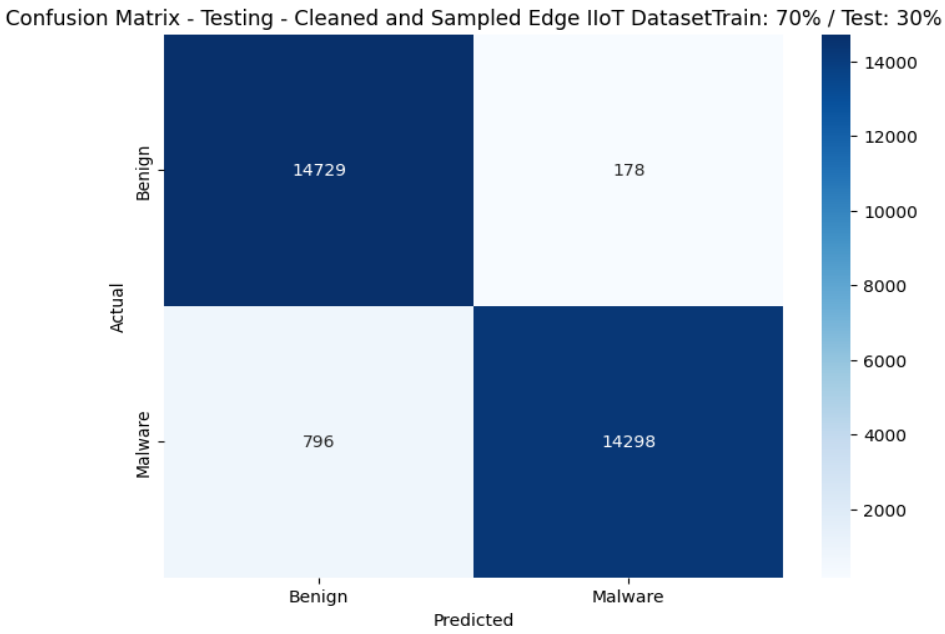


Figure 4.190: Confusion matrix for a two-hidden-layer' 70/30 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

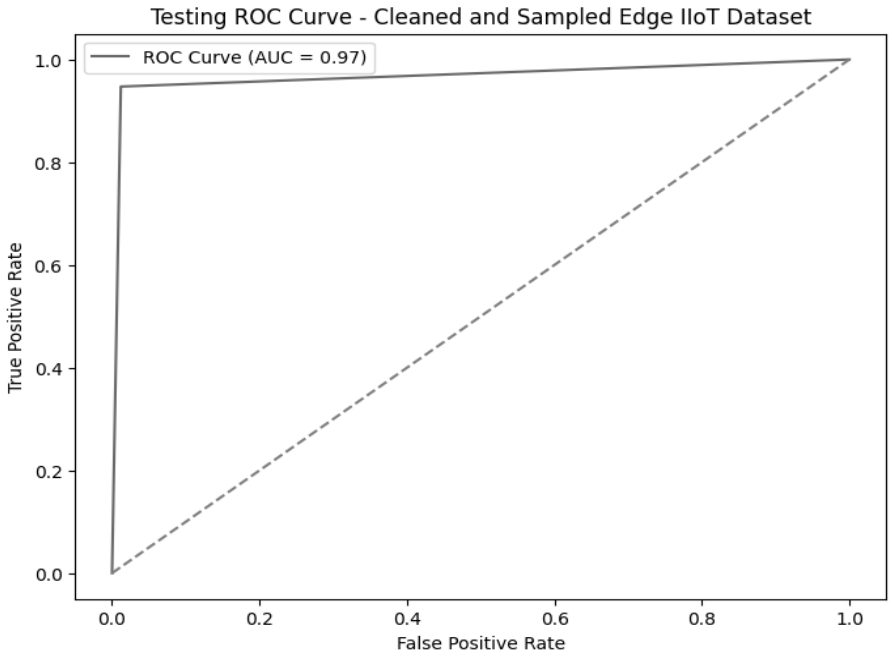


Figure 4.191: Receiver Operating Characteristics curve for a two-hidden-layer 70/30 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

Analysis of the confusion matrix and ROC curve for the 60/40 model with three hidden layers provided insights into its detection performance. The model's confusion matrix indicates 374 false positives and 1118 false negatives. This situation implies that the model's classification missed 0.94% of threats across the entire classification, indicating a substantial improvement over the performance of the preceding models. Further, there was an increase in false classification, accounting for 3.73%. Considering that this was the highest 60/40 ratio selected, it was evident that the ratio does not provide a suitable environment for model training and testing. The analysis of the ROC curve further supports this perception, as the curve yielded an AUC of 0.96. While this score provided reliable predictive performance on unseen datasets, it lagged behind those recorded by the one- and four-hidden-layer 80/0 variants and the two-hidden-layer 70/30 configuration. These results for the model are shown in Figures 4.192 and 4.193.

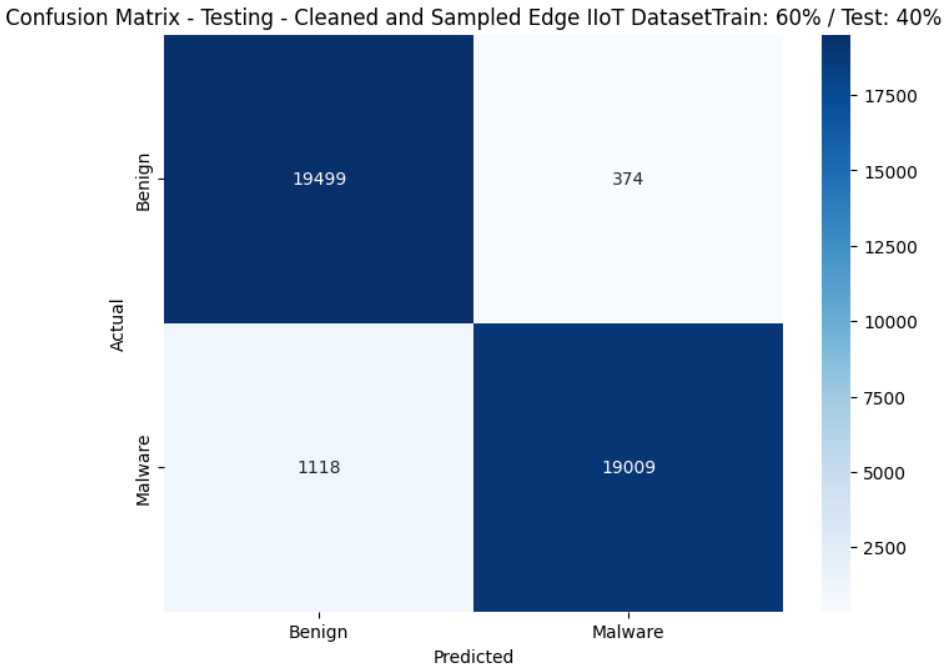


Figure 4.192: Confusion matrix for three hidden layers' 60/40 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

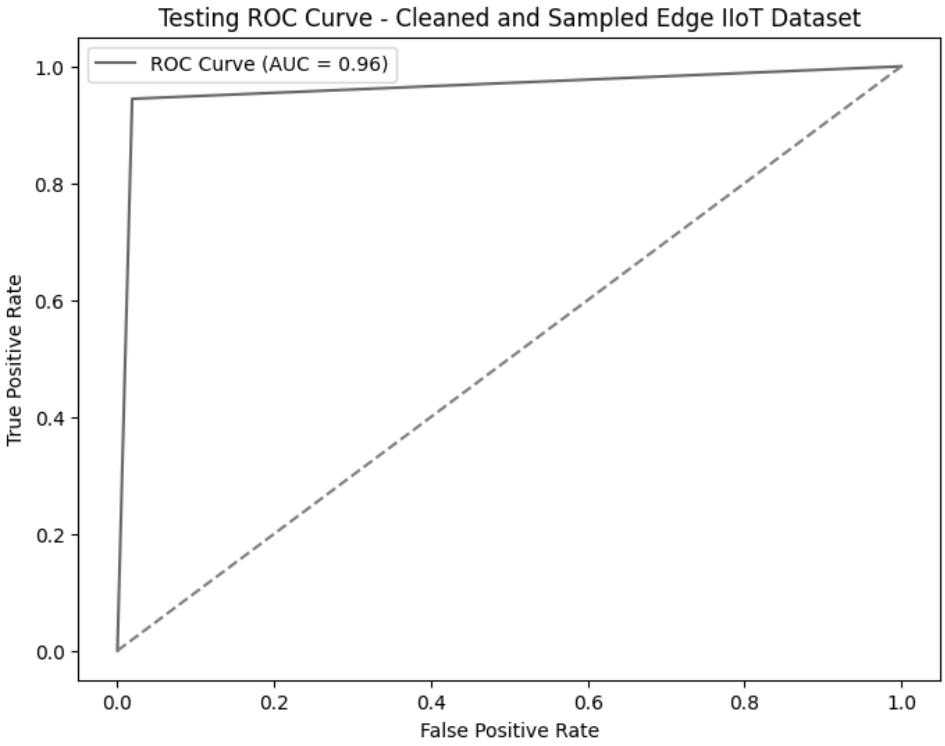


Figure 4.193: Receiver Operating Characteristics curve for a 3-hidden-layer 60/40 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

The review of the confusion matrix and ROC curve for the four-hidden-layer 70/30 model on the Edge IIoT dataset showed that performance improved as the number of

layers increased from two to four. The confusion matrix showed 220 false positives and 976 false negatives. This phenomenon implies that the number of threats allowed by the model increased from 0.59% to 0.73%, while the overall misclassification rate increased from 3.25% to 3.99% with the addition of two hidden layers. Based on these results, it was evident that increasing the number of hidden layers from 2 to 4 at a 70/30 training/testing ratio affected the model's performance. The assessment of the model's ROC curve showed a slight decline as the number of hidden layers increased. The model's AUC score was 0.96, which was lower than the 0.97 that had been recorded in a two-hidden-layer 70/30 configuration. Based on these scores, the four-layer and one-layer 80/20 variants, and two two-layer 70/30 models emerged as the best performers in the dataset. The results for the four-hidden-layers' 70/30 variant are shown in Figures 4.194 and 4.195.

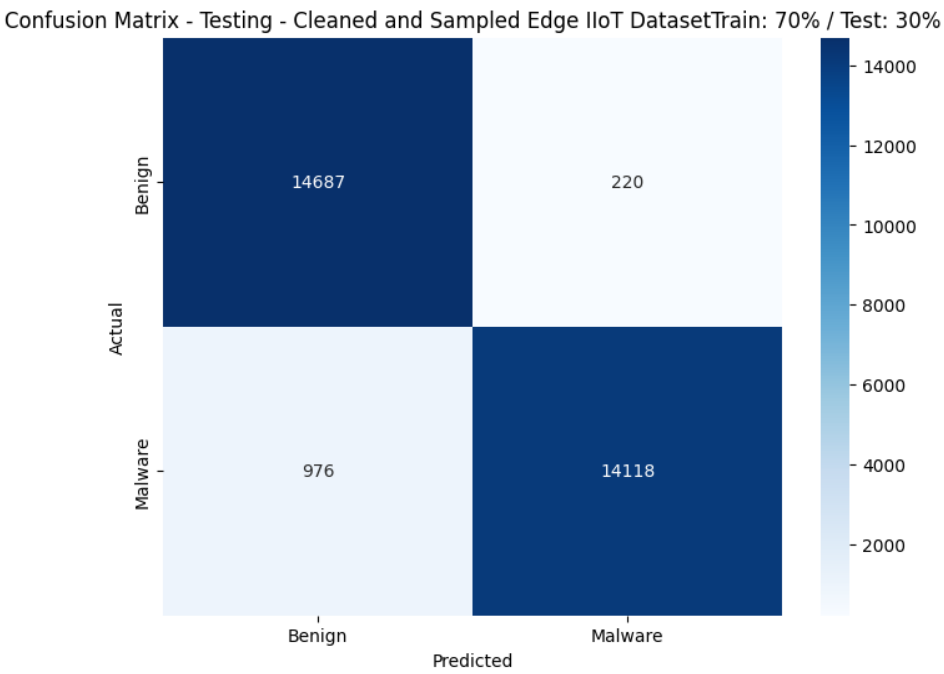


Figure 4.194: Confusion matrix for four-hidden-layers' 70/30 MLP-Enhanced and Nadam-optimized model on Edge IIoT dataset

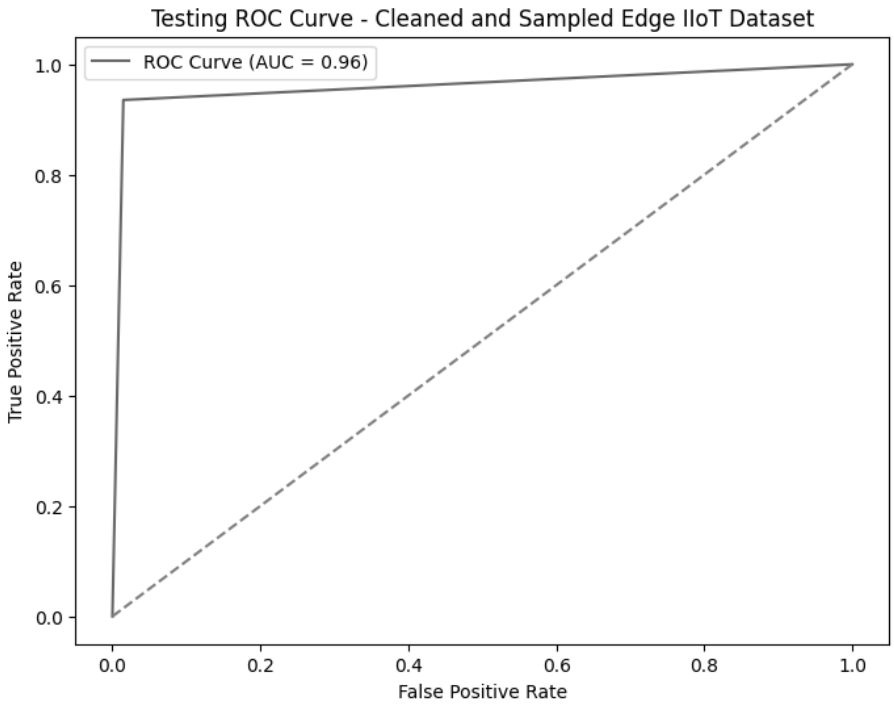


Figure 4.195: Receiver Operating Characteristics curve for four-hidden-layers' 70/30 MLP-Enhanced and Nadam-optimized model on Edge IIoT dataset

The review of the loss curve for the four-hidden-layer' 80/20 model provided insights into performance. For instance, the model's test loss started at zero and increased slightly to about 0.05 in the fourth epoch. It then slightly decreased to about 0.025 in the sixth epoch, before rising to 0.5 in the seventh epoch. It maintained this score in the eighth epoch, then fell to about 0.25 in the ninth, and stabilized at this level until the end of the testing session. The sudden rise in the seventh epoch was attributed to the adaptive learning rate that was modified after the sixth epoch in the model. On the other hand, the validation loss started at 0.7 and declined to 0.35 in the third epoch, then wobbled until the twelfth epoch. In the thirteenth epoch, it rises to 0.4, then returns to 0.35, and wobbles until the end of the testing session. This phenomenon suggests that with additional training epochs, the model may stabilize and improve its predictive performance. This behavior was presented in the loss curve in Figure 4.196.

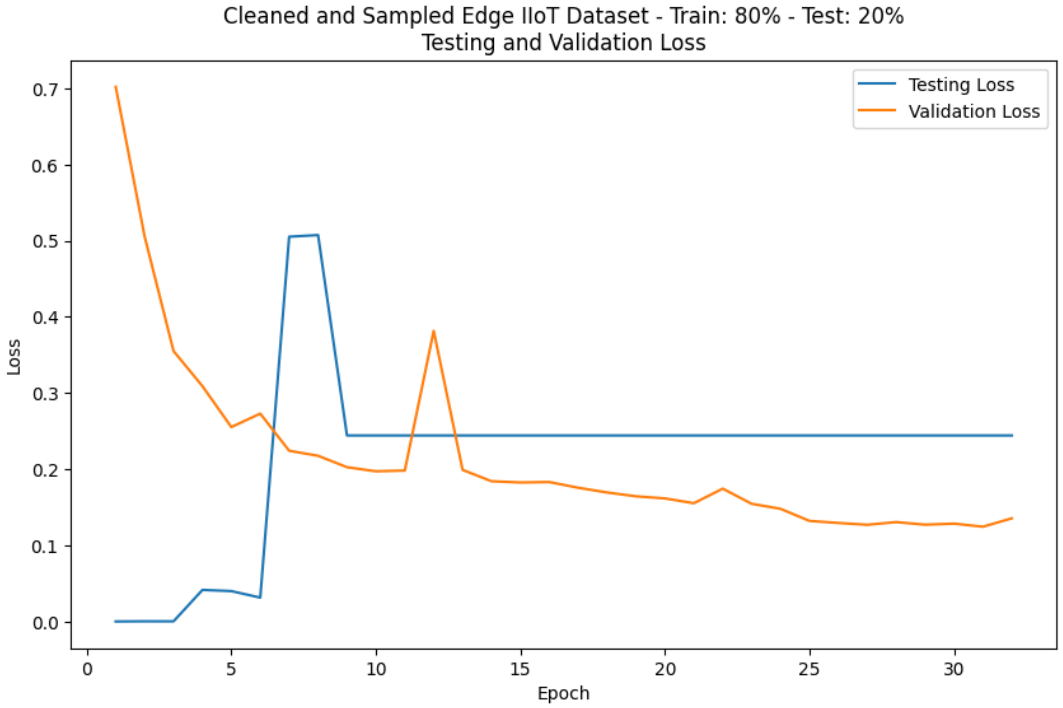


Figure 4.196: Testing and validation loss curves for a four-hidden-layer 80/20 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

The analysis of the loss curve for the 80/20 variant of one hidden layer provided insights into its detection performance, enhancing understanding of its scores. The loss curve for the model started at zero and increased to about 0.5 in the fourth epoch, then fell to 0.5 in the sixth epoch. It then rose to 0.5 in the seventh epoch, fell to 0.2 in the eighth epoch, and returned to 0.5 in the ninth epoch. It maintained this level until the end of the testing session, demonstrating stability of the testing loss. The sudden changes in the seventh epoch were attributed to the dynamic learning rate, which was adjusted after the model failed to improve. The validation loss started high at 0.525 and steeply declined to 0.2 in the seventh epoch. It then wobbled slightly throughout the testing session until the end. This phenomenon suggests that, with additional epochs, the model may stabilize in performance. However, the model's test and validation loss scores were slightly lower than those reported for the four-hidden-layers 80/20 variant. These results are shown in Figure 4.197.

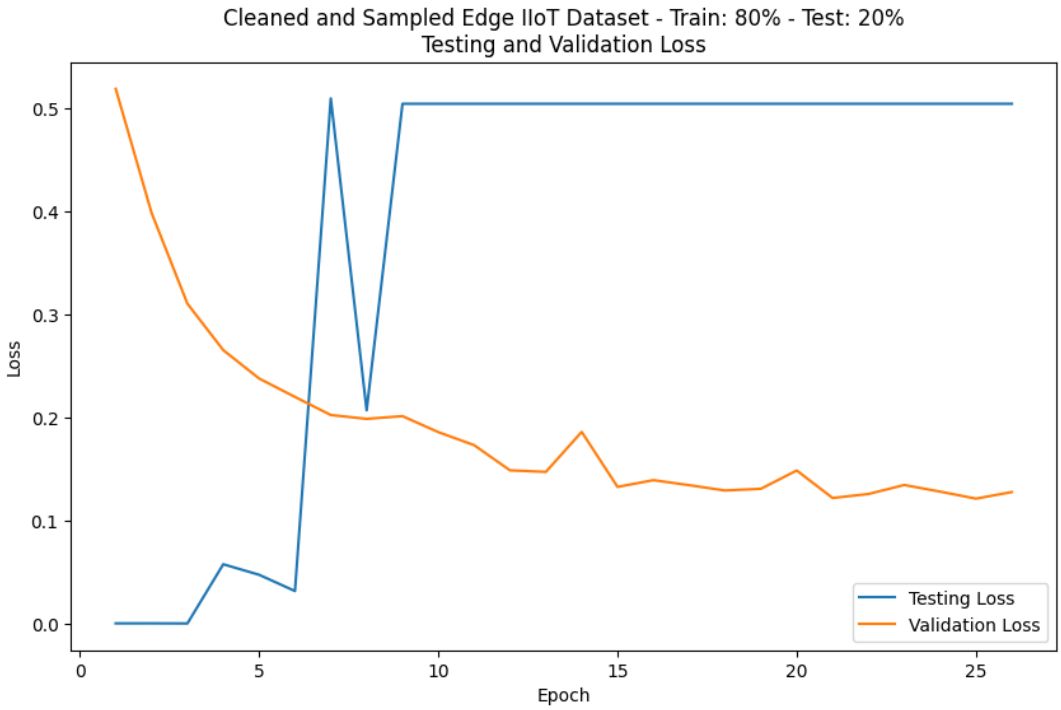


Figure 4.197: Testing and validation loss curves for one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on Edge IIoT dataset

The review of the loss curve for the two-hidden-layer' 70/30 provided information on its testing behavior and helped understand its scores in the confusion matrix. The model's loss curve started at zero and increased slightly to 0.05 in the fifth epoch, then rose to 0.5 in the seventh epoch due to a learning rate adjustment. It maintained this score in the eighth epoch, but fell to 0.25 in the ninth and maintained it until the end of the testing session. The validation loss started high at about 0.58 and declined steadily to 0.2 in the thirteenth epoch, and wobbled until the end of the testing session. The testing and validation loss for this model were in the same range as those of the four-hidden-layers' 80/20 towards the end of the testing session. Nevertheless, the 80/20 variant with four hidden layers outperforms the model in the confusion matrix scores. The results for the two-hidden-layer' 70/30 variant are shown in Figure 4.198.

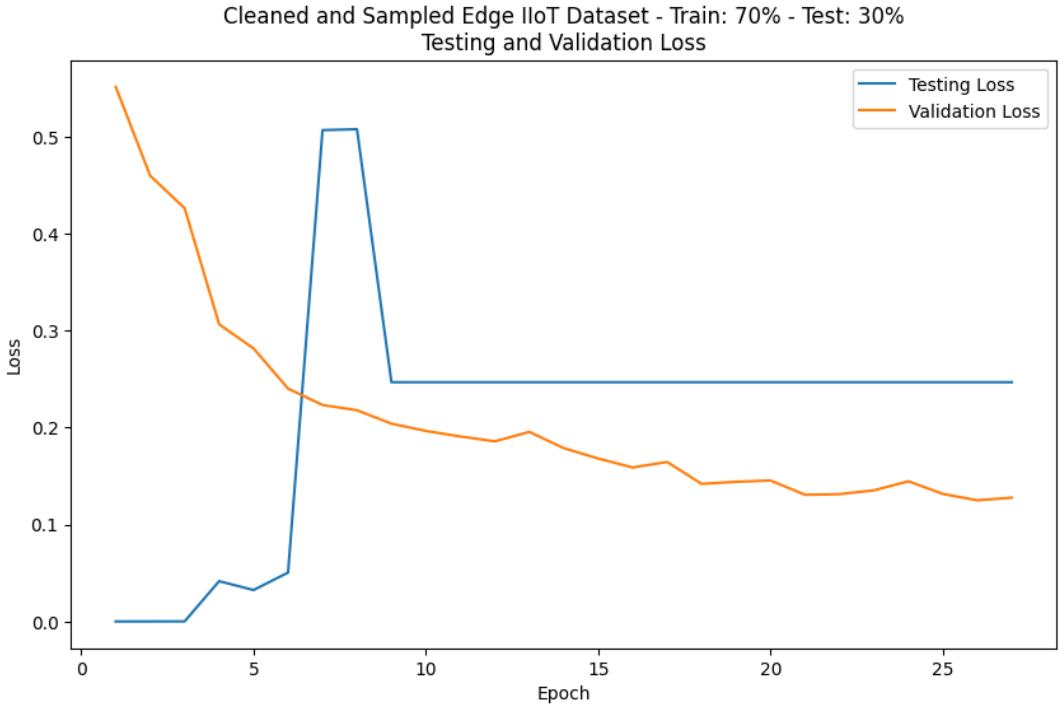


Figure 4.198: Testing and validation loss curves for a two-hidden-layer 70/30 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

The analysis of the loss curve for the 60/40 variant of the three hidden layers provided insights into the model's performance. The testing loss started at zero and increased slightly to 0.05 in the fourth epoch, then rose to 0.5 in the seventh epoch. This sudden rise was attributed to an adjustment in the learning rate, which led to a new loss score for the model. The model wobbled slightly before settling at 0.5 in the tenth epoch. It maintained this score until the end of the testing session. This score was higher than that experienced by the four-hidden-layers' 80/20 variant, reducing its potential adoption. The validation loss for the model started at about 0.65 and steadily decreased to 0.35 by the fifth epoch. It then wobbled until the end of the training session, where it ended with 0.2 in the 25th epoch. This phenomenon implies that with additional training and testing epochs, the model may improve its performance. These results for the model are shown in Figure 4.199.

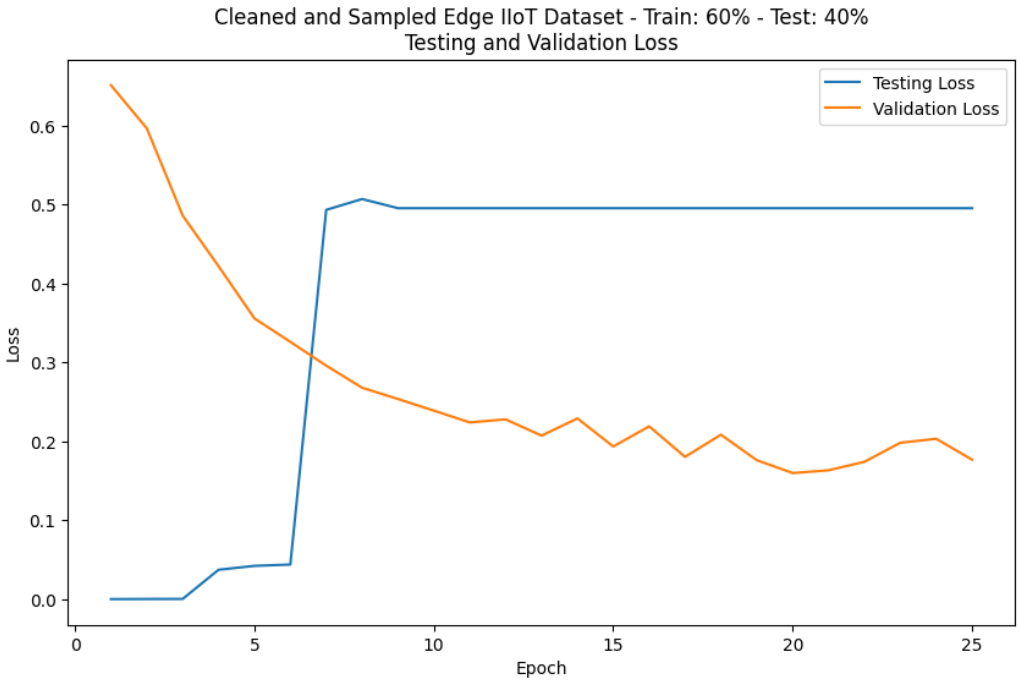


Figure 4.199: Testing and validation loss curves for a three-hidden-layer 60/40 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

The analysis of the loss curve for the 70/30 four-hidden-layer model provided insights into its detection performance. The testing loss for the model started at zero and increased slightly to 0.05 in the third epoch, then rose to 0.5 in the seventh epoch. This rise in the seventh epoch was attributed to adjustments to the learning rates after the sixth epoch. The model then stabilized briefly and fell to about 0.25 in the ninth epoch. It maintained this score until the end of the testing session. This final score for testing loss was higher than that recorded in the four-hidden-layers' 80/20 variant but lower than that recorded in one hidden layer's 80/20 and three hidden layers' 60/40 variants. The validation loss curve for the model started at about 0.75 before falling to 0.425 in the third epoch. This score further declined to 0.3 in the fifth score, and wobbled to about 0.2 towards the end of the testing session in the twentieth epoch. This phenomenon implies that the model may improve its performance with additional training and testing epochs. Based on these loss curves, the recommended model for the Edge IIoT dataset was the 80/20 variant of the fourth hidden layer, which yielded the best confusion matrix scores and a better loss curve. The loss curve for the four-hidden-layers' 70/30 model is shown in Figure 4.200.

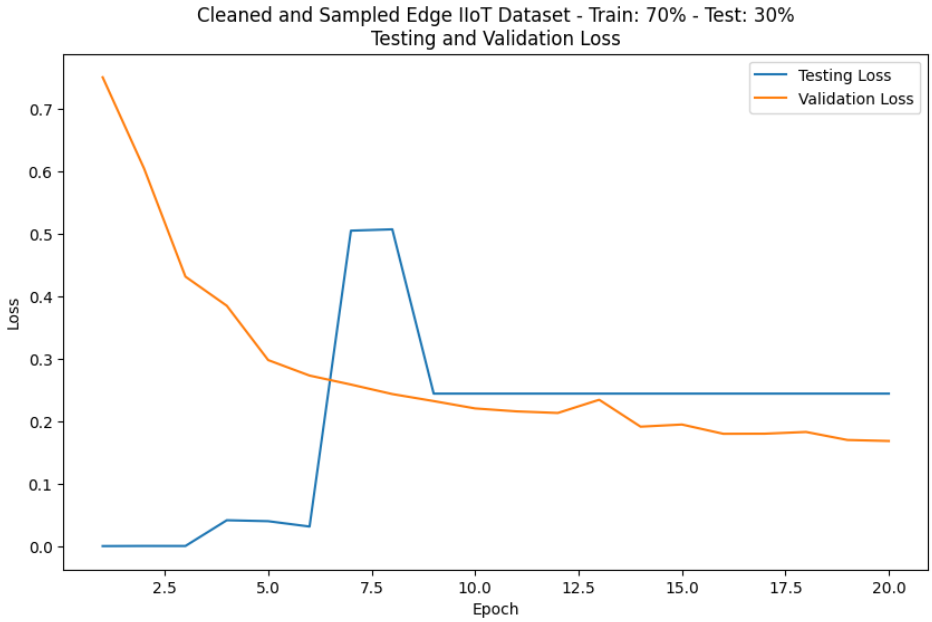


Figure 4.200: Testing and validation loss curves for a four-hidden-layer 70/30 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

The analysis of loss curves for the selected models offers insights into the model's testing performance. The training loss curve for the four-hidden-layers' 80/20 model started at about 0.88 and wobbled downwards to 0.2, while the validation loss began at 0.7 and wobbled downwards, with a slight increase in the ninth epoch before wobbled down again. The loss curve for one hidden layer's 80/20 variant demonstrated similar behavior, but started at 0.68 and ended at about 0.15. The validation loss curve for this model began at 0.52 and wobbled downwards, reaching 0.13 in the 26th epoch. The training loss curve for the two-hidden-layer 70/30 model started at 0.69 and wobbled downwards to 0.15, while the validation loss curve began at 0.55 and wobbled to 0.12 in the 26th epoch. This behavior was also evident in the 60/40 and 70/30 models with three hidden layers and four hidden layers, respectively. This situation implies that these models would have slightly improved with additional training epochs, as they would have helped them stabilize. Such stabilization would have led to higher model accuracy, potentially similar to that in the ICU dataset. These results are shown in Figures 4.201 to 4.205.

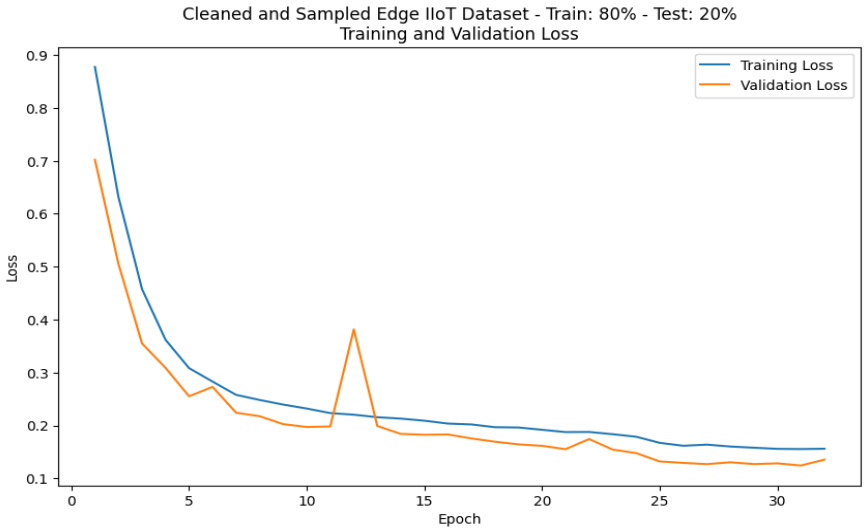


Figure 4.201: Training and validation loss curves for a four-hidden-layer 80/20 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

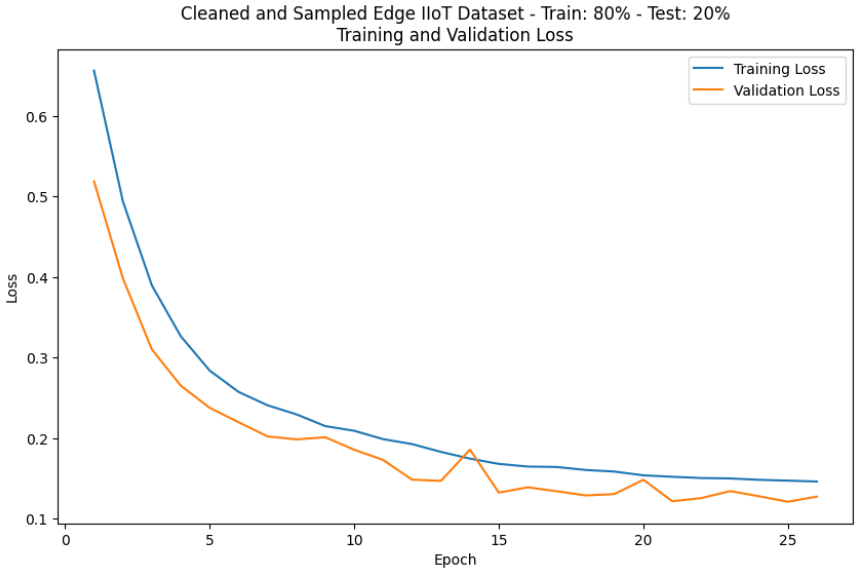


Figure 4.202: Training and validation loss curves for one hidden layer's 80/20 MLP-Enhanced and Nadam-optimized model on Edge IIoT dataset

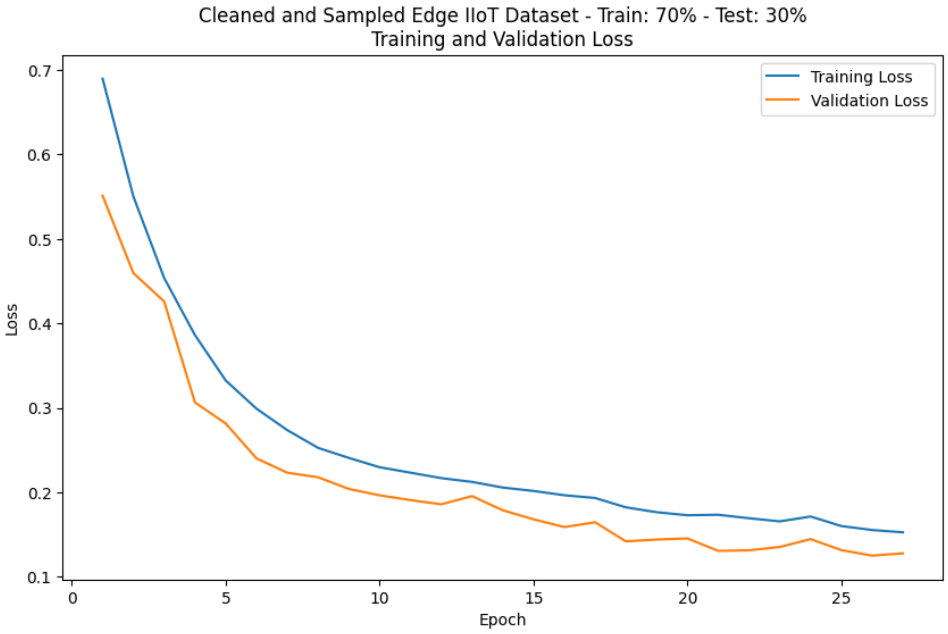


Figure 4.203: Training and validation loss curves for a hidden layer's 70/30 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

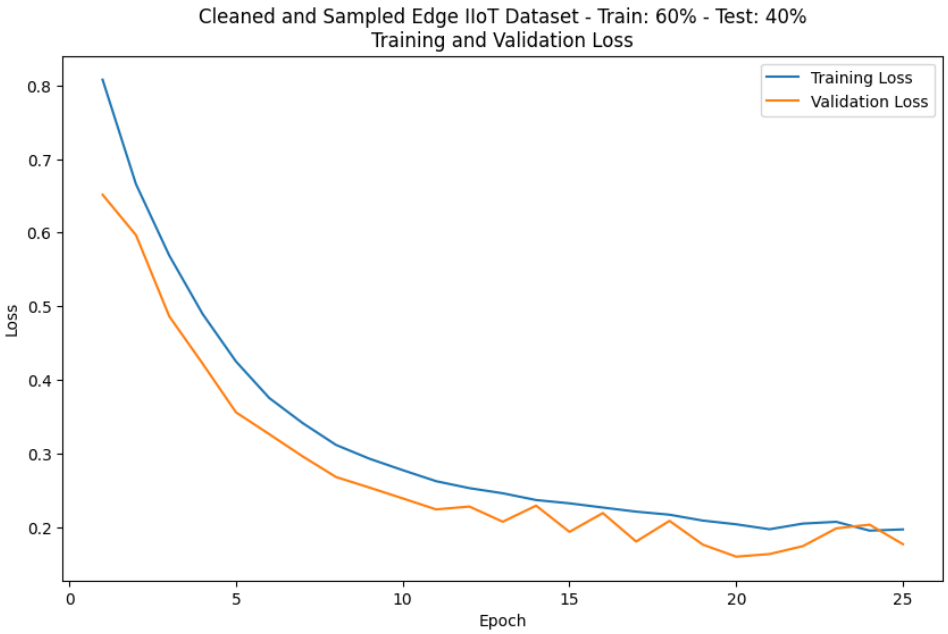


Figure 4.204: Training and validation loss curves for three hidden layers' 60/40 MLP-Enhanced and Nadam-optimized model on Edge IIoT dataset

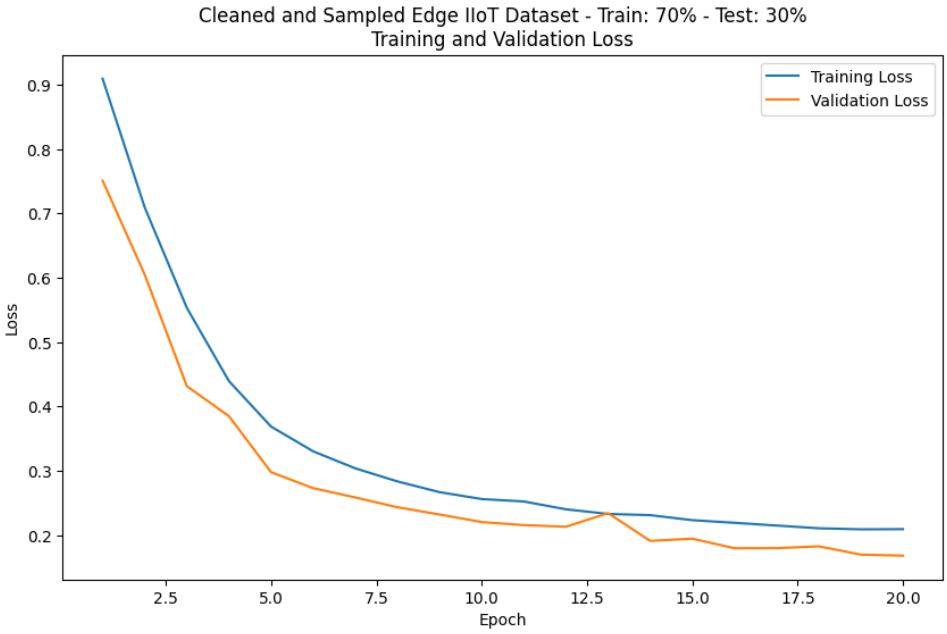


Figure 4.205: Training and validation loss curves for a four-hidden-layer 70/30 MLP-Enhanced and Nadam-optimized model on the Edge IIoT dataset

4.7 Web-Based User Interface

The hybrid model was used to test the entire ICU dataset as a sample to visualize the performance of the developed model. This approach was intended to simulate the performance of model deployment in healthcare information infrastructure. The simulation was done in three steps. The first step was running the trained model on the entire ICU dataset and recording the evaluation results in the updated comma-separated values (CSV) file for the dataset. The updated CSV file was then downloaded, along with the model's metrics. These resources were then used to create a database that stored information about the model and its performance. The second step involved configuring the email and short messaging service (SMS) to test the notification system. A personal Gmail account was configured, and a trial version of the SMS Mobile API was set to support the functionalities. The last step was to create a Flask application to simulate notifications in a real-life scenario. The application was then initialized, sending messages via email and SMS for attacks exceeding 10. The success of this simulation demonstrates the viability of deploying the model to monitor IMD traffic and promptly notify administrators regarding threats targeting the infrastructure. The visualization of the application, email, and SMS notifications, and the SMS Mobile API log from the simulation are shown in Figures 4.206 to 4.209.

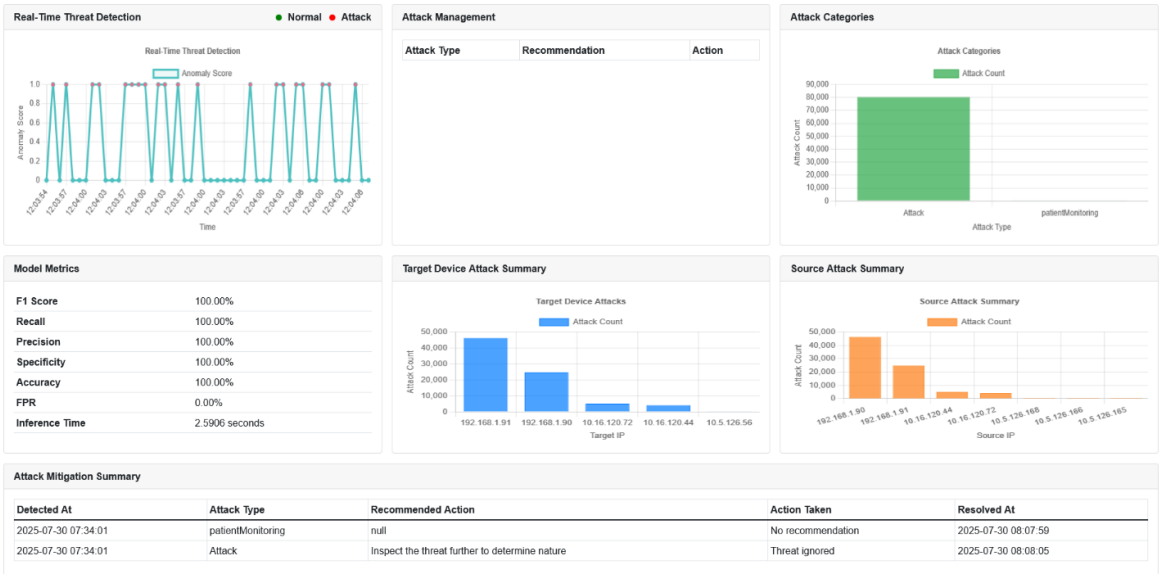


Figure 4.206: IDS user interface

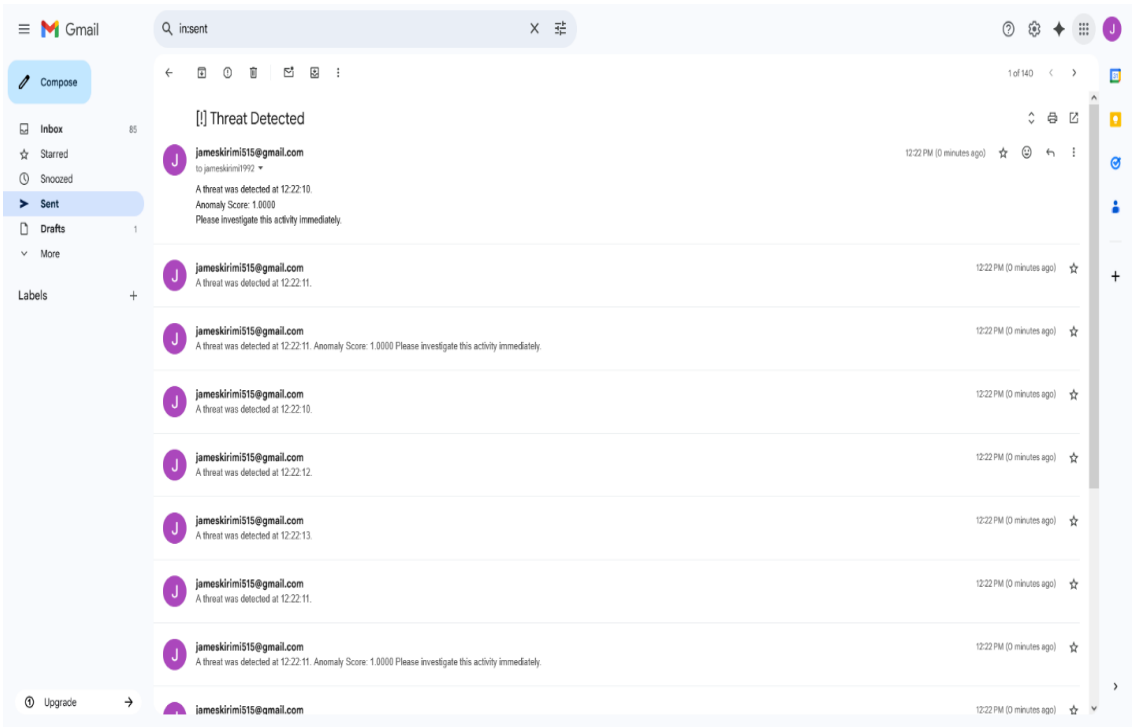


Figure 4.207: Email notification sent to administrator

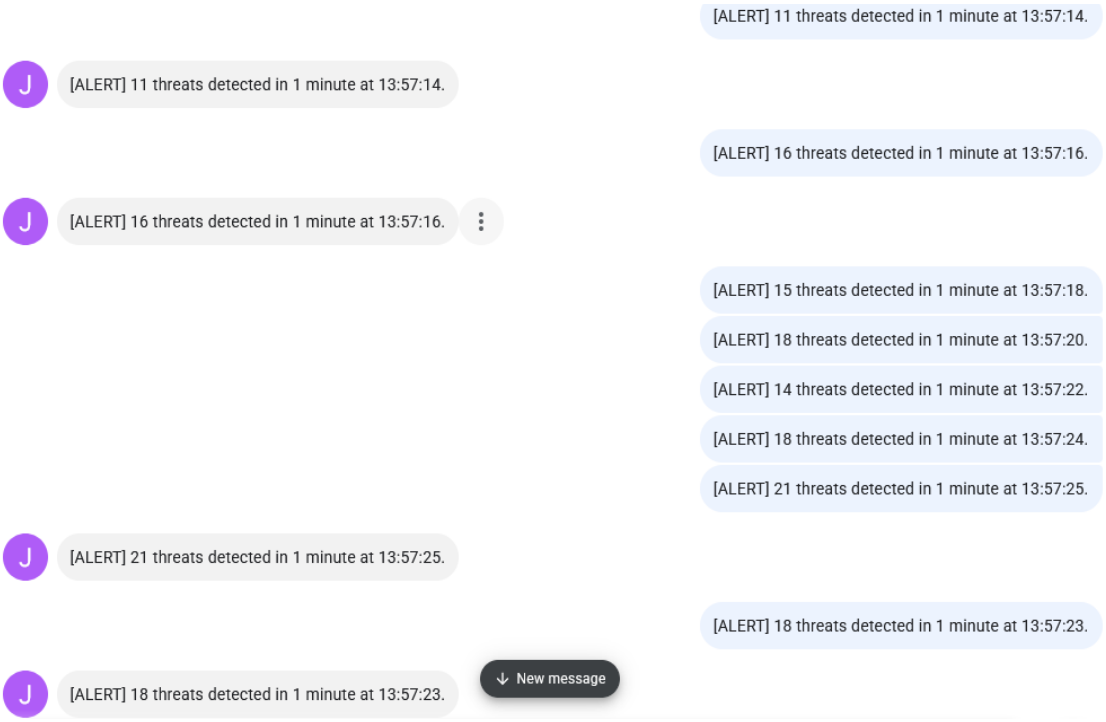


Figure 4.208: Message notification sent to administrator

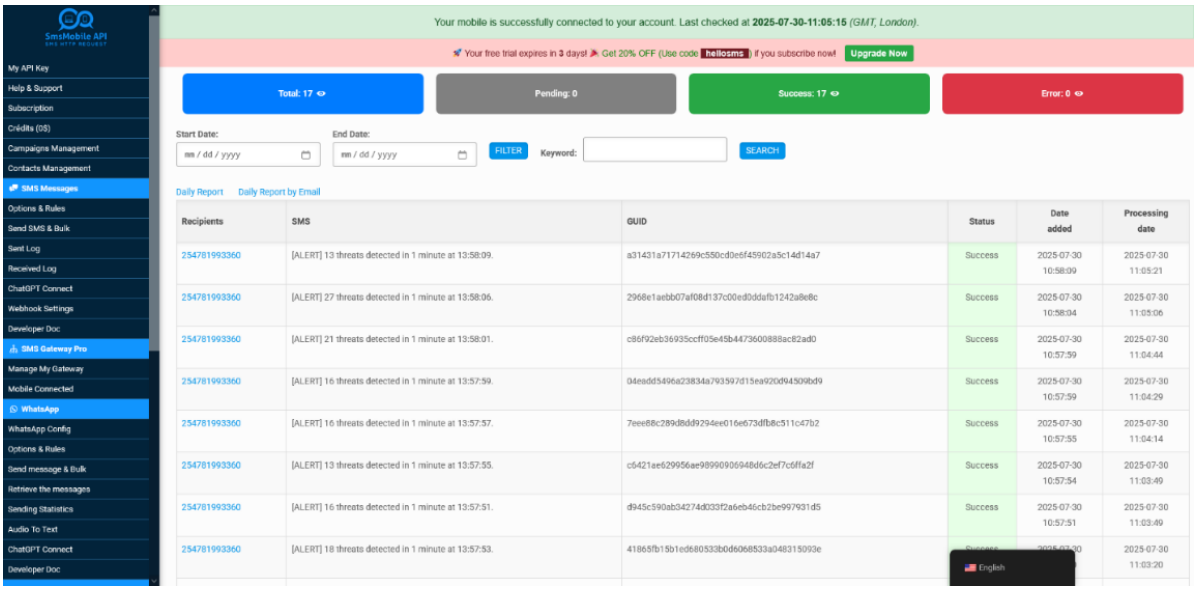


Figure 4.209: View of sent messages from the SMS gateway

CHAPTER FIVE
DISCUSSION

5.1 Model Design and Training

The design approach offered a robust framework for classifying threats in the hybrid deep autoencoder model. For instance, the encoder, through convolutional neural network layers, captured localized, flow-level patterns in the dataset, enabling the model to identify spatial dependencies that determined classification. The dependencies learned through this approach were refined by the decoder's reconstruction errors, thereby enhancing the model's performance. This feature enabled the model to understand the dataset's salient structure for anomaly detection. On the other hand, the bottleneck layer, using long short-term memory, captured the latent vectors in the dataset. This approach enabled the model to identify temporal dependencies across the data flow that determined distributed attacks. This design consideration enabled the developed model to identify stealthy, time-distributed attacks that might otherwise go undetected by conventional detection strategies. The deep autoencoder design adopted in the model enhanced compression, enabling it to leverage deep learning capabilities with reasonable deployment requirements.

The modular design strategy for the hybrid deep autoencoder model enabled scalability and facilitated the integration of new features and functionalities. For instance, the basic design used a single hidden layer for both the encoder and decoder. Integrating Nadam optimization into a single hidden layer did not significantly alter the architecture and maintained the model's size. However, incorporating a Multilayer Perceptron significantly affected the overall model architecture, resulting in a much smaller model.

The adaptive learning strategy used in the model played a crucial role in improving its performance. For instance, the early stopping feature enabled the model to monitor validation accuracy over five consecutive epochs and assess whether it changed. If the value did not change, the model triggered a change in the learning rate. Additionally, if the learning rate did not improve after five consecutive iterations, the model adjusted it to improve performance. The outcomes of this phenomenon were changes in the learning rates during model training, which improved the model's overall accuracy. Additionally, the model was trained for 50 epochs. However, the actual number of

epochs used depended on the specific model variant, as the validation accuracy and learning rate performance were monitored. For instance, if the model failed to improve after five consecutive epochs, the training stopped, and the model was evaluated. This design approach reduced overall resource use in Google Colab while ensuring optimal model performance.

5.2 Development of a Hybrid Deep Autoencoder Model

The four-layered hybrid deep autoencoder model, which comprised a convolutional neural network-based encoder and decoder, a long short-term memory-based bottleneck, and a multilayer perceptron-based classification head, provided a robust threat-detection environment for cloud-based Implantable Medical Devices. The model achieved 100% accuracy across recall, precision, specificity, and F1, and a 0.00% false-positive rate in the ICU dataset. These scores outperformed those from CNN, LSTM, and autoencoder models, demonstrating the model's efficiency at identifying threats to the IMD ecosystem. The performance in the Medical Internet of Things environment was also commendable, as the model achieved 79% accuracy, recall, specificity, precision, and F1 scores, and a 20.59% false-positive rate. While these scores in Medical IoT did not reach 90%, they demonstrate a significant improvement over CNNs, LSTMs, and autoencoders. The performance on the Edge Industrial Internet of Things dataset was 96% for precision, recall, specificity, and accuracy, and 3.12% for the false positive rate.

The four-layered architecture of the hybrid model deviated from conventional three-layer architectures by eliminating the classification heads. Many studies on deep autoencoder models for intrusion detection systems have focused on the encoder, bottleneck, and decoder layers, emphasizing their roles in threat identification through compression and reconstruction (Cunningham *et al.*, 2020; Basati & Faghih, 2023). However, this design approach failed to yield reasonable scores, with the best scores in the ICU dataset being 50% for accuracy, recall, and precision, 49.67% for F1, and 88.54% for specificity. This situation led to the integration of machine learning into the hybrid deep autoencoder, thereby addressing the deep autoencoder's limitations. This approach aligns with the findings of Eguavoen *et al.* (2025), who integrated an AdaBoost machine learning model into a CNN-LSTM-based deep learning model. The researchers noted

that this design approach improved the model's performance, leading to 99% accuracy, precision, recall, and F1 scores (Eguavoen *et al.*, 2025). The findings of these researchers align with this study, as the MLP classification head for the developed model improved overall performance from 50% to 100% on the ICU dataset. Additionally, the incorporation of a bidirectional LSTM (Bi-LSTM) bottleneck provides a robust environment for capturing temporal dependencies, outperforming conventional LSTM models (Jouhari & Guizani, 2024). This Bi-LSTM layer improved the model's performance, further outperforming the conventional hybrid deep autoencoder. Thus, the model's design and component selection aligned with those of other researchers' experiments, demonstrating consistency in IMD research.

The results obtained from this study have three implications for securing IMD ecosystems. First, the results suggest that cloud-based IMD IDS solutions can achieve near-perfect classification functionalities for intrusions. This situation encourages further development of specialized hybrid intrusion detection systems for healthcare ecosystems to address challenges encountered with conventional solutions. For instance, the 100% accuracy scores in the ICU dataset demonstrate that real-time intrusion detection for advanced threats and zero-day attacks is feasible in a healthcare setting if comprehensive training data is available. Secondly, the findings show the benefits of hybrid approaches in developing cybersecurity solutions. The improvements achieved by integrating LSTM, CNN, and MLP classification heads into the autoencoder model resulted in high accuracy scores. This phenomenon shows that temporal dynamics within traffic flows should not be ignored when monitoring for threats, especially in healthcare ecosystems. Lastly, the excellent scores of the developed model across different datasets demonstrate that hybrid IDS models are generalizable beyond the initial focus area for the IoT devices. This feature reduces the design challenges that may require restructuring core features to handle different scenarios. Instead, the appropriate dataset is obtained and appropriately pre-processed to enhance the high performance of the deployed model.

5.3 Optimization of Inference Speed and Detection Accuracy

The integration of Nadam optimization and an MLP classification head into the hybrid deep autoencoder model improved detection accuracy and inference speed. For instance,

the Nadam-optimized, MLP-enhanced hybrid model achieved 100% accuracy on the ICU dataset with a 0% false-positive rate. This performance was significantly higher than that of the standard hybrid model, which recorded an accuracy of 50.72% and a false-positive rate of 11.46%. The model's high-performance improvements demonstrate the benefits of hybrid optimization strategies for refining the performance of deep autoencoder models. The model's detection speed slightly improved with optimization. For instance, the best detection speed for the Nadam-optimized, MLP-enhanced model in the ICU dataset was 1.20 seconds, whereas that of the standard hybrid model was 1.35 seconds. However, the models with the best performance recorded 1.34 seconds for four-hidden-layers' 60/40, 2.59 seconds for one hidden layer's 80/20, 2.62 seconds for five-hidden-layers' 70/30, and 2.72 seconds for six-hidden-layer 70/30. These speeds were substantially higher than those recorded by the machine learning models, where the Gradient Boosting model had 0.03 seconds. Further, they were higher than those in baseline models, such as LSTM, which recorded 0.64 seconds.

The comparison of scores obtained from the proposed model with those from other deep learning-based intrusion detection models demonstrates the hybrid model's reasonable performance. For instance, a hybrid model comprising a CNN and a Bidirectional LSTM achieved accuracies of 97.90%, recall of 97.90%, and F1 of 97.91% on the UNSW-NB15 dataset. While the proposed model leveraged a different dataset, it achieved significant improvements over these scores (Jouhari *et al.*, 2024). The model had a detection score of 1.1 seconds. A separate deep learning-based intrusion detection model that used CNN and bidirectional LSTM scored 99.53% accuracy on the CIC-IDS2017 dataset, and another scored 99.9% accuracy on the NSL-KDD dataset (Jouhari & Guizani, 2024). These scores were slightly lower than those recorded in the proposed model's ICU dataset, but were higher than those in the WUSTL and Edge IIoT datasets. This phenomenon indicates that the proposed hybrid model excelled in the IMD-specific dataset but performed fairly in healthcare IoT and general IoT datasets.

While these models lacked scores for detection time, the accuracy scores offer comparative insights into the proposed model. For instance, there is a trade-off between performance and detection speed, where excellent accuracy and low false-positive rates

have a detrimental impact on detection time. While the 60/40 four-hidden-layer variant of the hybrid model is preferred for its high performance and low detection time, it does not achieve the excellent detection speeds seen in machine learning and baseline models. Nevertheless, the relatively low inference time of 1.34 seconds offers a reasonable detection environment for the IMD ecosystems. This relatively low inference time enables healthcare systems to respond to incidents. As a result, the cloud-based IDS for the IMDs can achieve high detection accuracy without significantly compromising on detection speed. Further, the gains in inference time for the four-hidden-layer model compared to the one-hidden-layer model imply that more complex models are feasible.

5.4 Comparative Evaluation of the Hybrid Deep Autoencoder Model

The comparative evaluation of the developed hybrid deep autoencoder against autoencoders, long short-term memory networks, and convolutional neural networks, based on speed, memory utilization, and detection accuracy, demonstrates significant improvements in the model. For instance, the CNN model achieved 71.45% accuracy on the ICU dataset, while the LSTM achieved 50.33%. The autoencoder achieved 30.05% accuracy, while the hybrid model achieved 100%. However, the detection speed for the best hybrid model was 1.33 seconds, whereas the LSTM recorded 0.64 seconds, and the autoencoder recorded 1.33 seconds. While memory utilization was not directly captured, these comparative results demonstrate that the hybrid model achieved significant improvements and performed reasonably well in terms of inference time. In machine learning models like Multilayer Perceptron, Gradient Boosting, and Random Forests, the models achieved 100% accuracy and detection speeds of 0.71 seconds, 0.3 seconds, and 0.5 seconds, respectively. While the developed model achieved similar accuracy, it did not match the detection speeds of these machine learning approaches.

The developed model performs reasonably well compared to other models. The 100% accuracy scores in the ICU dataset outperform those reported for the CNN-TM hybrid model proposed by Harshavardhan *et al.* (2025). This model achieved 97.89% accuracy and 97.20% PR. However, the model outperformed the MLP-enhanced, Nadam-optimized model on the WUSTL dataset but was within the same range on the Edge IIoT dataset. The MLP-enhanced model performed within the same range as the hybrid LSTM-CNN-Attention-based model proposed by Gueriani *et al.* (2025). This MLP-

enhanced model achieved 100% precision, recall, and F1, and a 0% false-positive rate. However, this model outperforms the MLP-enhanced model in Edge IIoT and WUSTL datasets. The comparative performance of the developed model against existing hybrid models demonstrates consistency and significant performance improvements achieved through the MLP classifier.

Based on the model's results, there is a trade-off between model accuracy, recall, precision, and F1, and inference speed. The hybrid model achieves 100% accuracy, recall, precision, and F1, and a 0% false-positive rate. This phenomenon indicates that the model provides the optimal detection capability required by the IMD ecosystem. However, the hybrid model shows significant degradation in inference speed. For example, the model’s accuracy score of 1.33 seconds is significantly lower than that of the baseline model, such as Long Short-Term Memory at 0.64 seconds. This situation indicates that LSTM identifies threats twice as fast as the proposed hybrid model. While this detection speed is a concern in a real-time detection environment, the model's overall accuracy must also be considered. There are numerous risks linked with missed threats in the IMD ecosystem. For instance, the attackers might shut down the device, modify dosage administration, or tamper with the appropriate communication of the biomarkers being monitored. On the other hand, a twice-slow detection with 100% accuracy would be preferred as this detection speed is less than 1.5 seconds, established in section 3.4.3. Consequently, the model cannot be considered as a real-time detection framework. Instead, it is a near-real-time detection system with relatively limited functionality.

5.5 Clinical and Operational Implications

There are two clinical implications of these findings, especially in the Implantable Medical Devices ecosystem. First, the high accuracy, recall, precision, and F1 scores in the MLP-enhanced hybrid model enhance clinical safety. In the testing environment, the model achieved 100% accuracy and 0% false-positive rate, indicating no missed threats. While the actual implementation may differ, the model is expected to produce negligible misclassifications. When misclassifications occur, the model needs to be retrained on the target dataset to resolve the errors. This approach recognizes the evolving threat landscape and acknowledges the need to adapt the threat detection

engine to changing environmental factors. However, this training is done on a cloud platform, reducing disruption to patient devices. Secondly, the low inference speed identified from the model demonstrates promise for near-real-time detection capabilities in IMD ecosystems. The IMD devices require real-time monitoring and prompt threat identification to enhance patient safety. The 1.33 second latency experienced by the model is reasonable, offering near-real-time detection capabilities for the IMD ecosystem.

Three operational implications derived from these results shape the adoption of the IDS in healthcare security governance. First, the high accuracy scores obtained in the ICU dataset, despite its small sample size, demonstrate the viability of implementing Deep Autoencoder solutions in IMD controllers. The resource-constrained nature of these devices limits the implementation of comprehensive security solutions. However, the results demonstrate that even with limited datasets, it is possible to achieve high accuracy comparable to that of conventional systems. This insight enhances the adoption of these advanced solutions even in IoT ecosystems. Secondly, the ability to train the model in the cloud and use a pre-trained model on the entire ICU dataset demonstrates the promise of integrating advanced threat-detection mechanisms via application programming interfaces. The APIs reduce the need to locally host the dataset or model, reducing the storage requirements for managing elaborate IDS solutions. Lastly, the trade-off between inference time and detection accuracy shapes the model's deployment in threat monitoring. While real-time cloud-based IMD systems may experience slight latency when using the IDS, near-real-time cloud-based IMD ecosystems may not exhibit similar characteristics. As a result, healthcare system administrators need to implement the device where a classification latency of 1.33 seconds does not compromise the efficiency of service provision.

There are three considerations to make before implementing the IDS to ensure it aligns with the U.S. Food and Drug Administration (FDA) and the Health Insurance Portability and Accountability Act (HIPAA). First, the healthcare service provider needs to limit data collection to the minimum necessary and ensure the collected data is encrypted at rest and in transit (Catuogno & Galdi, 2024). This approach ensures compliance with HIPAA's privacy and security rules. Secondly, regular risk

assessments are needed for the IDS solution. This risk assessment needs to define threat notification procedures, evaluate the IDS's efficiency in identifying past threats, and maintain the risk registers (Martínez *et al.*, 2023). Lastly, it is crucial for the healthcare service providers implementing the IDS to integrate its behavior into the medical device lifecycle. This approach involves incorporating it into design controls and risk management. Additionally, it is essential to conduct adequate clinical tests and provide comprehensive documentation detailing the solution's performance scores and workflows. This approach enables the FDA to review the IDS and make recommendations based on its tests.

CHAPTER SIX
SUMMARY, CONCLUSION, AND RECOMMENDATIONS

6.1 Summary

The first research sought to develop a hybrid deep autoencoder model by integrating convolutional neural networks, long short-term memory, and autoencoder components to provide a robust threat-detection environment for implantable medical devices. The experimental results show that the autoencoders, convolutional neural networks, and long shot-term memory models brought their strengths and weaknesses when implemented in the hybrid deep autoencoder model. For instance, the autoencoders provided an unsupervised learning environment that handled unseen datasets, while CNNs offered the spatial feature mapping to monitor short packet flows. The LSTM model captured the sequential patterns in the dataset, thereby enhancing the monitoring of long-range packet flows. A Multilayer Perceptron-based classification head was integrated into the model to refine its performance. While this classification head was not envisioned at the outset of the research, it proved effective in improving the model's performance. Thus, the design objectives intended for the first objective were adequately covered in the study.

The second objective is to optimize inference speed and detection accuracy for the hybrid deep autoencoder model. The integration of different components into the hybrid model led to a significant improvement in the detection accuracy. For instance, there was 100% accuracy, recall, precision, and F1 scores for one hidden layer's 80/20, four-hidden-layers' 60/40, and the 70/30 variants of five and six hidden layers on the ICU dataset. These models achieved a 0% false-positive rate, demonstrating the benefits of hybridization. However, inference speed did not improve with optimization as expected. While the hybrid model's inference speed outperformed that of the 70/30 and 80/20 variants of the autoencoder, it lagged behind those of the LSTM and CNN variants. This phenomenon demonstrates a subtle trade-off between inference time and detection accuracy. Increasing one without considering the other may negatively affect the overall model performance.

The last objective focused on assessing the performance of a hybrid deep autoencoder model against the standalone models. The experimental results showed that the hybrid model outperformed standalone models in precision, recall, specificity, accuracy, and

F1 scores across all datasets and training/testing ratios. The superiority of the hybrid model over these standalone variants was attributed to its ability to leverage the strengths of individual components to refine its performance. However, the model lagged behind most standalone models in inference speed. This phenomenon demonstrates the model's complexity when integrating different components, thereby increasing computational costs. As the computational costs increased, the model's inference time increased. Despite this latency, the difference between the hybrid model and standalone models was in seconds, demonstrating significant improvement. For instance, the selected hybrid model achieved an inference time of 1.33 seconds, while the 70/30 variant of the autoencoder had 1.34 seconds, coinciding with the score obtained for the CNN's 60/40 variant. The LSTM's 80/20 variant recorded 1.36 seconds, which was slower than the hybrid model. Based on these findings, the hybrid model provided a more balanced and robust framework for real-time intrusion detection in IMDs than standalone models.

6.2 Conclusion

Three conclusions are derived from the research findings, shaping the adoption of a hybrid intrusion detection system for Implantable Medical Devices. First, integrating a Multilayer Perceptron into the model significantly improved its performance across the datasets. The MLP-enhanced models outperformed both the standard and the Nadam-optimized hybrid models, even with the same number of hidden layers. This phenomenon was attributed to the MLP component of the model, which enabled it to capture nonlinear representations in the datasets. This nonlinear feature mapping extended the capabilities of the CNN and LSTM components of the model, achieving 100% accuracy, precision, recall, and F1 on the ICU dataset. The model's superior performance in a comparable number of hidden layers demonstrates that the interaction between the CNN, LSTM, and MLP components in the hybrid model refined its performance.

Secondly, increasing the number of hidden layers in the hybrid model did not consistently improve accuracy or inference time. The model's accuracy scores were not correlated with the number of hidden layers. There were numerous cases across the datasets where a low or moderate number of hidden layers outperformed a higher

number. This finding dispels the misconception that more hidden layers lead to better accuracy, recall, and precision. Further, there are instances in which inference time for a moderate number of hidden layers, e.g., four, was higher than that for one or two. This phenomenon demonstrates that increasing the number of layers does not necessarily increase the model's computational cost. Instead, such aspects are determined by other parameters apart from the number of layers. Nevertheless, having a high number of hidden layers has detrimental impacts on the model's accuracy and inference time. This situation is attributed to the model's increasing complexity and overfitting, which affect the solution's ability to generalize to unseen data.

Lastly, there is a subtle trade-off between performance and latency for a hybrid deep autoencoder model. While more layers can improve performance, this phenomenon increases the model's complexity. The outcome of this situation is increased computational costs, affecting the training and deployment of the models. This computational cost is demonstrated by inference time: models with a high number of hidden layers take longer to detect. There is a need for a delicate balance among accuracy, precision, recall, and F1, while accounting for latency. This balance ensures that critical healthcare systems are adequately protected through a near-real-time IDS solution without compromising its detection capabilities. A model with shorter detection time but high accuracy is preferred over one with high accuracy but slow detection speed. While the developed model achieved twice the detection time of the LSTM model, the high accuracy and relatively short detection time of 1.33 seconds are a reasonable trade-off. Thus, these insights advocate for a balance between latency and accuracy in IMD ecosystems.

6.3 Recommendations of the Study

i. The four-hidden-layer' 60/40 variant trained on the ICU dataset should be adopted for implementation. This model is selected due to its high performance and low detection time compared to the other major models in the dataset. The trained model must be implemented as a controlled prototype within the IMD controller. This implementation approach ensures that a human administrator monitors the model, assessing its performance and detection behavior. This

monitoring should be conducted for at least a month to provide a comprehensive assessment of the device's environment.

ii. There is a need to embed artificial intelligence intrusion detection as a standard component in the IMD ecosystems. Healthcare regulators need to mandate that all new IMD solutions include certified on-device or on-edge-deployed IDS modules that comply with ISO/IEC 27001 for information security systems. This compliance ensures that these devices integrate robust security features that strengthen the confidentiality, integrity, and availability of information services within the devices.

iii. There is a need for an annual framework of third-party audits. This framework verifies the detector's efficacy, update frequency, and threat coverage, and provides mitigation measures for IMD solutions that fail to meet industry security needs. Similarly, there is a need for a secure data-sharing consortium among the healthcare service providers. This consortium provides anonymous IMD network tracing for continuous model refinement and collective threat intelligence. The government needs to support this venture, stimulating research and development in the model and its accompanying computing components.

iv. There is a need to publish open interoperability standards that focus on application programming interface (API) definitions, data schemas, and alert formats. Publication of these standards will enhance collective monitoring of threats and notifications across the healthcare service providers, strengthening responsiveness in the sector. Thus, these considerations will improve the service of healthcare delivery objectives through robust security frameworks for the IMD ecosystems.

6.4 Recommendations for Future Research

i. Additional research is required to improve the model’s performance across all conditions. The model performed relatively poorly on the WUSTL and Edge IIoT datasets compared to the ICU dataset. Such low performance provides opportunities for improvement in medical and general IoT scenarios. As a result, there is a need for additional research on design approaches to improve the model's performance across all datasets.

ii. There is a need to research suitable model design strategies that enhance detection speeds, enabling the model to meet benchmarks provided by machine learning models. This consideration identifies setbacks in the current design strategy. It offers practical solutions to address these limitations and evaluates their efficiency at mitigating latency issues in the current design.

iii. There is a need to explore explainable artificial intelligence solutions for healthcare security, enabling clinicians to understand the decisions made by these systems. This focus will facilitate healthcare security professionals to understand autonomous IDS systems, identify and address their limitations, and contribute to the sector's overall security landscape. For instance, there is a need to examine the implementation of feature importance ranking mechanisms such as the Shapley Additive Explanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME). The focus on these explainable AI approaches for feature importance ranking will enhance understanding of feature engineering and model performance.

iv. There there is a need for subsequent research to validate using real clinical workflows. This research direction aims to deploy the developed model in live clinical settings, assess its performance in a production environment, address implementation challenges, and facilitate its implementation to address IMD security issues. This approach will offer an evidence-based assessment of the developed model, aligning with the healthcare security goals.

REFERENCES

Abou-Nassar, E. M., Iliyasu, A. M., El-Kafrawy, P. M., Song, O. Y., Bashir, A. K., & El-Latif, A. A. A. (2020). DITrust Chain: Towards Blockchain-Based Trust Models for Sustainable Healthcare IoT Systems. *IEEE Access*, *8*, 111223–111238. https://doi.org/10.1109/ACCESS.2020.2999468

Alkhabbas, F., Spalazzese, R., & Davidsson, P. (2020). An Agent-Based Approach to Realize Emergent Configurations in the Internet of Things. *Electronics*, *9*(9), 1347. https://doi.org/10.3390/electronics9091347

Aloul, F., Zualkernan, I., Abdalgawad, N., Hussain, L., & Sakhnini, D. (2021). Network Intrusion Detection on the IoT Edge Using Adversarial Autoencoders. *2021 International Conference on Information Technology, ICIT 2021 - Proceedings*, 120–125. https://doi.org/10.1109/ICIT52682.2021.9491694

Andresini, G., Appice, A., Di Mauro, N., Loglisci, C., & Malerba, D. (2019). Exploiting the auto-encoder residual error for intrusion detection. *Proceedings - 4th IEEE European Symposium on Security and Privacy Workshops, EUROS and PW 2019*, 281–290. https://doi.org/10.1109/EuroSPW.2019.00038

Arias-Duart, A., Mariotti, E., Garcia-Gasulla, D., & Alonso-Moral, J. M. (2023). A Confusion Matrix for Evaluating Feature Attribution Methods. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 3709–3714. https://doi.org/10.1109/CVPRW59228.2 023.00380

Asghari, P., Rahmani, A. M., & Haj Seyyed Javadi, H. (2019). A medical monitoring scheme and health-medical service composition model in cloud-based IoT platform. *Transactions on Emerging Telecommunications Technologies*, *30*(6). https://doi.org/10.1002/ett.3637

Azimi, I., Takalo-Mattila, J., Anzanpour, A., Rahmani, A. M., Soininen, J.-P., & Liljeberg, P. (2018). Empowering healthcare IoT systems with hierarchical edge-based deep learning. *Proceedings of the 2018 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies*, 63–68. https://doi.org/10.1145/3278576.3278597

Azizjon, M., Jumabek, A., & Kim, W. (2020). 1D CNN based network intrusion detection with normalization on imbalanced data. *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 218–224. https://doi.org/10.1109/ICAIIC48513.2020.9064976

Balaji, S. (2020, August 29). *Binary Image classifier CNN using TensorFlow*. Medium.Com. https://jokerpt.medium.com/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697

Basati, A., & Faghih, M. M. (2023). APAE: an IoT intrusion detection system using asymmetric parallel auto-encoder. *Neural Computing and Applications*, *35*(7), 4813–4833. https://doi.org/10.1007/s00521-021-06011-9

Bhuiyan, M. N., Rahman, M. M., Billah, M. M., & Saha, D. (2021). Internet of Things (IoT): A Review of Its Enabling Technologies in Healthcare Applications, Standards Protocols, Security, and Market Opportunities. *IEEE Internet of Things Journal*, *8*(13), 10474–10498. https://doi.org/10.1109/JIOT.2021.3062630

Butt, S. A., Diaz-Martinez, J. L., Jamal, T., Ali, A., De-La-Hoz-Franco, E., & Shoaib, M. (2019). IoT Smart Health Security Threats. *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*, 26–31. https://doi.org/10.1109/ICCSA.2019.000-8

Catuogno, L., & Galdi, C. (2024). Implantable Medical Device Security. *Cryptography*, *8*(4). https://doi.org/10.3390/cryptography8040053

Chandekar, P., Mehta, M., & Chandan, S. (2025). *Enhanced Anomaly Detection in IoMT Networks using Ensemble AI Models on the CICIoMT2024 Dataset*. http://arxiv.org/abs/2502.11854

Chandra, N. A., Ramli, K., Ratna, A. A. P., & Gunawan, T. S. (2022). Information Security Risk Assessment Using Situational Awareness Frameworks and Application Tools. *Risks*, *10*(8), 1–26. https://doi.org/10.3390/risks10080165

Cheng, J., Dekkers, J. C. M., & Fernando, R. L. (2021). Cross□validation of best linear unbiased predictions of breeding values using an efficient leave□one□out strategy. *Journal of Animal Breeding and Genetics*, *138*(5), 519–527. https://doi.org/10.1111/jbg.12545

Chinta, S. (2024). Edge AI for Real-Time Decision Making in IOT Networks. *International Journal of Innovative Research in Computer and Communication Engineering*, *12*(09), 11293–11309. https://doi.org/10.156 80 /ijircce.2024.1209044

Cunningham, J. D., Shu, D., Simpson, T. W., & Tucker, C. S. (2020). A sparsity preserving genetic algorithm for extracting diverse functional 3D designs from deep generative neural networks. *Design Science*, *6*(May), e11. https://doi.org/10.1017/dsj.2020.9

Dantas, P. V., Sabino da Silva, W., Cordeiro, L. C., & Carvalho, C. B. (2024). A comprehensive review of model compression techniques in machine learning. *Applied Intelligence*, *54*(22), 11804–11844. https://doi.org/10.10 07/s10489-024-05747-w

de Souza, C. A., Westphall, C. B., Machado, R. B., Sobral, J. B. M., & Vieira, G. dos S. (2020). Hybrid approach to intrusion detection in fog-based IoT environments. *Computer Networks*, *180*, 107417. https://doi.org/10.1016/j. comnet.2020.107417

Dhany, H. W., & Izhari, F. (2023). Exploratory Data Analysis (EDA) Methods for Healthcare Classification. *Journal of Intelligent Decision Support System*, *6*(4), 209–215

Dizdarević, J., Carpio, F., Jukan, A., & Masip-Bruin, X. (2019). A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Computing Surveys*, *51*(6), 1–30. https://doi.org/10.1145/3292674

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, *503*, 92–108. https://doi.org/10.1016/j.neucom.2022.06.111

Easttom, C., & Mei, N. (2019). Mitigating Implanted Medical Device Cybersecurity Risks. *2019 IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2019*, 0145–0148. https://doi.org/10.1109/UEMCON47517.2019.8992922

Edwards, A., Camacho-Collados, J., de Ribaupierre, H., & Preece, A. (2020). Go Simple and Pre-Train on Domain-Specific Corpora: On the Role of Training Data for Text Classification. *COLING 2020 - 28th International Conference on Computational Linguistics, Proceedings of the Conference*, 5522–5529. https://doi.org/10.18653/v1/2020.coling-main.481

Eguavoen, V. O., Olanrewaju, B. S., & Okafor, C. N. (2025). A Hybrid CNN-LSTM and Adaboost Model for Classifying Intrusion in IoT Networks. *Fudma Journal of Sciences*, *9*(5), 204–212. https://doi.org/10.33003/fjs-2025-0905-3495

El-Bakkouri, N., & Mazri, T. (2020). Security threats in smart healthcare. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIV-4/W3-*, 209–214. https://doi.org/10. 5194/isprs-archives-XLIV-4-W3-2020-209-2020

El-Sherif, M., Khattab, A., & El-Soudani, M. (2024). *Intrusion Detection Using TCP/IP Single Packet Header Binary Image*. https://doi.org/10.2139/ssrn. 5000577

Farahani, B., Firouzi, F., Chang, V., Badaroglu, M., Constant, N., & Mankodiya, K. (2017). *Towards Fog-driven IoT eHealth: Promises and Challenges of IoT in Medicine and Healthcare*.

Fashoto, S. G., Mbunge, E., Ogunleye, G., & Van Den Burg, J. (2021). Implementation of machine learning for predicting maize crop yields using multiple linear regression and backward elimination. *Malaysian Journal of Computing*, *6*(1), 679–697.

Fenanir, S., Semchedine, F., Harous, S., & Baadache, A. (2020). A semi-supervised deep auto-encoder based intrusion detection for iot. *Ingenierie Des Systemes d'Information*, *25*(5), 569–577. https://doi.org/10.1828 0/ISI.250503

Fernandez, J., Na, C., Tiwari, V., Bisk, Y., Luccioni, S., & Strubell, E. (2025). *Energy Considerations of Large Language Model Inference and Efficiency Optimizations*. http://arxiv.org/abs/2504.17674

Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L., & Janicke, H. (2023). *Edge IIoTset Dataset*. IEEE. https://doi.org/10.21227/mbc1-1h68

Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, *50*, 102419. https://doi.org/10.1016/j.jisa.2019.102419

Firouzi, F., Chakrabarty, K., & Nassif, S. (2020). Intelligent Internet of Things: From Device to Fog and Cloud. In *Intelligent Internet of Things: From Device to Fog and Cloud*. Springer International Publishing. https://doi.org/10.1007/97 8-3-030-30367-9

Forestiero, A., & Papuzzo, G. (2021). Agents-Based Algorithm for a Distributed Information System in Internet of Things. *IEEE Internet of Things Journal*, *8*(22), 16548–16558. https://doi.org/10.1109/JIOT.2021.3074830

Goyal, P., Sahoo, A. K., & Sharma, T. K. (2021). Internet of things: Architecture and enabling technologies. *Materials Today: Proceedings*, *34*, 719–735. https://doi.org/10.1016/j.matpr.2020.04.678

Gueriani, A., Kheddar, H., & Mazari, A. C. (2025). *Adaptive Cyber-Attack Detection in IIoT Using Attention-Based LSTM-CNN Models*. https://doi.org/10.1 109/ICTI S62692.2024.10894509

Gupta, S. (2025). AI Agents Collaboration Under Resource Constraints: Practical Implementations. *International Journal of Artificial Intelligence Research and Development*, *3*(1), 51–63. https://doi.org/10.34218/IJA IRD_03_01_004

Habibzadeh, H., Dinesh, K., Rajabi Shishvan, O., Boggio-Dandry, A., Sharma, G., & Soyata, T. (2020). A Survey of Healthcare Internet of Things (HIoT): A Clinical Perspective. In *IEEE Internet of Things Journal* (Vol. 7, Issue 1, pp. 53–71). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1 109/JIOT.2019.2946359

Harshavardhan, A., Sree Vani, D. M., Anitha Patil, D., Yamsani, N., Archana, K., & Asst, S. (2025). Hybrid Deep Learning Framework for Intrusion Detection: Integrating CNN, LSTM, and Attention Mechanisms to Enhance Cybersecurity. *Journal of Theoretical and Applied Information Technology*, *15*(1). www.jatit.org

Hassija, V., Chamola, V., Bajpai, B. C., Naren, & Zeadally, S. (2021). Security issues in implantable medical devices: Fact or fiction? *Sustainable Cities and Society*, *66*(November 2020), 102552. https://doi.org/10.1016/j.scs. 2020.102552

Hayou, S., Doucet, A., & Rousseau, J. (2019). On the Impact of the Activation Function on Deep Neural Networks Training. *Proceedings of the 36th International Conference on Machine Learning*, 2672–2680

Hussain, F., Abbas, S. G., Shah, G. A., Pires, I. M., Fayyaz, U. U., Shahzad, F., Garcia, N. M., & Zdravevski, E. (2021). *IoT Healthcare Security Dataset*. IEEE. https://doi.org/10.21227/9w13-2t13

Isa, M. M., & Mhamdi, L. (2022). Hybrid Deep Autoencoder with Random Forest in Native SDN Intrusion Detection Environment. *IEEE International Conference on Communications*, *2022-May*, 1698–1703. https://doi.org/10.1109/ICC45855.2022.9838282

Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network Intrusion Detection Combined Hybrid Sampling with Deep Hierarchical Network. *IEEE Access*, *8*, 32464–32476. https://doi.org/10.1109/ACCESS.2020.2973730

Jouhari, M., & Guizani, M. (2024). *Lightweight CNN-BiLSTM based Intrusion Detection Systems for Resource-Constrained IoT Devices*. https://doi.org/10.1109/IWCMC61514.2024.10592352

Jouhari, M., Benaddi, H., & Ibrahimi, K. (2024). *Efficient Intrusion Detection: Combining $χ^2$ Feature Selection with CNN-BiLSTM on the UNSW-NB15 Dataset*. http://arxiv.org/abs/2407.14945

Kankam, P. K. (2019). The use of paradigms in information research. *Library & Information Science Research*, *41*(2), 85–92. https://doi.org/10.10 16/j.lisr.2019.04.003

Kassab, W., & Darabkh, K. A. (2020). A–Z survey of Internet of Things: Architectures, protocols, applications, recent advances, future directions and recommendations. *Journal of Network and Computer Applications*, *163*(2), 1–59. https://doi.org/10.1016/j.jnca.2020.102663

Kelly, L. M., & Cordeiro, M. (2020). Three principles of pragmatism for research on organizational processes. *Methodological Innovations*, *13*(2), 1–10. https://doi.org/10.1177/2059799120937242

Kilicarslan, S., Adem, K., & Çelik, M. (2021). An overview of the activation functions used in deep learning algorithms. *Journal of New Results in Science*, *10*(3), 75–88. https://doi.org/10.54187/jnrs.1011739

Kirimi, J. (2023). Examining the Performance of Gradient Boosting, Multilayer Perceptron, and Random Forests in Threat Detection for Medical Internet of Things. In V. K. Nyaga, L. Musalia, D. Obote, S. Munanu, D. Kiptui, F. Ngugi, J. Simiyu, E. Kathuni, J. Kiplangat, M. Karuri, J. Kirugua, D. Moywaywa, M. Wepukhulu, K. Tuei, L. Karema, M. Mwiti, W. Micheni, A. Rwanda, E. Karonco, … C. Gitonga (Eds.), *Tharaka University 5th Annual International Research Conference* (pp. 171–178). Tharaka University.

Kong, H.-J., An, S., Lee, S., Cho, S., Hong, J., Kim, S., & Lee, S. (2022). Usage of the Internet of Things in Medical Institutions and its Implications. *Healthcare Informatics Research*, *28*(4), 287–296. https://doi.org/10.4258 /hir.2022.28.4.287

Krichen, M. (2023). Convolutional Neural Networks: A Survey. *Computers*, *12*(8), 1–41. https://doi.org/10.3390/computers12080151

Kuchuk, H., & Malokhvii, E. (2024). Integration of IoT with Cloud, Fog, and Edge Computing: A Review. *Advanced Information Systems*, *8*(2), 65–78. https://doi.org/10.20998/2522-9052.2024.2.08

Kwarteng, E., & Cebe, M. (2022). A survey on security issues in modern Implantable Devices: Solutions and future issues. *Smart Health*, *25*. https://doi.org/10.1016/j.smhl.2022.100295

Lampe, B., & Meng, W. (2023). A survey of deep learning-based intrusion detection in automotive applications. In *Expert Systems with Applications* (Vol. 221). Elsevier Ltd. https://doi.org/10.1016/j.eswa.2023.119771

Lee, G. Y., Alzamil, L., Doskenov, B., & Termehchy, A. (2021). *A Survey on Data Cleaning Methods for Improved Machine Learning Model Performance*. http://arxiv.org/abs/2109.07127

Lee, K. J. (2021). Architecture of neural processing unit for deep neural networks. In *Advances in Computers* (Vol. 122, pp. 217–245). Academic Press Inc. https://doi.org/10.1016/bs.adcom.2020.11.001

Li, P., Rao, X., Blase, J., Zhang, Y., Chu, X., & Zhang, C. (2021). CleanML: A study for evaluating the impact of data cleaning on ml classification tasks. *Proceedings - International Conference on Data Engineering*, *2021-April*, 13–24. https://doi.org/10.1109/ICDE51399.2021.00009

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(12), 6999–7019. https://doi.org/10.1109/TNNLS.2021.3084827

Long, W., Lu, Z., & Cui, L. (2019). Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, *164*, 163–173. https://doi.org/10.1016/j.knosys.2018.10.034

Longras, A., Oliveira, H., & Paiva, S. (2020). Security Vulnerabilities on Implantable Medical Devices. *Iberian Conference on Information Systems and Technologies, CISTI*, *2020-Jun e*(June), 24–27. https://doi.org/10.2 391 9/CISTI495 56.2020.9141043

Martínez, A. L., Pérez, M. G., & Ruiz-Martínez, A. (2023). A Comprehensive Review of the State-of-the-Art on Security and Privacy Issues in Healthcare. *ACM Computing Surveys*, *55*(12). https://doi.org/10.1 145/3571156

Messinis, S., Temenos, N., Protonotarios, N. E., Rallis, I., Kalogeras, D., & Doulamis, N. (2024). Enhancing Internet of Medical Things security with artificial intelligence: A comprehensive review. In *Computers in Biology and Medicine* (Vol. 170). Elsevier Ltd. https://doi.org/10.10 16/j.compbiomed.2024.108036

Mienye, I. D., & Swart, T. G. (2025). Deep Autoencoder Neural Networks: A Comprehensive Review and New Perspectives. In *Archives of Computational Methods in Engineering*. Springer Science and Business Media B.V. https://doi.org/10.1007/s11831-025-10260-5

Moussa, M. M., & Alazzawi, L. (2021). A Hybrid Deep Learning Cyber-Attacks Intrusion Detection System for CAV Path Planning. *Midwest Symposium on Circuits and Systems*, *2021-Augus*, 607–610. https://doi.org/10.1109/MWSCAS47672.2021.9531858

Mwitta, C., Rains, G. C., & Prostko, E. (2024). Evaluation of Inference Performance of Deep Learning Models for Real-Time Weed Detection in an Embedded Computer. *Sensors*, *24*(2). https://doi.org/10.33 90/s24020514

Naseer, I. (2020). Implementation of Hybrid Mesh firewall and its future impacts on Enhancement of cyber security. *MZ Computing Journal*, *1*(2), 1–8.

Nehra, D., Mangat, V., & Kumar, K. (2021). *A Deep Learning Approach for Network Intrusion Detection Using Non-symmetric Auto-encoder* (P. Singh, B., Coello Coello, C.A., Jindal, P., Verma, Ed.; pp. 371–382). Springer. https://doi.org/10.1007/978-981-16-1295-4_38

Ortega-Fernandez, I., Sestelo, M., Burguillo, J. C., & Piñón-Blanco, C. (2023). Network intrusion detection system for DDoS attacks in ICS using deep autoencoders. *Wireless Networks*. https://doi.org/10.1007/s11276-022-03214-3

Pal, K., & Patel, B. V. (2020). Data Classification with k-fold Cross Validation and Holdout Accuracy Estimation Methods with 5 Different Machine Learning Techniques. *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, *Iccmc*, 83–87. https://doi.org/10.1109/ICCMC48092.2020.ICCMC-00016

Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., & Liljeberg, P. (2018). Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Generation Computer Systems*, *78*, 641–658. https://doi.org/10.1016 /j.future.2017.02.014

Rassam, M. A. (2024). Autoencoder-Based Neural Network Model for Anomaly Detection in Wireless Body Area Networks. *Internet of Things*, *5*(4), 852–870. https://doi.org/10.3390/iot5040039

Ray, P. P. (2018). A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, *30*(3), 291–319. https://doi.org/10.1016/j.jksuci.2016.10.003

Roy, B., & Cheung, H. (2018). A Deep Learning Approach for Intrusion Detection in Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network. *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, 1–6. https://doi.org /10.1109/ATNAC.2018.8615294

Sahoo, K., Samal, A. K., Pramanik, J., & Pani, S. K. (2019). Exploratory Data Analysis using Python. *International Journal of Innovative Technology and Exploring Engineering*, *8*(12), 4727–4735. https://doi.org/10.35940/ijitee.L3591.1081219

Savaglio, C., Ganzha, M., Paprzycki, M., Bădică, C., Ivanović, M., & Fortino, G. (2020). Agent-based Internet of Things: State-of-the-art and research challenges. *Future Generation Computer Systems*, *102*, 1038–1053. https://doi.org/10.1016/j.future.2019.09.016

Selvaraj, S., & Sundaravaradhan, S. (2020). Challenges and opportunities in IoT healthcare systems: a systematic review. In *SN Applied Sciences* (Vol. 2, Issue 1). Springer Nature. https://doi.org/10.1007/s42452-019-1925-y

Shah, R., & Chircu, A. (2018). IoT And AI in Healthcare: A Systematic Literature Review. *Issues In Information Systems*. https://doi.org/10.48009 /3_iis_2018 _33-41

Shehadeh, A., Altaweel, H., & Qusef, A. (2023). Analysis of Data Mining Techniques on KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets for Intrusion Detection. *2023 24th International Arab Conference on Information Technology, ACIT 2023*. https://doi.org/10.1109/ACIT 58888.2023.10453884

Siddiqi, M. A., Doerr, C., & Strydis, C. (2020). IMDfence: Architecting a Secure Protocol for Implantable Medical Devices. *IEEE Access*, *8*, 147948–147964. https://doi.org/10.1109/ACCESS.2020.3015686

Siddiqui, A. J., & Boukerche, A. (2021). Adaptive ensembles of autoencoders for unsupervised IoT network intrusion detection. *Computing*, *103*(6), 1209–1232. https://doi.org/10.1007/s00607-021-00912-2

Sinha, P., Sahu, D., Prakash, S., Yang, T., Rathore, R. S., & Pandey, V. K. (2025). A high-performance hybrid LSTM CNN secure architecture for IoT environments using deep learning. *Scientific Reports*, *15*(1). https://doi.org/10.1038/s41598-025-94500-5

Slimani, C., Morge-Rollet, L., Lemarchand, L., Espes, D., Le Roy, F., & Boukhobza, J. (2024). A study on characterizing energy, latency and security for Intrusion Detection Systems on heterogeneous embedded platforms. *Future Generation Computer Systems*, *162*. https://doi. org/10.1016/j.future.2024.07.051

Sobin, C. C. (2020). A Survey on Architecture, Protocols and Challenges in IoT. In *Wireless Personal Communications* (Vol. 112, Issue 3). https://doi.org/10.1007/s11277-020-07108-5

Soydaner, D. (2020). A Comparison of Optimization Algorithms for Deep Learning. *International Journal of Pattern Recognition and Artificial Intelligence*, *34*(13), 1–27. https://doi.org/10.1142/S0218001420520138

Sun, W., Cai, Z., Li, Y., Liu, F., Fang, S., & Wang, G. (2018). Security and Privacy in the Medical Internet of Things: A Review. In *Security and Communication Networks* (Vol. 2018). Hindawi Limited. https://doi.org/10.1155/2018/ 5978636

Taleb, H., Nasser, A., Andrieux, G., Charara, N., & Motta Cruz, E. (2021). Wireless technologies, medical applications and future challenges in WBAN: a survey. *Wireless Networks*, *27*(8), 5271–5295. https://doi.org/10.1007/s11276-021-02780-2

Tarricone, R., Ciani, O., Torbica, A., Brouwer, W., Chaloutsos, G., Drummond, M. F., Martelli, N., Persson, U., Leidl, R., Levin, L., Sampietro-Colom, L., & Taylor, R. S. (2020). Lifecycle evidence requirements for high-risk implantable medical devices: a European perspective. *Expert Review of Medical Devices*, *17*(10), 993–1006. https://doi.org/10.1080/174344 40.2020.1825074

Telikani, A., & Gandomi, A. H. (2021). Cost-sensitive stacked auto-encoders for intrusion detection in the Internet of Things. *Internet of Things (Netherlands)*, *14*. https://doi.org/10.1016/j.iot.2019.100122

Thamilarasu, G., Odesile, A., & Hoang, A. (2020). An intrusion detection system for internet of medical things. *IEEE Access*, *8*, 181560–181576. https://doi.org/10.1109/ACCESS.2020.3026260

Türk, F. (2023). Analysis of Intrusion Detection Systems in UNSW-NB15 and NSL-KDD Datasets with Machine Learning Algorithms. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, *12*(2), 465–477. https://doi.org/ 10.17798/bit lisfen.1240469

Varoquaux, G., & Colliot, O. (2023). Evaluating Machine Learning Models and Their Diagnostic Value. In *Neuromethods* (Vol. 197, pp. 601–630). Humana Press Inc. https://doi.org/10.1007/978-1-0716-3195-9_20

Wang, L., Jiang, K., & Shen, G. (2021). Wearable, Implantable, and Interventional Medical Devices Based on Smart Electronic Skins. *Advanced Materials Technologies*, *6*(6), 1–18. https://doi.org/10.1002/admt.202100107

Wang, Y., Han, D., & Cui, M. (2023). Intrusion detection model of internet of things based on deep learning. *Computer Science and Information Systems*, *20*(4), 1519–1540. https://doi.org/10.2298/CSIS230418058W

WUSTL. (2020). *WUSTL EHMS 2020*. WUSTL.

Yogev, D., Goldberg, T., Arami, A., Tejman-Yarden, S., Winkler, T. E., & Maoz, B. M. (2023). Current state of the art and future directions for implantable sensors in medical technology: Clinical needs and engineering challenges. In *APL Bioengineering* (Vol. 7, Issue 3). American Institute of Physics Inc. https://doi.org/10.1063/5.0152290

Zhao, X., Liang, J., & Dang, C. (2019). A stratified sampling based clustering algorithm for large-scale data. *Knowledge-Based Systems*, *163*, 416–428. https://doi.org/10.1016/j.knosys.2018.09.007

Zhou, J., Fu, W., Hu, W., Sun, Z., He, T., & Zhang, Z. (2024). Challenges and Advances in Analyzing TLS 1.3-Encrypted Traffic: A Comprehensive Survey. In *Electronics (Switzerland)* (Vol. 13, Issue 20). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/electronics1 3204000

APPENDICES

Appendix 1: Dataset Snapshots

Appendix 1.1: IoT Healthcare Security Dataset

frame.len	tcp.srcpor	tcp.dstpor	tcp.flags	tcp.time_	tcp.len	tcp.ack	tcp.pdu.si	tcp.windo	mqtt.clier	mqtt.clier	mqtt.cona	mqtt.kaliv
105	35161	1883	0x0000001	0	37	1	37	512	393b1640f	23	0	60
105	34237	1883	0x0000001	0	37	1	37	512	3790fc6f4	23	0	60
74	56808	1883	0x0000000	0	0	0	0	29200	0	0	0	0
74	1883	56808	0x0000001	5.20E-05	0	1	0	28960	0	0	0	0
74	56810	1883	0x0000000	0	0	0	0	29200	0	0	0	0
74	1883	56810	0x0000001	1.20E-05	0	1	0	28960	0	0	0	0
74	56812	1883	0x0000000	0	0	0	0	29200	0	0	0	0
74	1883	56812	0x0000001	1.00E-05	0	1	0	28960	0	0	0	0
105	40629	1883	0x0000001	0	37	1	37	512	59052927c	23	0	60
72	1883	40629	0x0000001	5.30E-05	4	38	4	512	0	0	0x0000000	0
105	45639	1883	0x0000001	0	37	1	37	512	82a0800ac	23	0	60
72	1883	45639	0x0000001	2.00E-05	4	38	4	512	0	0	0x0000000	0

Appendix 1.2: Electronic Health Management System Dataset

I	J	M	N	Q	R	W	AA	AF	AQ	AR
cLoad	DstLoa	SIntPkt	DIntPkt	SrcJitte	DstJitte	Dur	Load	Rate	ST	Attack
276914	92305	3.582333	1.9015	2.946239	1.6235	0.010747	369219	558.295	0	normal
230984	76995	4.294667	2.9015	3.091654	2.8625	0.012884	307979	465.694	0.4	normal
218470	72823	4.540667	3.2945	2.849841	3.1655	0.013622	291293	440.464	0.4	normal
203376	67792	4.877667	3.332	2.452252	3.257	0.014633	271168	410.032	0.4	normal
21394	7131	46.368	41.8805	6.425858	41.7855	0.139104	28525	43.133	0.44	Data Alter
20074	6691	49.416	44.197	10.0825	44.191	0.148248	26766	40.473	0.44	Data Alter
22537	7512	44.01733	42.798	2.12914	42.793	0.132052	30049	45.437	0.32	Data Alter
19026	6342	52.13967	50.406	2.943382	50.401	0.156419	25368	38.359	0.32	Data Alter
20160	6720	49.20667	42.166	10.02241	42.161	0.14762	26880	40.645	0.32	Data Alter

Appendix 1.3: Edge Industrial IoT Dataset

tcp.ack	tcp.ack_ra	tcp.checks	tcp.dstpor	tcp.flags	tcp.flags.a	tcp.len	tcp.optior	tcp.payloa	tcp.seq	tcp.srcpor	Attack_type
0	0	0	0	0	0	0	0	0	0	0	MITM
0	0	0	0	0	0	0	0	0	0	0	MITM
0	0	0	0	0	0	0	0	0	0	0	Fingerprinting
0	0	0	0	0	0	0	0	0	0	0	Fingerprinting
1	1.01E+09	12939	4321	24	1	464	0101080ac	9c951be3:	1	35306	Ransomware
465	2.66E+08	42781	35306	16	1	0	0101080a(	0	1	4321	Ransomware
1	1.59E+09	37278	58900	24	1	149	0101080a4	485454502	1	80	Uploading
2	1.59E+09	6248	58900	17	1	0	0101080a4	0	150	80	Uploading
0	0	45787	80	2	0	0	020405b4(	0	0	37048	SQL_injection
367	1.22E+09	20136	80	16	1	0	0101080a9	0	307	37046	SQL_injection
0	0	1052	80	2	0	0	020405b4(	0	0	56044	DDoS_HTTP
0	0	26933	80	2	0	0	020405b4(	0	0	56060	DDoS_HTTP
0	0	10089	80	2	0	0	020405b4(	0	0	56070	DDoS_HTTP
5.64E+08	5.64E+08	54332	80	2	0	120	0	5.9E+239	0	33443	DDoS_TCP
121	1.53E+09	37944	33589	20	1	0	0	0	1	80	DDoS_TCP
0	0	22925	80	2	0	0	020405b4(	0	0	37672	Password
222	3.64E+09	47958	37696	17	1	0	0101080ac	0	1759	80	Password

Appendix 2: Standard Hybrid Model Architectures

1 hidden layer of the Standard Model

Deep Autoencoder Model without Optimizer

Model: "sequential_4"

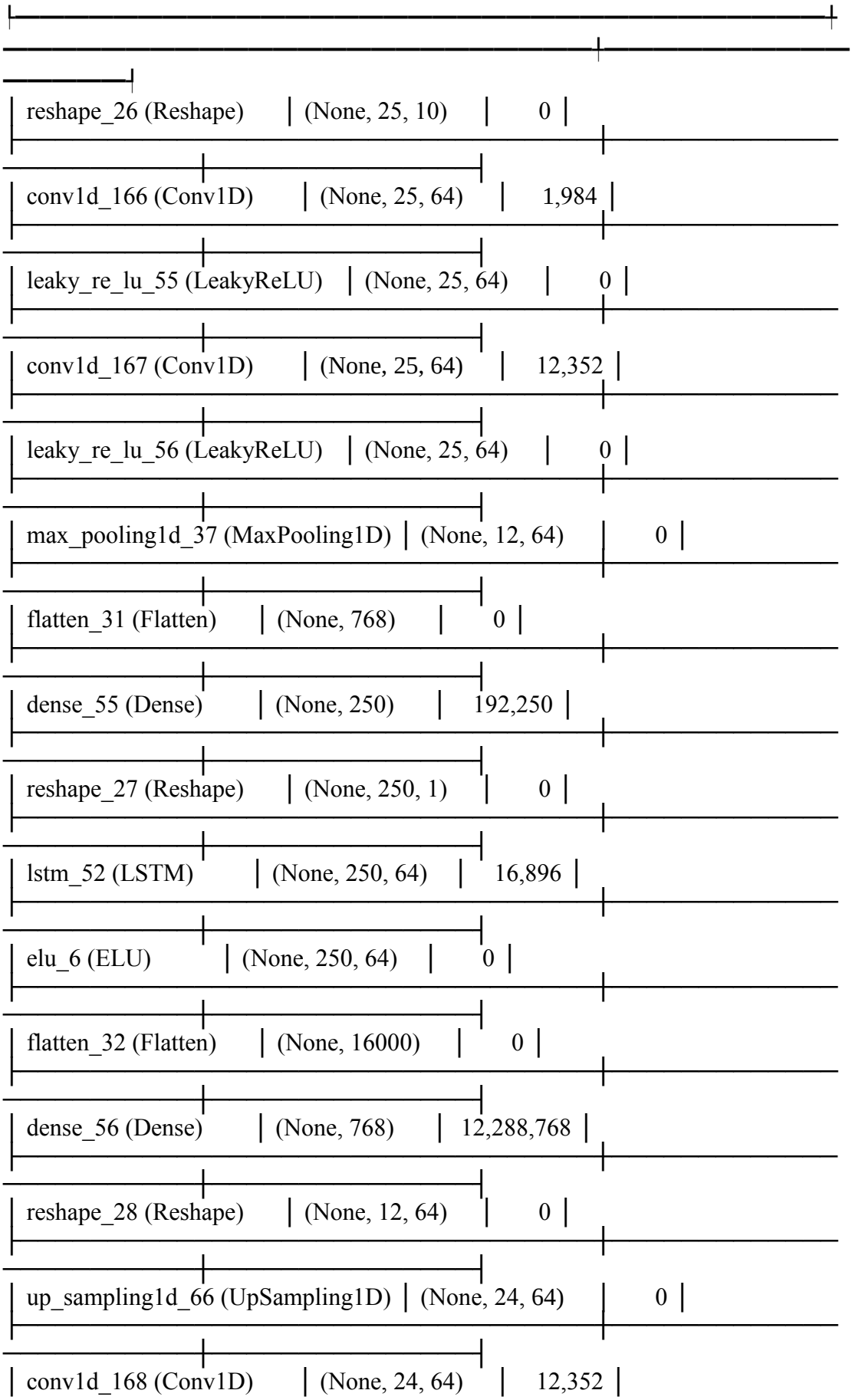
Layer (type)	Output Shape	Param #
reshape_14 (Reshape)	(None, 25, 10)	0
conv1d_160 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_49 (LeakyReLU)	(None, 25, 64)	0
max_pooling1d_34 (MaxPooling1D)	(None, 12, 64)	0
flatten_22 (Flatten)	(None, 768)	0
dense_46 (Dense)	(None, 250)	192,250

reshape_15 (Reshape)	(None, 250, 1)	0
lstm_49 (LSTM)	(None, 250, 64)	16,896
elu_3 (ELU)	(None, 250, 64)	0
flatten_23 (Flatten)	(None, 16000)	0
dense_47 (Dense)	(None, 768)	12,288,768
reshape_16 (Reshape)	(None, 12, 64)	0
up_sampling1d_63 (UpSampling1D)	(None, 24, 64)	0
conv1d_161 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_50 (LeakyReLU)	(None, 24, 64)	0
flatten_24 (Flatten)	(None, 1536)	0
dense_48 (Dense)	(None, 250)	384,250
reshape_17 (Reshape)	(None, 25, 10)	0

Total params: 12,896,500 (49.20 MB)
Trainable params: 12,896,500 (49.20 MB)
Non-trainable params: 0 (0.00 B)
2 hidden layers of the Standard Model

Deep Autoencoder Model without Optimizer
Model: "sequential_7"

Layer (type)	**Output Shape**	**Param #**


reshape_26 (Reshape) | (None, 25, 10) | 0 |
conv1d_166 (Conv1D) | (None, 25, 64) | 1,984 |
leaky_re_lu_55 (LeakyReLU) | (None, 25, 64) | 0 |
conv1d_167 (Conv1D) | (None, 25, 64) | 12,352 |
leaky_re_lu_56 (LeakyReLU) | (None, 25, 64) | 0 |
max_pooling1d_37 (MaxPooling1D) | (None, 12, 64) | 0 |
flatten_31 (Flatten) | (None, 768) | 0 |
dense_55 (Dense) | (None, 250) | 192,250 |
reshape_27 (Reshape) | (None, 250, 1) | 0 |
lstm_52 (LSTM) | (None, 250, 64) | 16,896 |
elu_6 (ELU) | (None, 250, 64) | 0 |
flatten_32 (Flatten) | (None, 16000) | 0 |
dense_56 (Dense) | (None, 768) | 12,288,768 |
reshape_28 (Reshape) | (None, 12, 64) | 0 |
up_sampling1d_66 (UpSampling1D) | (None, 24, 64) | 0 |
conv1d_168 (Conv1D) | (None, 24, 64) | 12,352 |

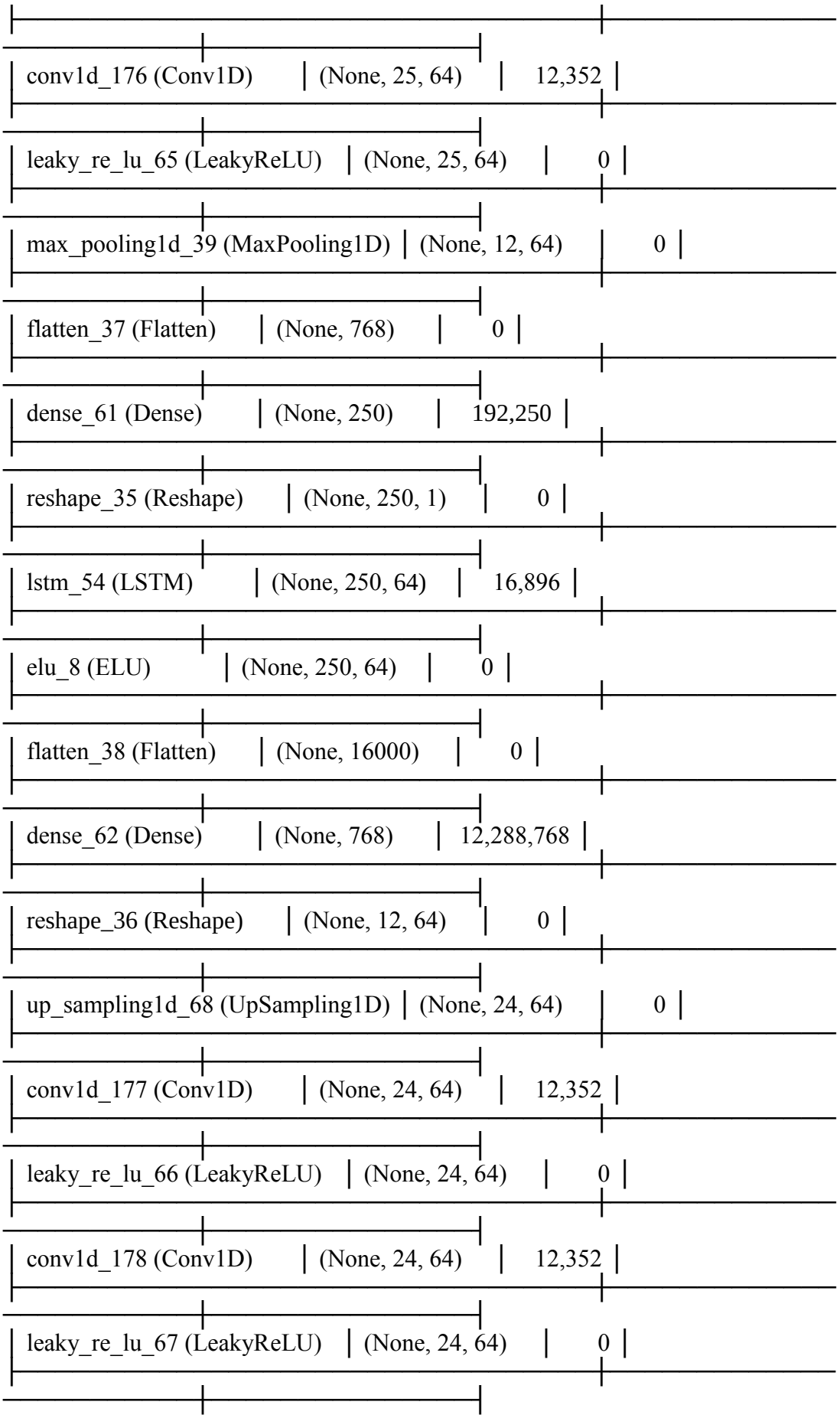
leaky_re_lu_57 (LeakyReLU)	(None, 24, 64)	0
conv1d_169 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_58 (LeakyReLU)	(None, 24, 64)	0
flatten_33 (Flatten)	(None, 1536)	0
dense_57 (Dense)	(None, 250)	384,250
reshape_29 (Reshape)	(None, 25, 10)	0

Total params: 12,921,204 (49.29 MB)
Trainable params: 12,921,204 (49.29 MB)
Non-trainable params: 0 (0.00 B)

3 hidden layers of the Standard Model

Deep Autoencoder Model without Optimizer
Model: "sequential_9"

Layer (type)	**Output Shape**	**Param #**
reshape_34 (Reshape)	(None, 25, 10)	0
conv1d_174 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_63 (LeakyReLU)	(None, 25, 64)	0
conv1d_175 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_64 (LeakyReLU)	(None, 25, 64)	0


conv1d_176 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_65 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_39 (MaxPooling1D) (None, 12, 64) 0
flatten_37 (Flatten) (None, 768) 0
dense_61 (Dense) (None, 250) 192,250
reshape_35 (Reshape) (None, 250, 1) 0
lstm_54 (LSTM) (None, 250, 64) 16,896
elu_8 (ELU) (None, 250, 64) 0
flatten_38 (Flatten) (None, 16000) 0
dense_62 (Dense) (None, 768) 12,288,768
reshape_36 (Reshape) (None, 12, 64) 0
up_sampling1d_68 (UpSampling1D) (None, 24, 64) 0
conv1d_177 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_66 (LeakyReLU) (None, 24, 64) 0
conv1d_178 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_67 (LeakyReLU) (None, 24, 64) 0

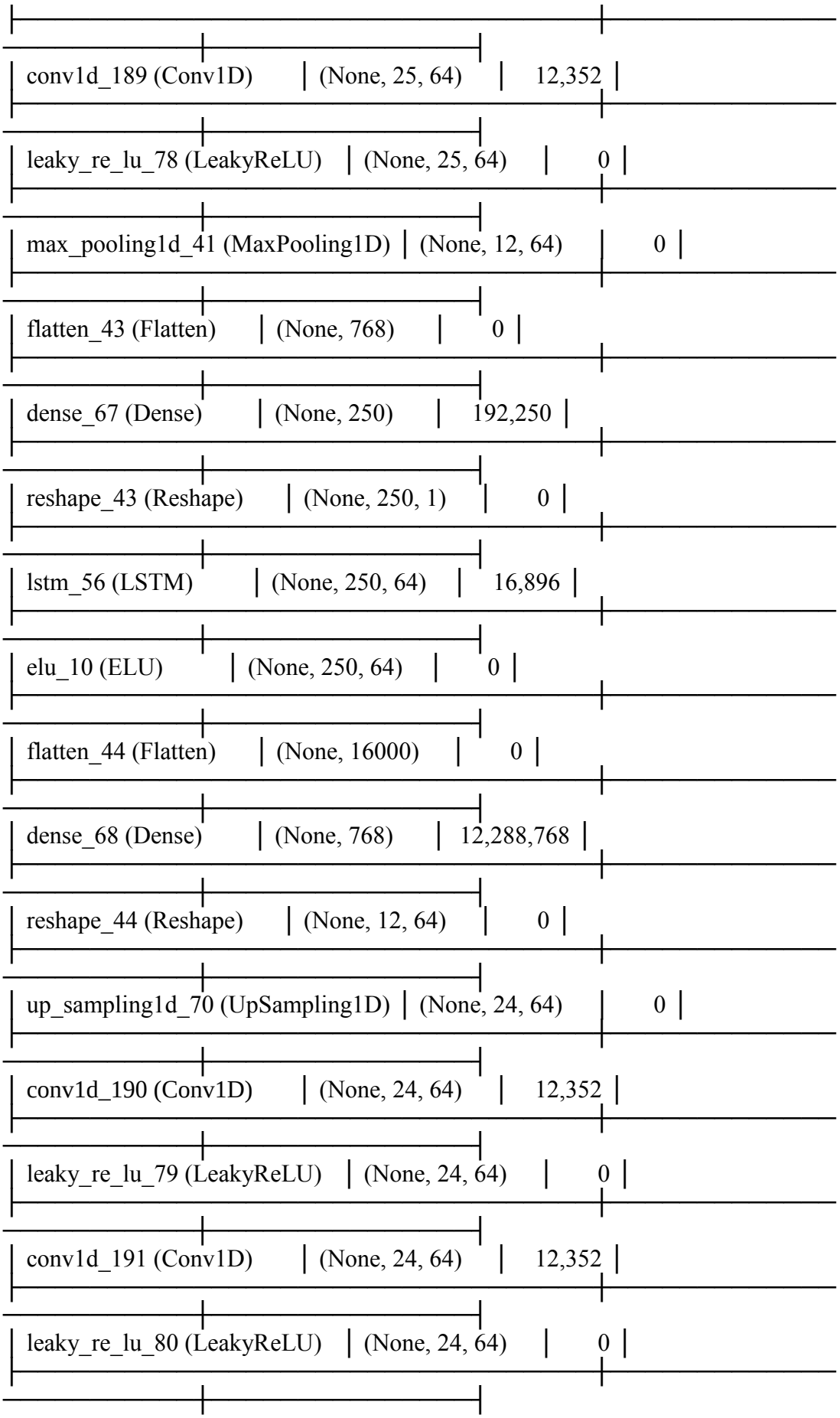
conv1d_179 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_68 (LeakyReLU)	(None, 24, 64)	0
flatten_39 (Flatten)	(None, 1536)	0
dense_63 (Dense)	(None, 250)	384,250
reshape_37 (Reshape)	(None, 25, 10)	0

Total params: 12,945,908 (49.38 MB)
Trainable params: 12,945,908 (49.38 MB)
Non-trainable params: 0 (0.00 B)

4 hidden layers of the Standard Model

Deep Autoencoder Model without Optimizer
Model: "sequential_11"

Layer (type)	**Output Shape**	**Param #**
reshape_42 (Reshape)	(None, 25, 10)	0
conv1d_186 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_75 (LeakyReLU)	(None, 25, 64)	0
conv1d_187 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_76 (LeakyReLU)	(None, 25, 64)	0
conv1d_188 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_77 (LeakyReLU)	(None, 25, 64)	0


conv1d_189 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_78 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_41 (MaxPooling1D) (None, 12, 64) 0
flatten_43 (Flatten) (None, 768) 0
dense_67 (Dense) (None, 250) 192,250
reshape_43 (Reshape) (None, 250, 1) 0
lstm_56 (LSTM) (None, 250, 64) 16,896
elu_10 (ELU) (None, 250, 64) 0
flatten_44 (Flatten) (None, 16000) 0
dense_68 (Dense) (None, 768) 12,288,768
reshape_44 (Reshape) (None, 12, 64) 0
up_sampling1d_70 (UpSampling1D) (None, 24, 64) 0
conv1d_190 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_79 (LeakyReLU) (None, 24, 64) 0
conv1d_191 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_80 (LeakyReLU) (None, 24, 64) 0

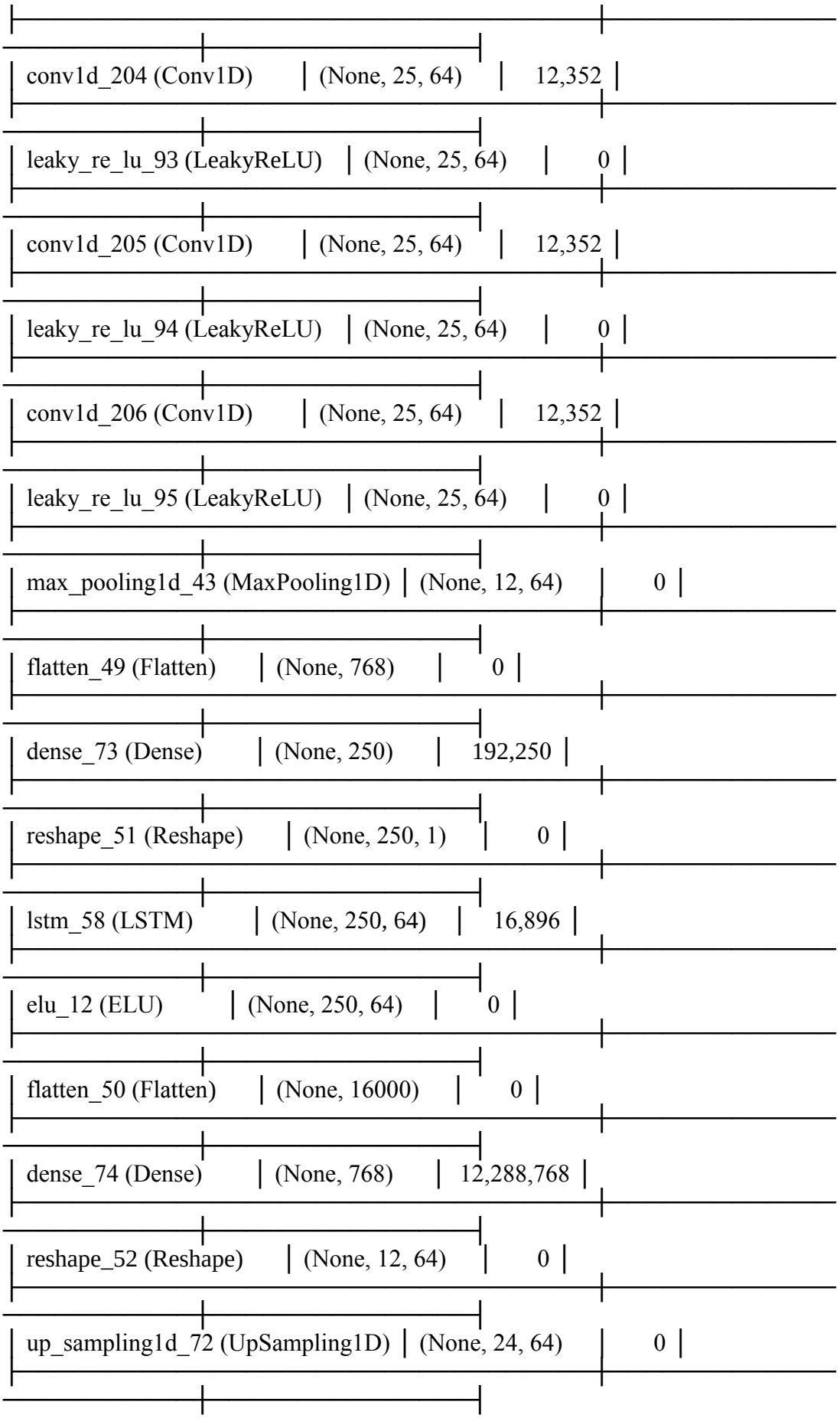
conv1d_192 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_81 (LeakyReLU)	(None, 24, 64)	0
conv1d_193 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_82 (LeakyReLU)	(None, 24, 64)	0
flatten_45 (Flatten)	(None, 1536)	0
dense_69 (Dense)	(None, 250)	384,250
reshape_45 (Reshape)	(None, 25, 10)	0

Total params: 12,970,612 (49.48 MB)
Trainable params: 12,970,612 (49.48 MB)
Non-trainable params: 0 (0.00 B)

5 hidden layers of the Standard Model

Deep Autoencoder Model without Optimizer
Model: "sequential_13"

Layer (type)	**Output Shape**	**Param #**
reshape_50 (Reshape)	(None, 25, 10)	0
conv1d_202 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_91 (LeakyReLU)	(None, 25, 64)	0
conv1d_203 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_92 (LeakyReLU)	(None, 25, 64)	0


conv1d_204 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_93 (LeakyReLU) (None, 25, 64) 0
conv1d_205 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_94 (LeakyReLU) (None, 25, 64) 0
conv1d_206 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_95 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_43 (MaxPooling1D) (None, 12, 64) 0
flatten_49 (Flatten) (None, 768) 0
dense_73 (Dense) (None, 250) 192,250
reshape_51 (Reshape) (None, 250, 1) 0
lstm_58 (LSTM) (None, 250, 64) 16,896
elu_12 (ELU) (None, 250, 64) 0
flatten_50 (Flatten) (None, 16000) 0
dense_74 (Dense) (None, 768) 12,288,768
reshape_52 (Reshape) (None, 12, 64) 0
up_sampling1d_72 (UpSampling1D) (None, 24, 64) 0

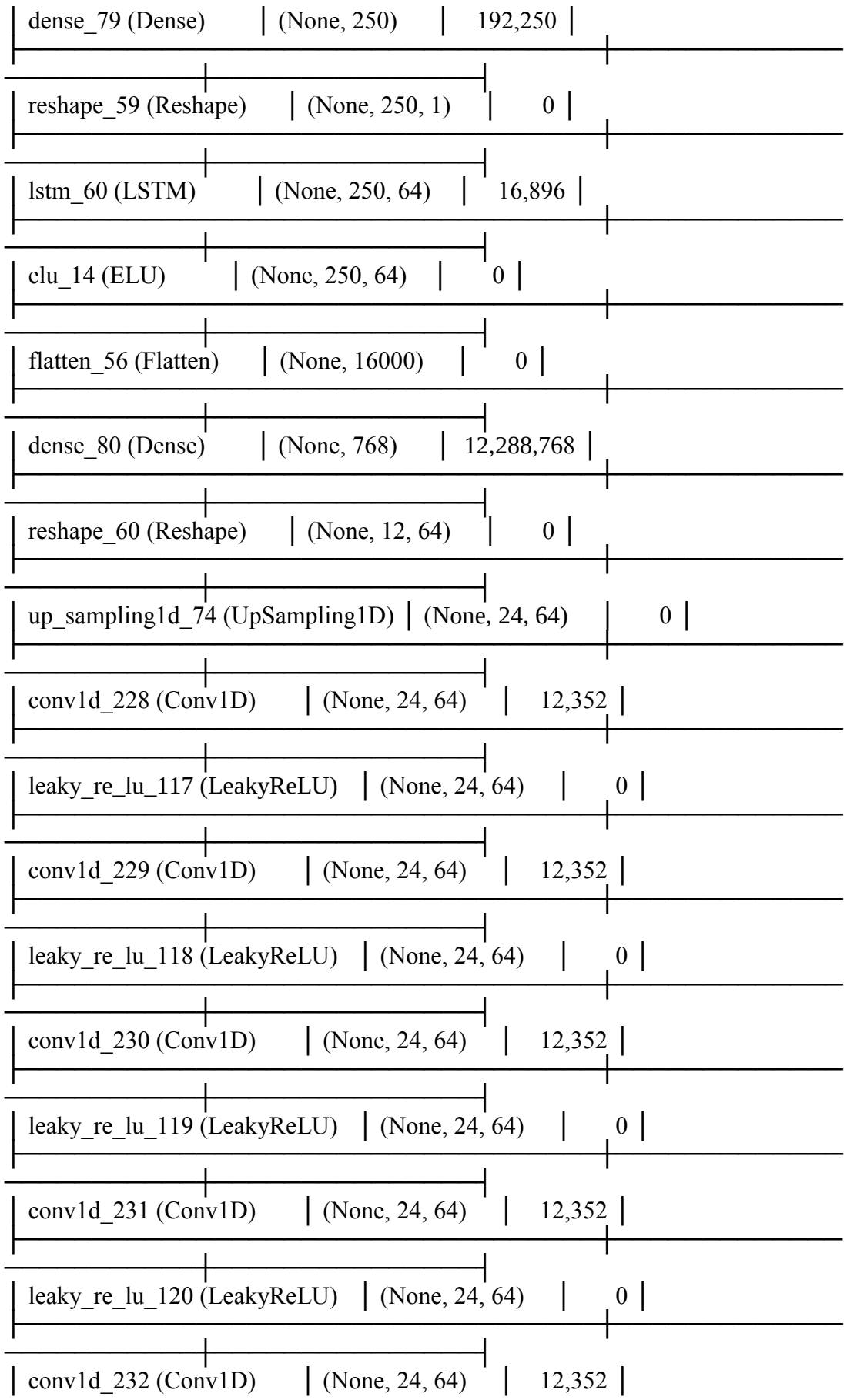
conv1d_207 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_96 (LeakyReLU)	(None, 24, 64)	0
conv1d_208 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_97 (LeakyReLU)	(None, 24, 64)	0
conv1d_209 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_98 (LeakyReLU)	(None, 24, 64)	0
conv1d_210 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_99 (LeakyReLU)	(None, 24, 64)	0
conv1d_211 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_100 (LeakyReLU)	(None, 24, 64)	0
flatten_51 (Flatten)	(None, 1536)	0
dense_75 (Dense)	(None, 250)	384,250
reshape_53 (Reshape)	(None, 25, 10)	0

Total params: 12,995,316 (49.57 MB)
Trainable params: 12,995,316 (49.57 MB)
Non-trainable params: 0 (0.00 B)

6 hidden layers of the Standard Model

Deep Autoencoder Model without Optimizer
Model: "sequential_15"

Layer (type)	Output Shape	Param #
reshape_58 (Reshape)	(None, 25, 10)	0
conv1d_222 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_111 (LeakyReLU)	(None, 25, 64)	0
conv1d_223 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_112 (LeakyReLU)	(None, 25, 64)	0
conv1d_224 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_113 (LeakyReLU)	(None, 25, 64)	0
conv1d_225 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_114 (LeakyReLU)	(None, 25, 64)	0
conv1d_226 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_115 (LeakyReLU)	(None, 25, 64)	0
conv1d_227 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_116 (LeakyReLU)	(None, 25, 64)	0
max_pooling1d_45 (MaxPooling1D)	(None, 12, 64)	0
flatten_55 (Flatten)	(None, 768)	0


dense_79 (Dense) (None, 250) 192,250
reshape_59 (Reshape) (None, 250, 1) 0
lstm_60 (LSTM) (None, 250, 64) 16,896
elu_14 (ELU) (None, 250, 64) 0
flatten_56 (Flatten) (None, 16000) 0
dense_80 (Dense) (None, 768) 12,288,768
reshape_60 (Reshape) (None, 12, 64) 0
up_sampling1d_74 (UpSampling1D) (None, 24, 64) 0
conv1d_228 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_117 (LeakyReLU) (None, 24, 64) 0
conv1d_229 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_118 (LeakyReLU) (None, 24, 64) 0
conv1d_230 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_119 (LeakyReLU) (None, 24, 64) 0
conv1d_231 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_120 (LeakyReLU) (None, 24, 64) 0
conv1d_232 (Conv1D) (None, 24, 64) 12,352

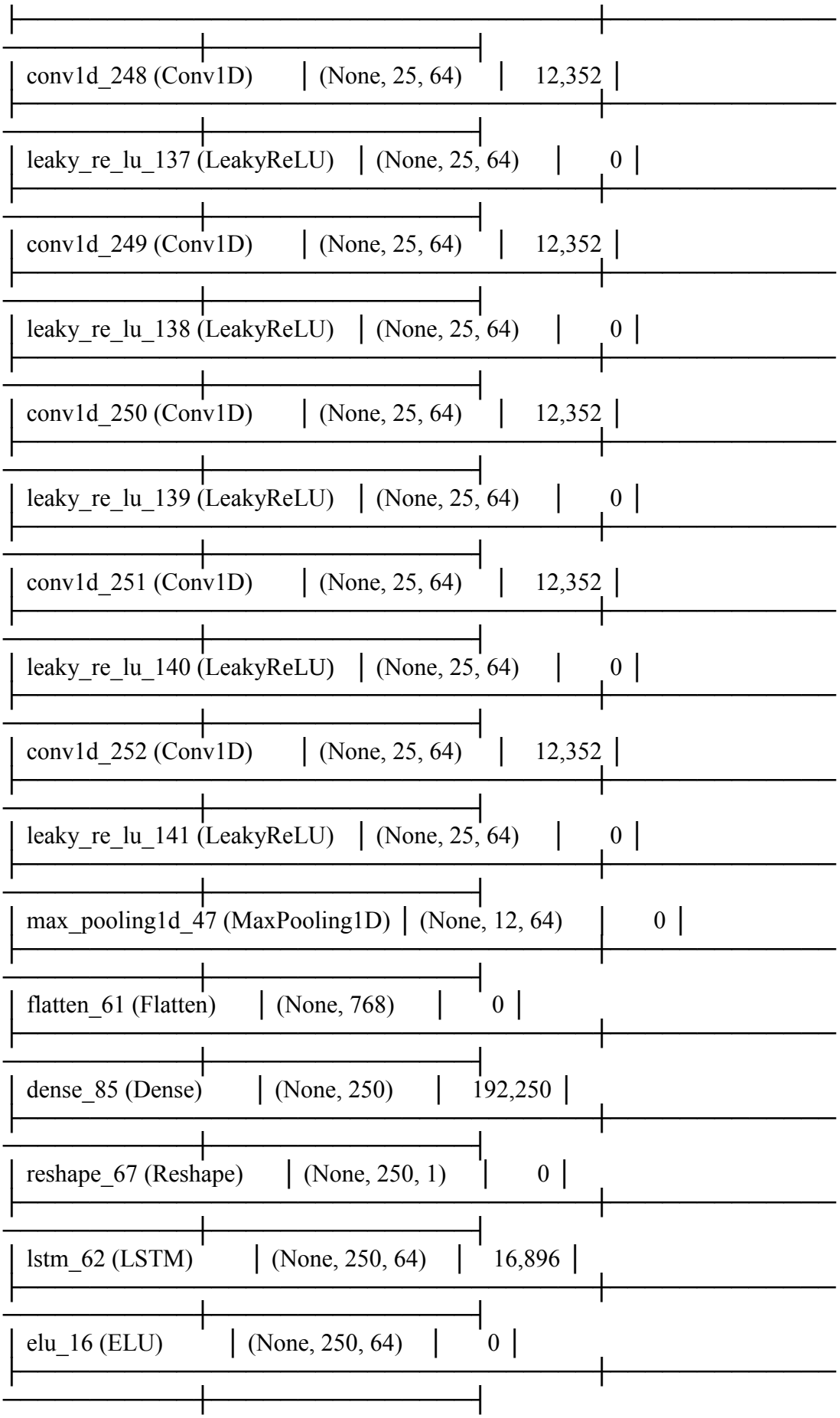
leaky_re_lu_121 (LeakyReLU)	(None, 24, 64)	0
conv1d_233 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_122 (LeakyReLU)	(None, 24, 64)	0
flatten_57 (Flatten)	(None, 1536)	0
dense_81 (Dense)	(None, 250)	384,250
reshape_61 (Reshape)	(None, 25, 10)	0

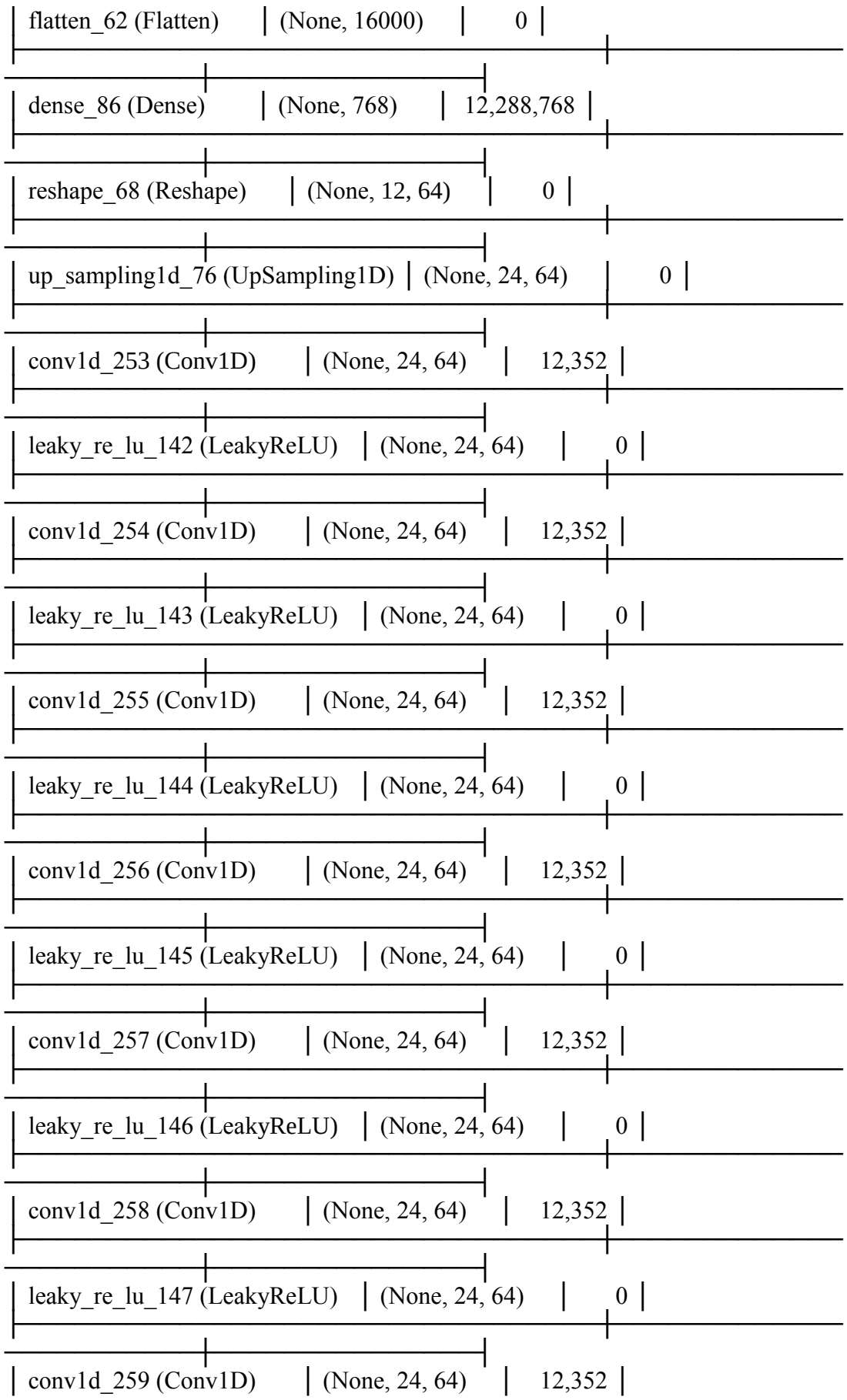
Total params: 13,020,020 (49.67 MB)
Trainable params: 13,020,020 (49.67 MB)
Non-trainable params: 0 (0.00 B)

7 hidden layers of the Standard Model

Deep Autoencoder Model without Optimizer
Model: "sequential_17"

Layer (type)	**Output Shape**	**Param #**
reshape_66 (Reshape)	(None, 25, 10)	0
conv1d_246 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_135 (LeakyReLU)	(None, 25, 64)	0
conv1d_247 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_136 (LeakyReLU)	(None, 25, 64)	0


conv1d_248 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_137 (LeakyReLU) (None, 25, 64) 0
conv1d_249 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_138 (LeakyReLU) (None, 25, 64) 0
conv1d_250 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_139 (LeakyReLU) (None, 25, 64) 0
conv1d_251 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_140 (LeakyReLU) (None, 25, 64) 0
conv1d_252 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_141 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_47 (MaxPooling1D) (None, 12, 64) 0
flatten_61 (Flatten) (None, 768) 0
dense_85 (Dense) (None, 250) 192,250
reshape_67 (Reshape) (None, 250, 1) 0
lstm_62 (LSTM) (None, 250, 64) 16,896
elu_16 (ELU) (None, 250, 64) 0


flatten_62 (Flatten) (None, 16000) 0
dense_86 (Dense) (None, 768) 12,288,768
reshape_68 (Reshape) (None, 12, 64) 0
up_sampling1d_76 (UpSampling1D) (None, 24, 64) 0
conv1d_253 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_142 (LeakyReLU) (None, 24, 64) 0
conv1d_254 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_143 (LeakyReLU) (None, 24, 64) 0
conv1d_255 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_144 (LeakyReLU) (None, 24, 64) 0
conv1d_256 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_145 (LeakyReLU) (None, 24, 64) 0
conv1d_257 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_146 (LeakyReLU) (None, 24, 64) 0
conv1d_258 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_147 (LeakyReLU) (None, 24, 64) 0
conv1d_259 (Conv1D) (None, 24, 64) 12,352

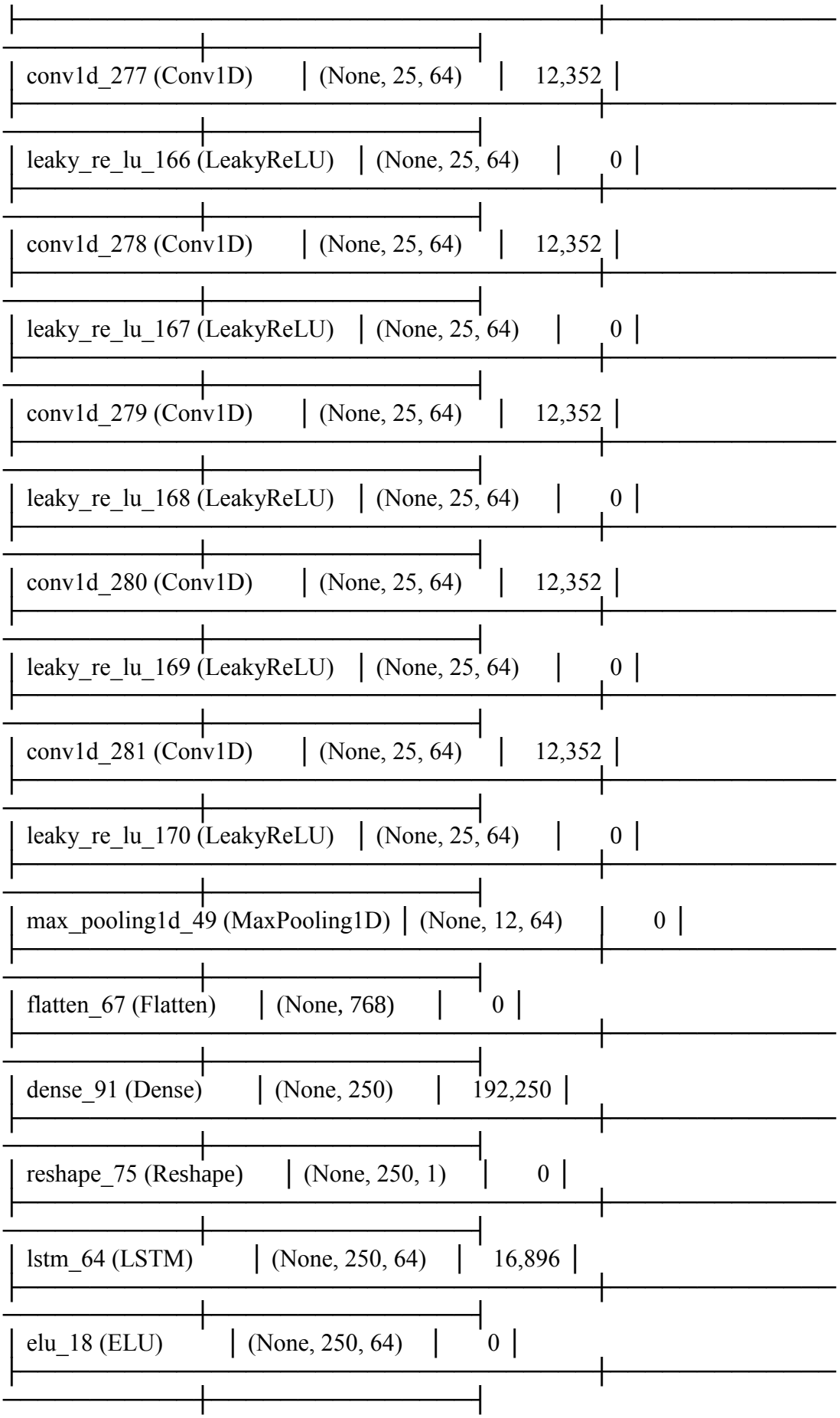
leaky_re_lu_148 (LeakyReLU)	(None, 24, 64)	0
flatten_63 (Flatten)	(None, 1536)	0
dense_87 (Dense)	(None, 250)	384,250
reshape_69 (Reshape)	(None, 25, 10)	0

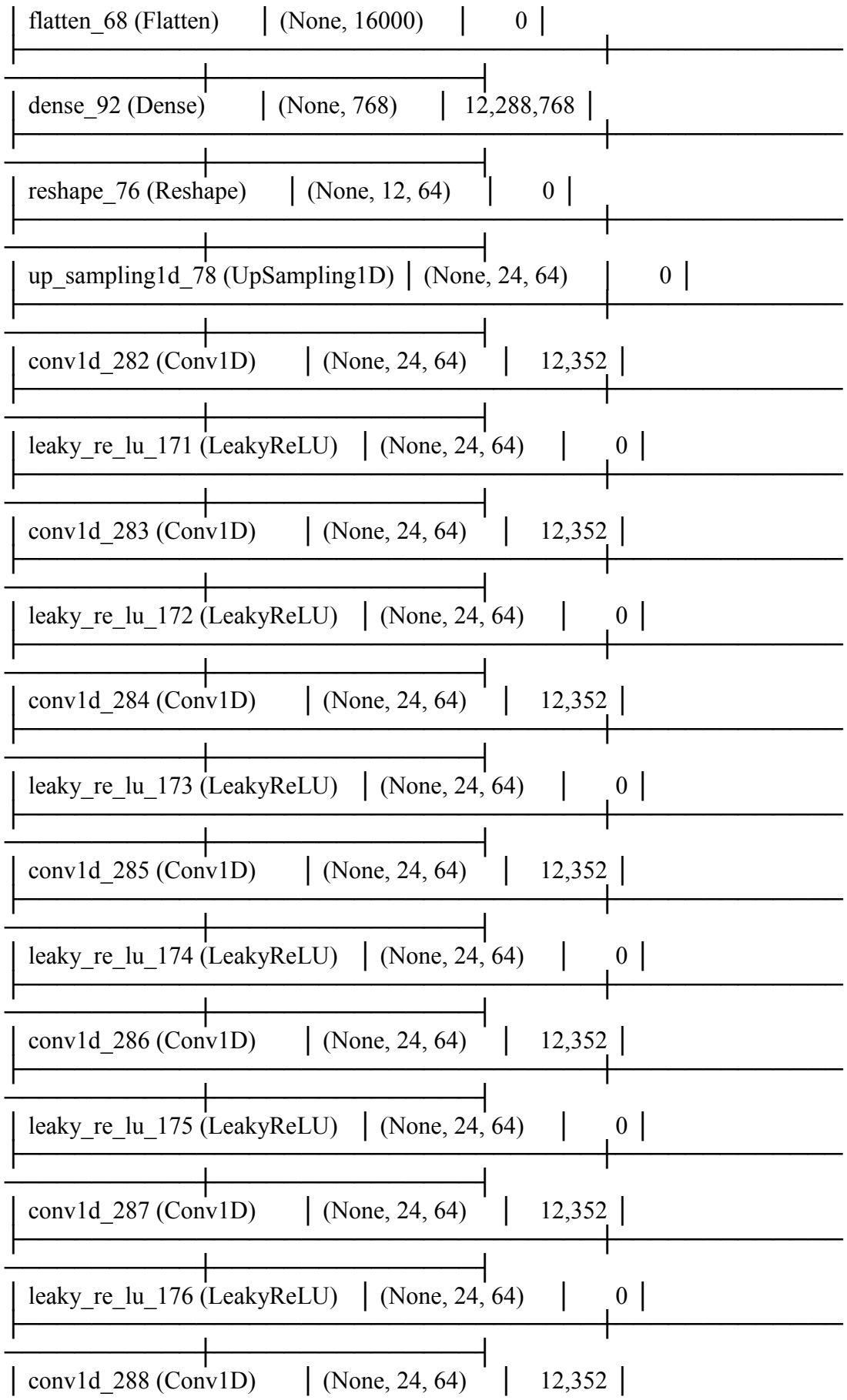
Total params: 13,044,724 (49.76 MB)
Trainable params: 13,044,724 (49.76 MB)
Non-trainable params: 0 (0.00 B)

8 hidden layers of the Standard Model

Deep Autoencoder Model without Optimizer
Model: "sequential_19"

Layer (type)	**Output Shape**	**Param #**
reshape_74 (Reshape)	(None, 25, 10)	0
conv1d_274 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_163 (LeakyReLU)	(None, 25, 64)	0
conv1d_275 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_164 (LeakyReLU)	(None, 25, 64)	0
conv1d_276 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_165 (LeakyReLU)	(None, 25, 64)	0


conv1d_277 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_166 (LeakyReLU) (None, 25, 64) 0
conv1d_278 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_167 (LeakyReLU) (None, 25, 64) 0
conv1d_279 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_168 (LeakyReLU) (None, 25, 64) 0
conv1d_280 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_169 (LeakyReLU) (None, 25, 64) 0
conv1d_281 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_170 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_49 (MaxPooling1D) (None, 12, 64) 0
flatten_67 (Flatten) (None, 768) 0
dense_91 (Dense) (None, 250) 192,250
reshape_75 (Reshape) (None, 250, 1) 0
lstm_64 (LSTM) (None, 250, 64) 16,896
elu_18 (ELU) (None, 250, 64) 0


flatten_68 (Flatten) (None, 16000) 0
dense_92 (Dense) (None, 768) 12,288,768
reshape_76 (Reshape) (None, 12, 64) 0
up_sampling1d_78 (UpSampling1D) (None, 24, 64) 0
conv1d_282 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_171 (LeakyReLU) (None, 24, 64) 0
conv1d_283 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_172 (LeakyReLU) (None, 24, 64) 0
conv1d_284 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_173 (LeakyReLU) (None, 24, 64) 0
conv1d_285 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_174 (LeakyReLU) (None, 24, 64) 0
conv1d_286 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_175 (LeakyReLU) (None, 24, 64) 0
conv1d_287 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_176 (LeakyReLU) (None, 24, 64) 0
conv1d_288 (Conv1D) (None, 24, 64) 12,352

leaky_re_lu_177 (LeakyReLU)	(None, 24, 64)	0
conv1d_289 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_178 (LeakyReLU)	(None, 24, 64)	0
flatten_69 (Flatten)	(None, 1536)	0
dense_93 (Dense)	(None, 250)	384,250
reshape_77 (Reshape)	(None, 25, 10)	0

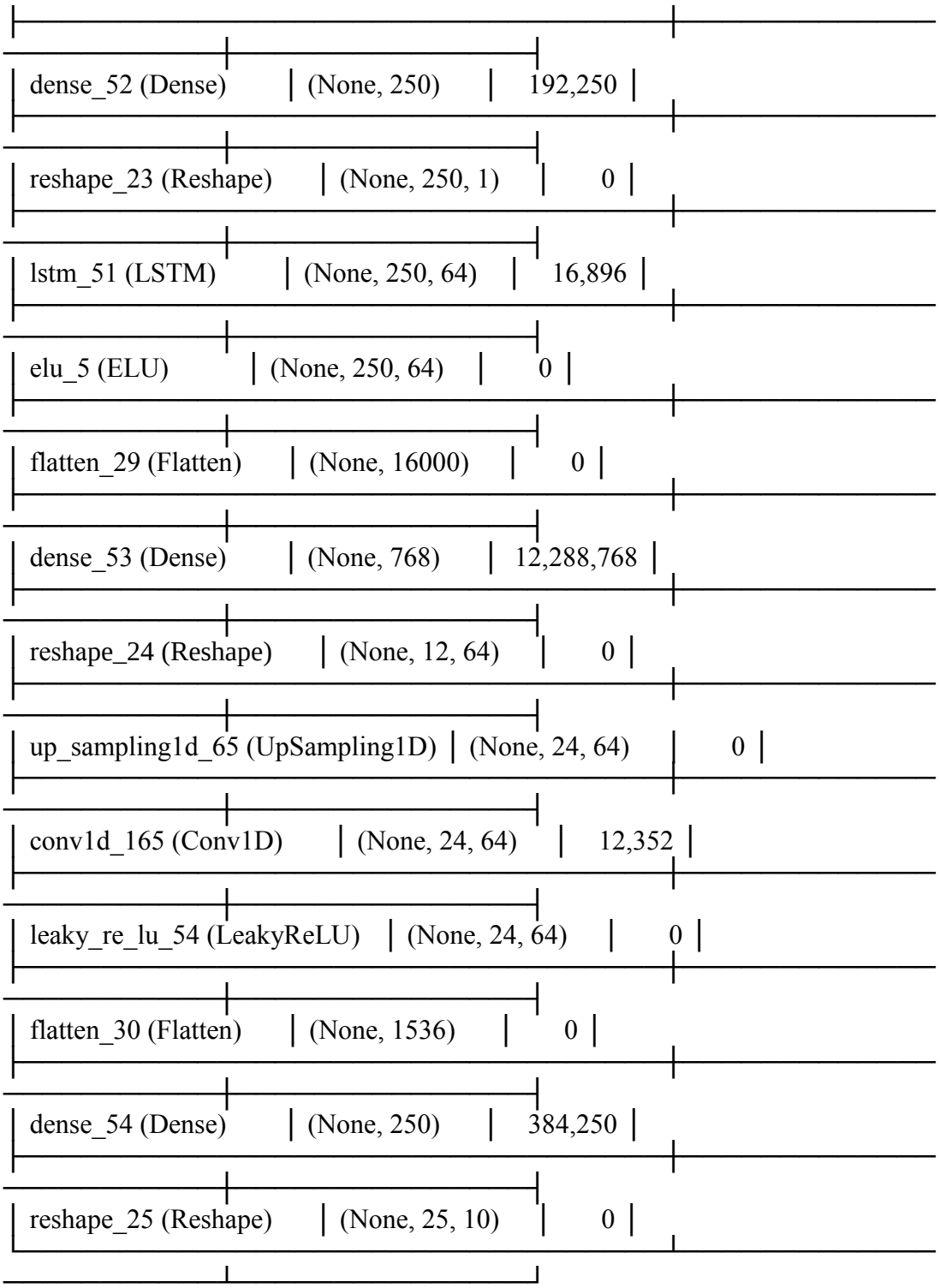
Total params: 13,069,428 (49.86 MB)
Trainable params: 13,069,428 (49.86 MB)
Non-trainable params: 0 (0.00 B)

Appendix 3: Nadam-Optimized Hybrid Model Architectures

1 hidden layer of Nadam-Optimized Model

Deep Autoencoder Model with Nadam Optimizer
Model: "sequential_6"

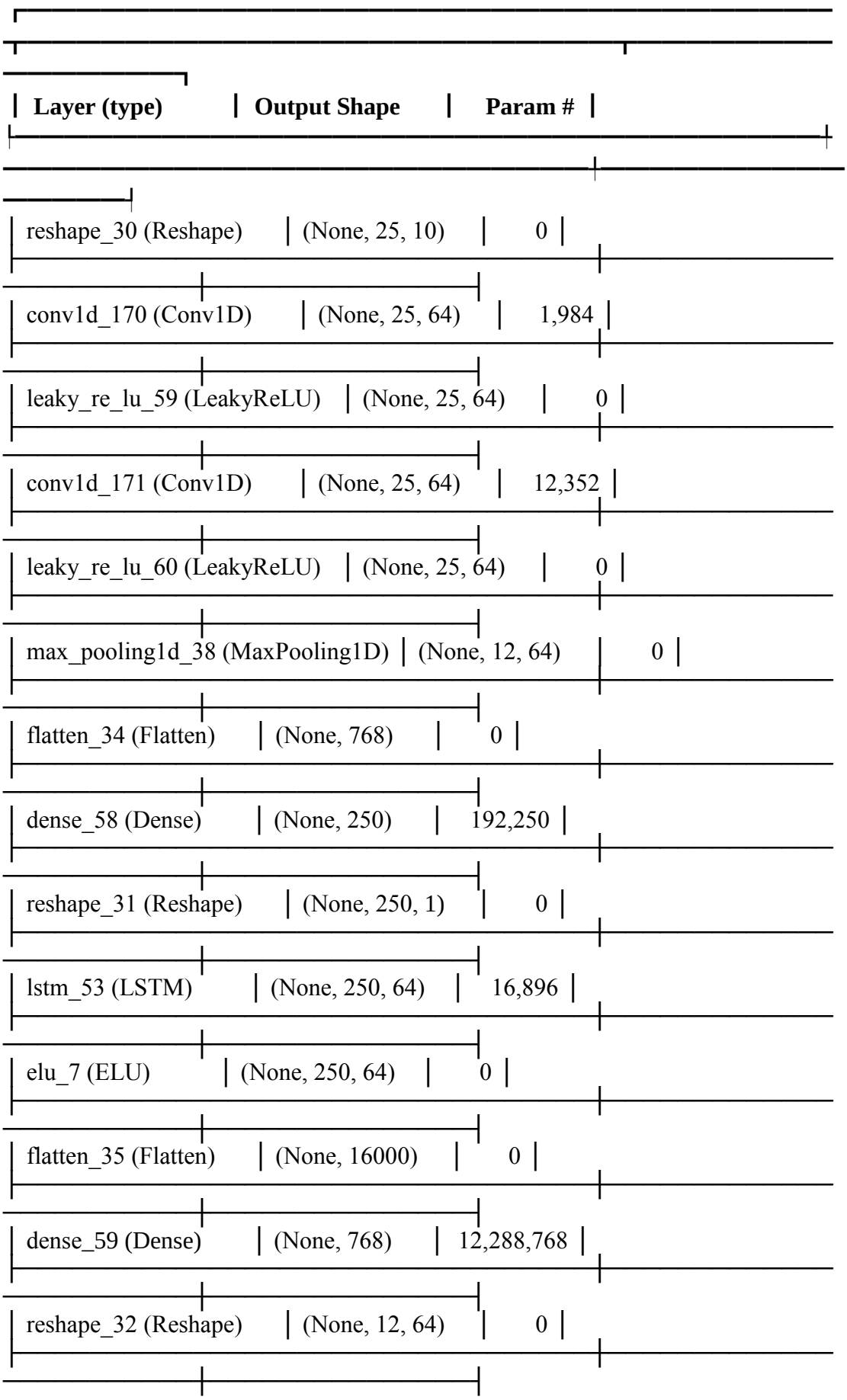
Layer (type)	Output Shape	Param #
reshape_22 (Reshape)	(None, 25, 10)	0
conv1d_164 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_53 (LeakyReLU)	(None, 25, 64)	0
max_pooling1d_36 (MaxPooling1D)	(None, 12, 64)	0
flatten_28 (Flatten)	(None, 768)	0



Total params: 12,896,500 (49.20 MB)
Trainable params: 12,896,500 (49.20 MB)
Non-trainable params: 0 (0.00 B)

2 hidden layers of Nadam-Optimized Model

Deep Autoencoder Model with Nadam Optimizer
Model: "sequential_8"



Layer (type)	Output Shape	Param #
reshape_30 (Reshape)	(None, 25, 10)	0
conv1d_170 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_59 (LeakyReLU)	(None, 25, 64)	0
conv1d_171 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_60 (LeakyReLU)	(None, 25, 64)	0
max_pooling1d_38 (MaxPooling1D)	(None, 12, 64)	0
flatten_34 (Flatten)	(None, 768)	0
dense_58 (Dense)	(None, 250)	192,250
reshape_31 (Reshape)	(None, 250, 1)	0
lstm_53 (LSTM)	(None, 250, 64)	16,896
elu_7 (ELU)	(None, 250, 64)	0
flatten_35 (Flatten)	(None, 16000)	0
dense_59 (Dense)	(None, 768)	12,288,768
reshape_32 (Reshape)	(None, 12, 64)	0

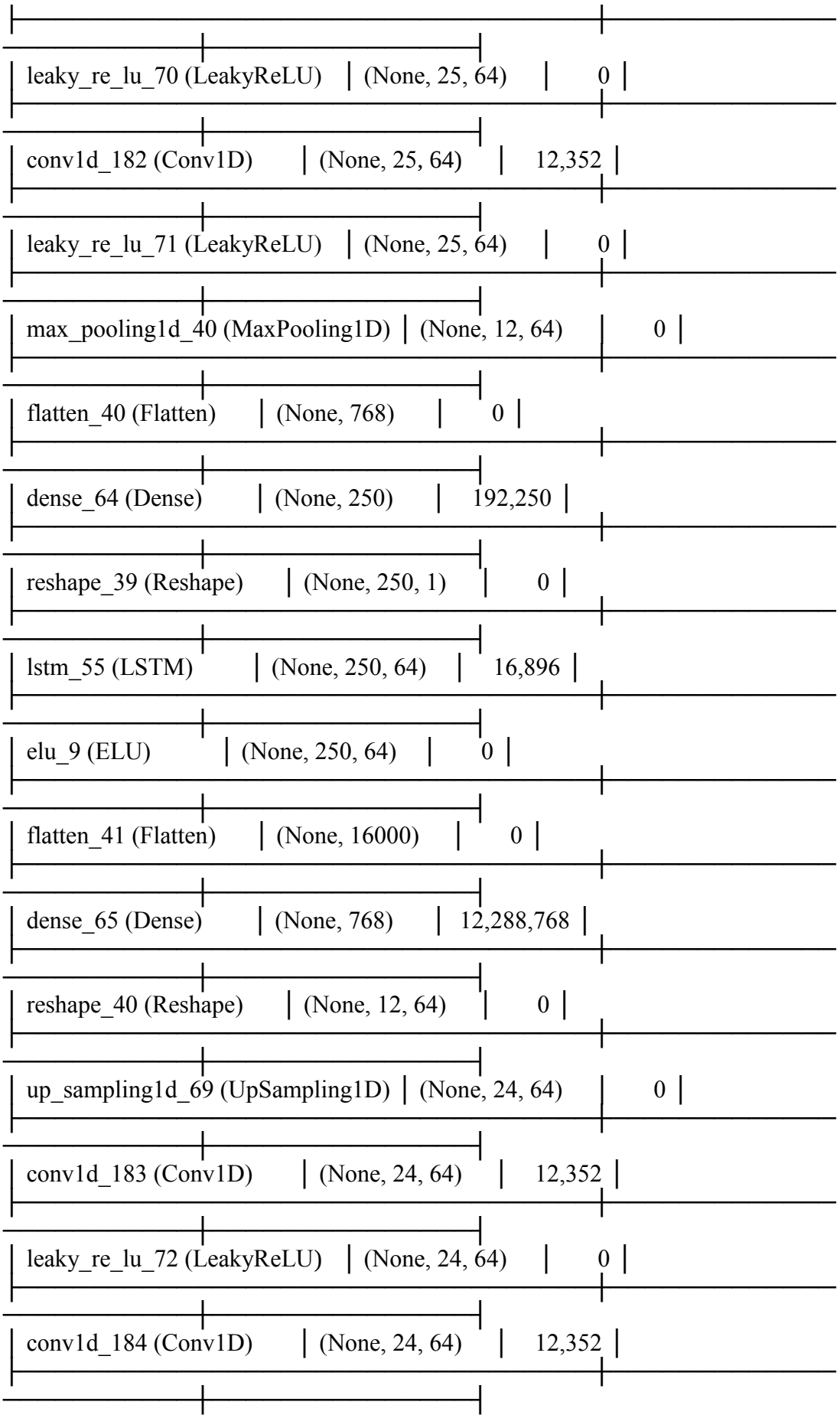
up_sampling1d_67 (UpSampling1D)	(None, 24, 64)	0
conv1d_172 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_61 (LeakyReLU)	(None, 24, 64)	0
conv1d_173 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_62 (LeakyReLU)	(None, 24, 64)	0
flatten_36 (Flatten)	(None, 1536)	0
dense_60 (Dense)	(None, 250)	384,250
reshape_33 (Reshape)	(None, 25, 10)	0

Total params: 12,921,204 (49.29 MB)
Trainable params: 12,921,204 (49.29 MB)
Non-trainable params: 0 (0.00 B)

3 hidden layers of Nadam-Optimized Model

Deep Autoencoder Model with Nadam Optimizer
Model: "sequential_10"

Layer (type)	Output Shape	Param #
reshape_38 (Reshape)	(None, 25, 10)	0
conv1d_180 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_69 (LeakyReLU)	(None, 25, 64)	0
conv1d_181 (Conv1D)	(None, 25, 64)	12,352


leaky_re_lu_70 (LeakyReLU) (None, 25, 64) 0
conv1d_182 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_71 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_40 (MaxPooling1D) (None, 12, 64) 0
flatten_40 (Flatten) (None, 768) 0
dense_64 (Dense) (None, 250) 192,250
reshape_39 (Reshape) (None, 250, 1) 0
lstm_55 (LSTM) (None, 250, 64) 16,896
elu_9 (ELU) (None, 250, 64) 0
flatten_41 (Flatten) (None, 16000) 0
dense_65 (Dense) (None, 768) 12,288,768
reshape_40 (Reshape) (None, 12, 64) 0
up_sampling1d_69 (UpSampling1D) (None, 24, 64) 0
conv1d_183 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_72 (LeakyReLU) (None, 24, 64) 0
conv1d_184 (Conv1D) (None, 24, 64) 12,352

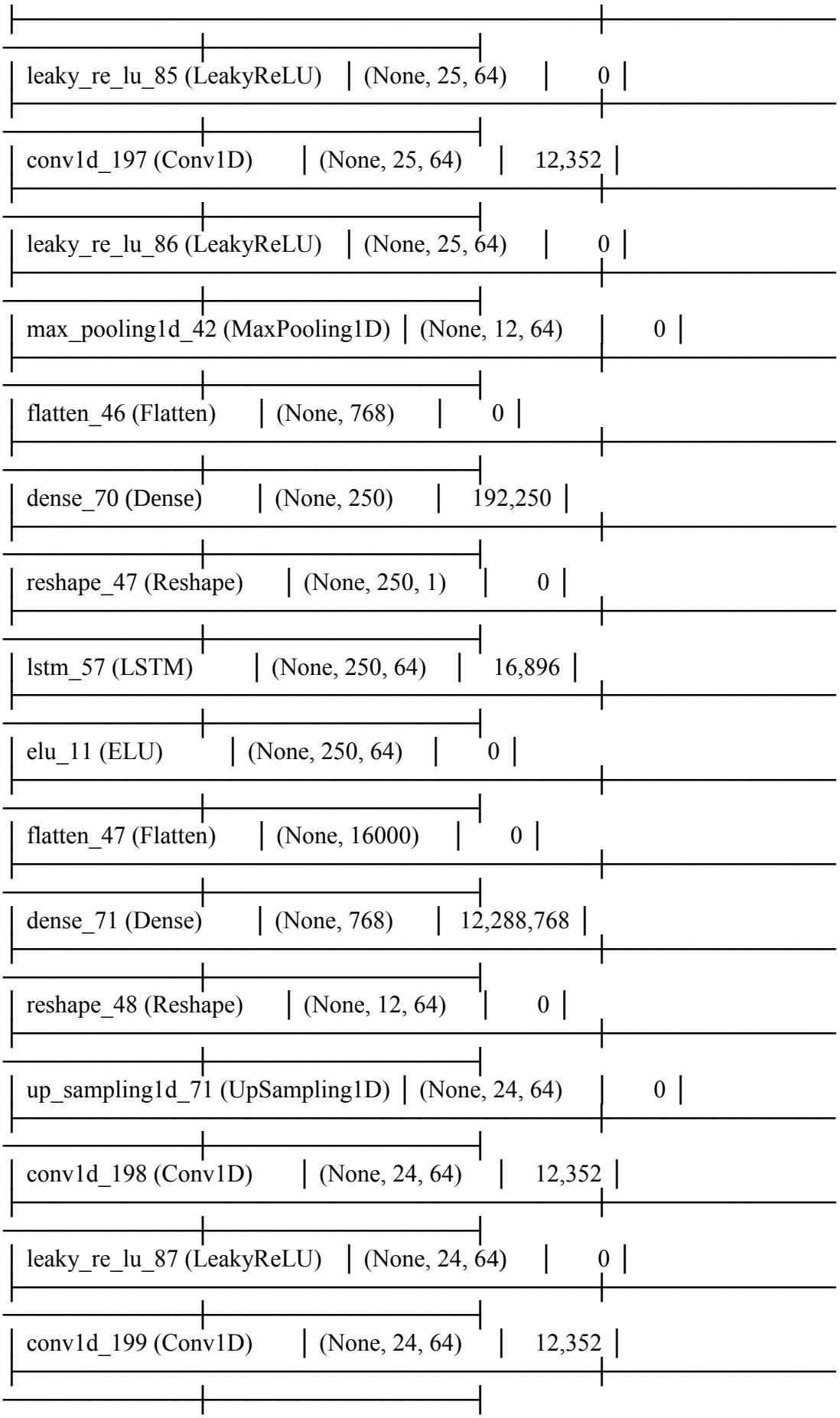
leaky_re_lu_73 (LeakyReLU)	(None, 24, 64)	0
conv1d_185 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_74 (LeakyReLU)	(None, 24, 64)	0
flatten_42 (Flatten)	(None, 1536)	0
dense_66 (Dense)	(None, 250)	384,250
reshape_41 (Reshape)	(None, 25, 10)	0

Total params: 12,945,908 (49.38 MB)
Trainable params: 12,945,908 (49.38 MB)
Non-trainable params: 0 (0.00 B)

4 hidden layers of Nadam-Optimized Model

Deep Autoencoder Model with Nadam Optimizer
Model: "sequential_12"

Layer (type)	**Output Shape**	**Param #**
reshape_46 (Reshape)	(None, 25, 10)	0
conv1d_194 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_83 (LeakyReLU)	(None, 25, 64)	0
conv1d_195 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_84 (LeakyReLU)	(None, 25, 64)	0
conv1d_196 (Conv1D)	(None, 25, 64)	12,352


leaky_re_lu_85 (LeakyReLU) (None, 25, 64) 0
conv1d_197 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_86 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_42 (MaxPooling1D) (None, 12, 64) 0
flatten_46 (Flatten) (None, 768) 0
dense_70 (Dense) (None, 250) 192,250
reshape_47 (Reshape) (None, 250, 1) 0
lstm_57 (LSTM) (None, 250, 64) 16,896
elu_11 (ELU) (None, 250, 64) 0
flatten_47 (Flatten) (None, 16000) 0
dense_71 (Dense) (None, 768) 12,288,768
reshape_48 (Reshape) (None, 12, 64) 0
up_sampling1d_71 (UpSampling1D) (None, 24, 64) 0
conv1d_198 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_87 (LeakyReLU) (None, 24, 64) 0
conv1d_199 (Conv1D) (None, 24, 64) 12,352

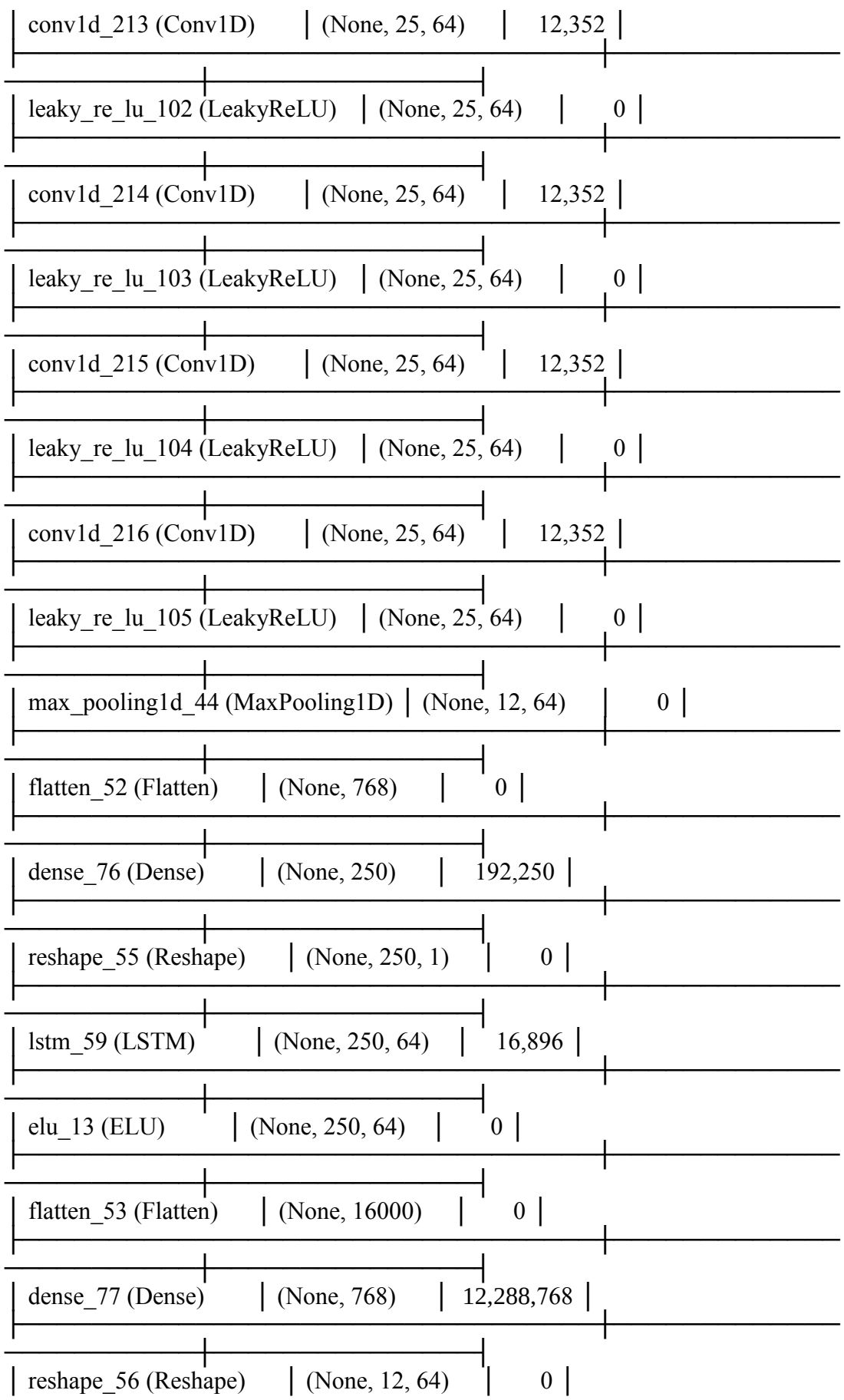
leaky_re_lu_88 (LeakyReLU)	(None, 24, 64)	0
conv1d_200 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_89 (LeakyReLU)	(None, 24, 64)	0
conv1d_201 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_90 (LeakyReLU)	(None, 24, 64)	0
flatten_48 (Flatten)	(None, 1536)	0
dense_72 (Dense)	(None, 250)	384,250
reshape_49 (Reshape)	(None, 25, 10)	0

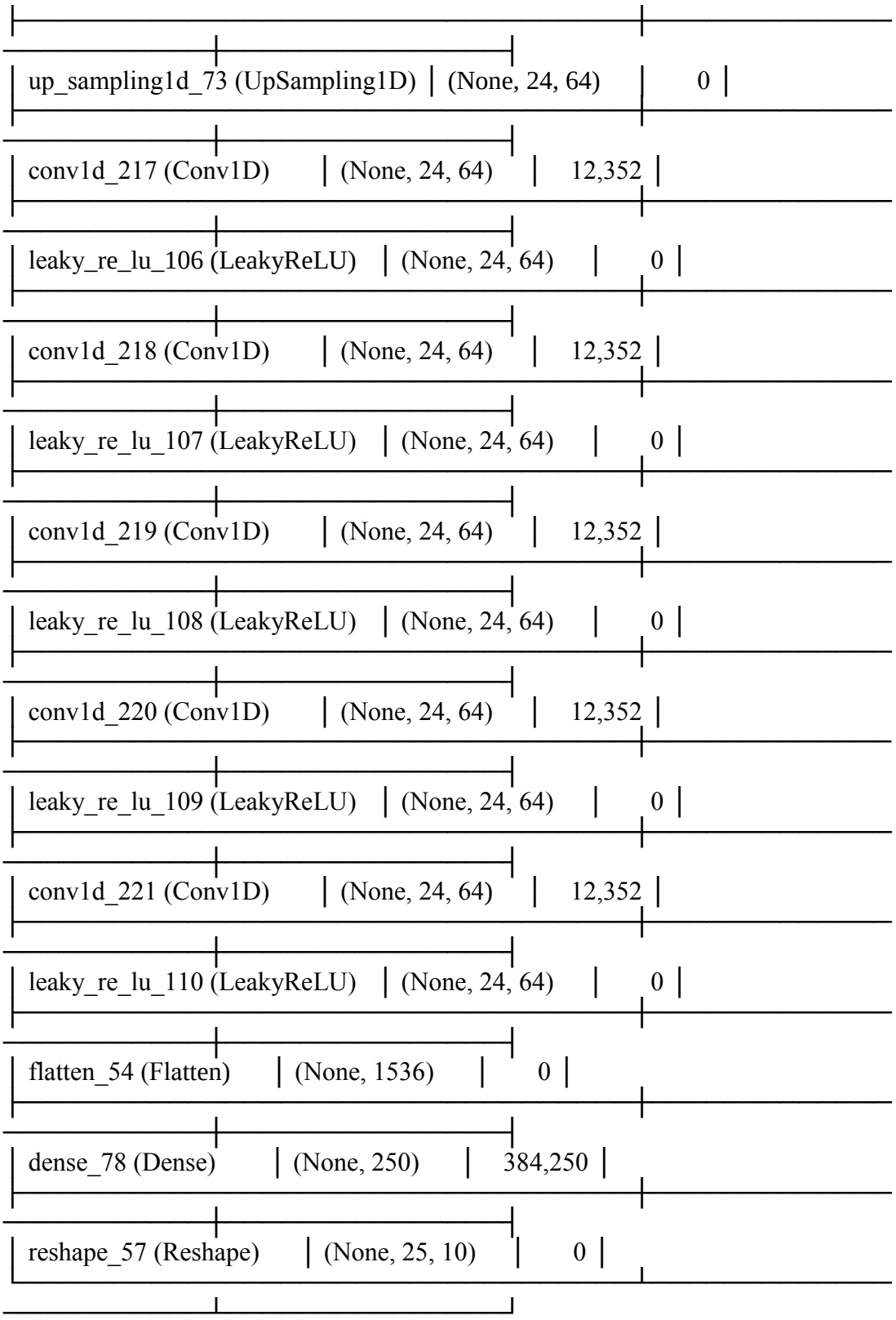
Total params: 12,970,612 (49.48 MB)
Trainable params: 12,970,612 (49.48 MB)
Non-trainable params: 0 (0.00 B)

5 hidden layers of Nadam-Optimized Model

Deep Autoencoder Model with Nadam Optimizer
Model: "sequential_14"

Layer (type)	**Output Shape**	**Param #**
reshape_54 (Reshape)	(None, 25, 10)	0
conv1d_212 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_101 (LeakyReLU)	(None, 25, 64)	0


conv1d_213 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_102 (LeakyReLU) (None, 25, 64) 0
conv1d_214 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_103 (LeakyReLU) (None, 25, 64) 0
conv1d_215 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_104 (LeakyReLU) (None, 25, 64) 0
conv1d_216 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_105 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_44 (MaxPooling1D) (None, 12, 64) 0
flatten_52 (Flatten) (None, 768) 0
dense_76 (Dense) (None, 250) 192,250
reshape_55 (Reshape) (None, 250, 1) 0
lstm_59 (LSTM) (None, 250, 64) 16,896
elu_13 (ELU) (None, 250, 64) 0
flatten_53 (Flatten) (None, 16000) 0
dense_77 (Dense) (None, 768) 12,288,768
reshape_56 (Reshape) (None, 12, 64) 0



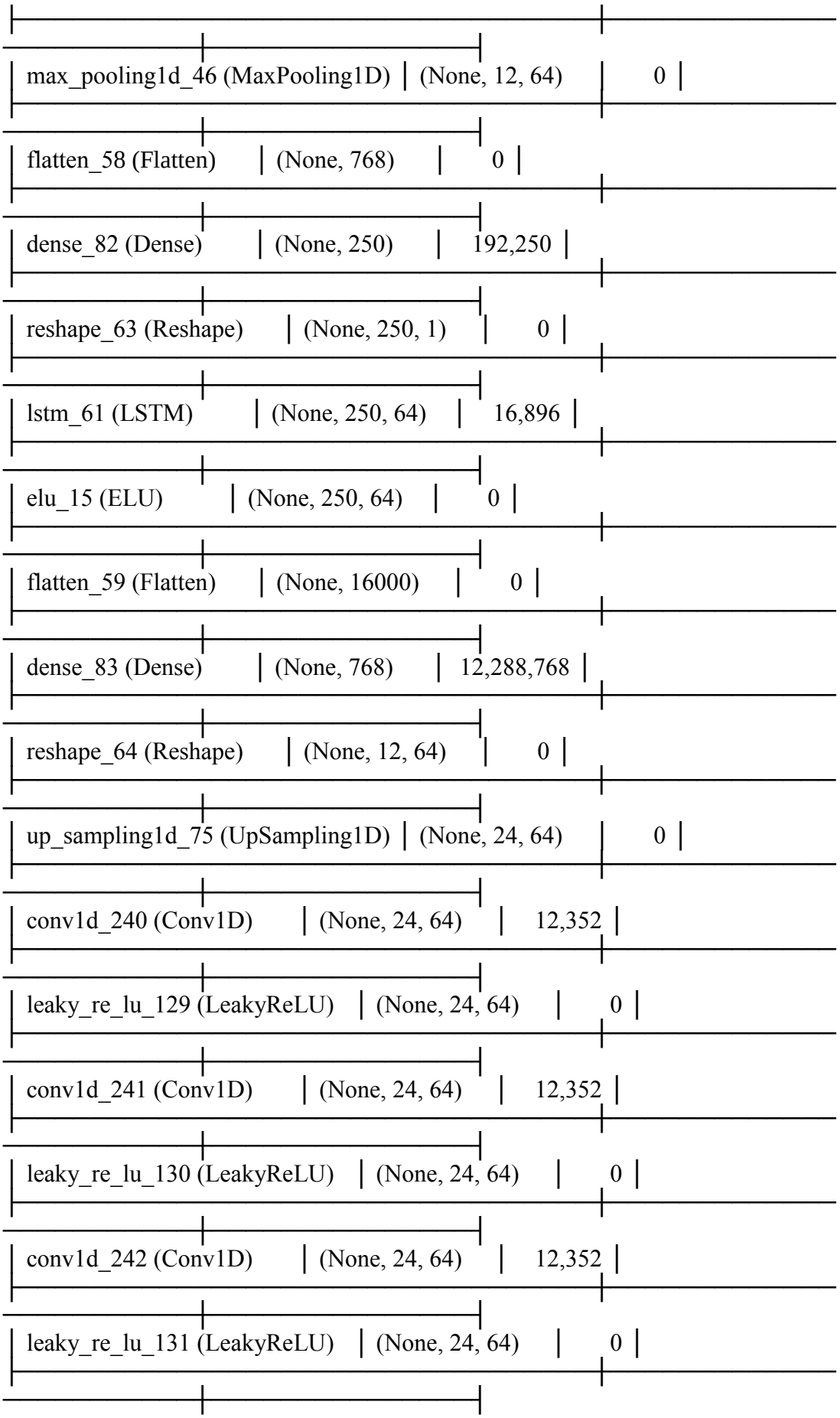
Total params: 12,995,316 (49.57 MB)
Trainable params: 12,995,316 (49.57 MB)
Non-trainable params: 0 (0.00 B)

6 hidden layers of Nadam-Optimized Model

Deep Autoencoder Model with Nadam Optimizer

Model: "sequential_16"

Layer (type)	Output Shape	Param #
reshape_62 (Reshape)	(None, 25, 10)	0
conv1d_234 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_123 (LeakyReLU)	(None, 25, 64)	0
conv1d_235 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_124 (LeakyReLU)	(None, 25, 64)	0
conv1d_236 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_125 (LeakyReLU)	(None, 25, 64)	0
conv1d_237 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_126 (LeakyReLU)	(None, 25, 64)	0
conv1d_238 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_127 (LeakyReLU)	(None, 25, 64)	0
conv1d_239 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_128 (LeakyReLU)	(None, 25, 64)	0


max_pooling1d_46 (MaxPooling1D) (None, 12, 64) 0
flatten_58 (Flatten) (None, 768) 0
dense_82 (Dense) (None, 250) 192,250
reshape_63 (Reshape) (None, 250, 1) 0
lstm_61 (LSTM) (None, 250, 64) 16,896
elu_15 (ELU) (None, 250, 64) 0
flatten_59 (Flatten) (None, 16000) 0
dense_83 (Dense) (None, 768) 12,288,768
reshape_64 (Reshape) (None, 12, 64) 0
up_sampling1d_75 (UpSampling1D) (None, 24, 64) 0
conv1d_240 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_129 (LeakyReLU) (None, 24, 64) 0
conv1d_241 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_130 (LeakyReLU) (None, 24, 64) 0
conv1d_242 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_131 (LeakyReLU) (None, 24, 64) 0

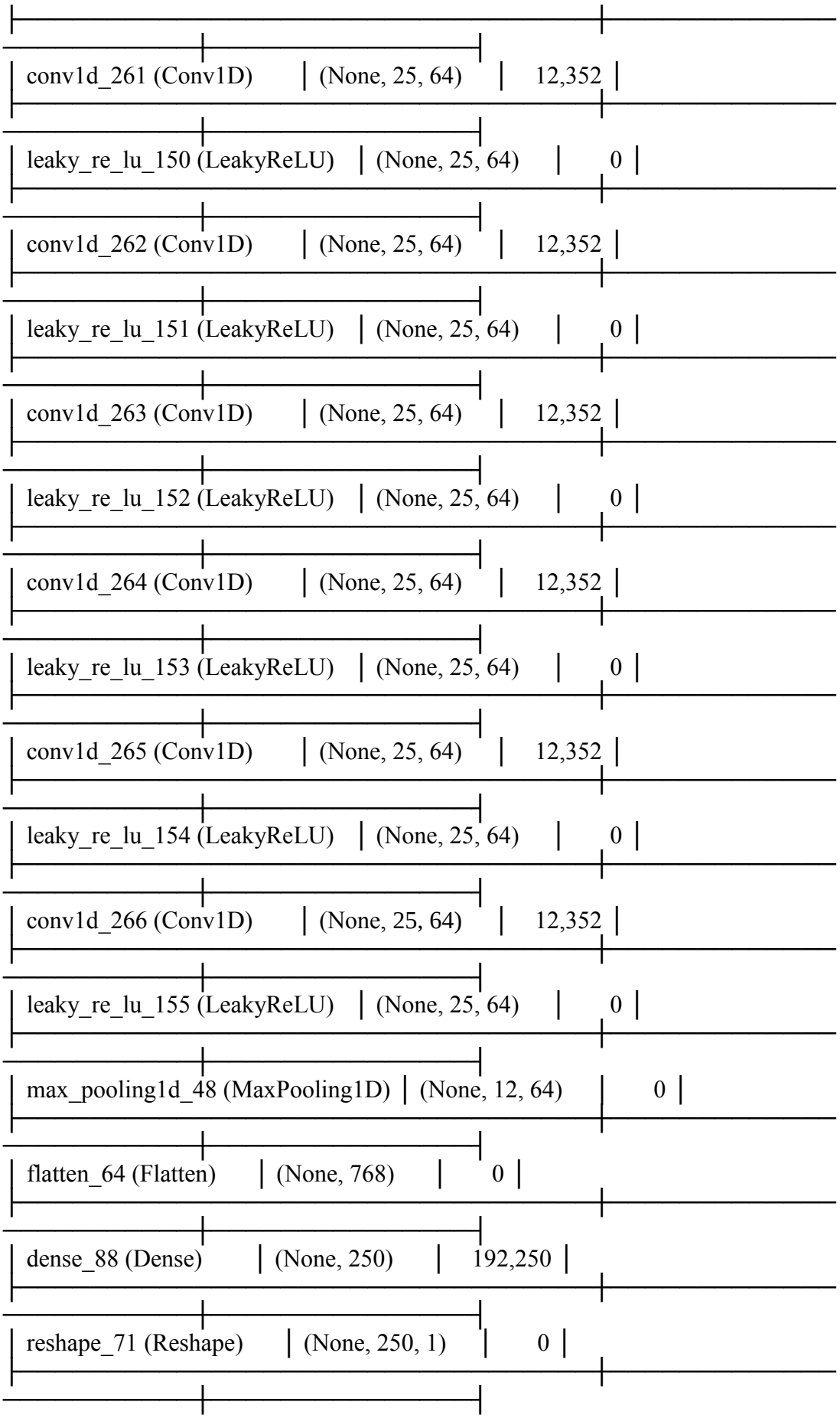
conv1d_243 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_132 (LeakyReLU)	(None, 24, 64)	0
conv1d_244 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_133 (LeakyReLU)	(None, 24, 64)	0
conv1d_245 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_134 (LeakyReLU)	(None, 24, 64)	0
flatten_60 (Flatten)	(None, 1536)	0
dense_84 (Dense)	(None, 250)	384,250
reshape_65 (Reshape)	(None, 25, 10)	0

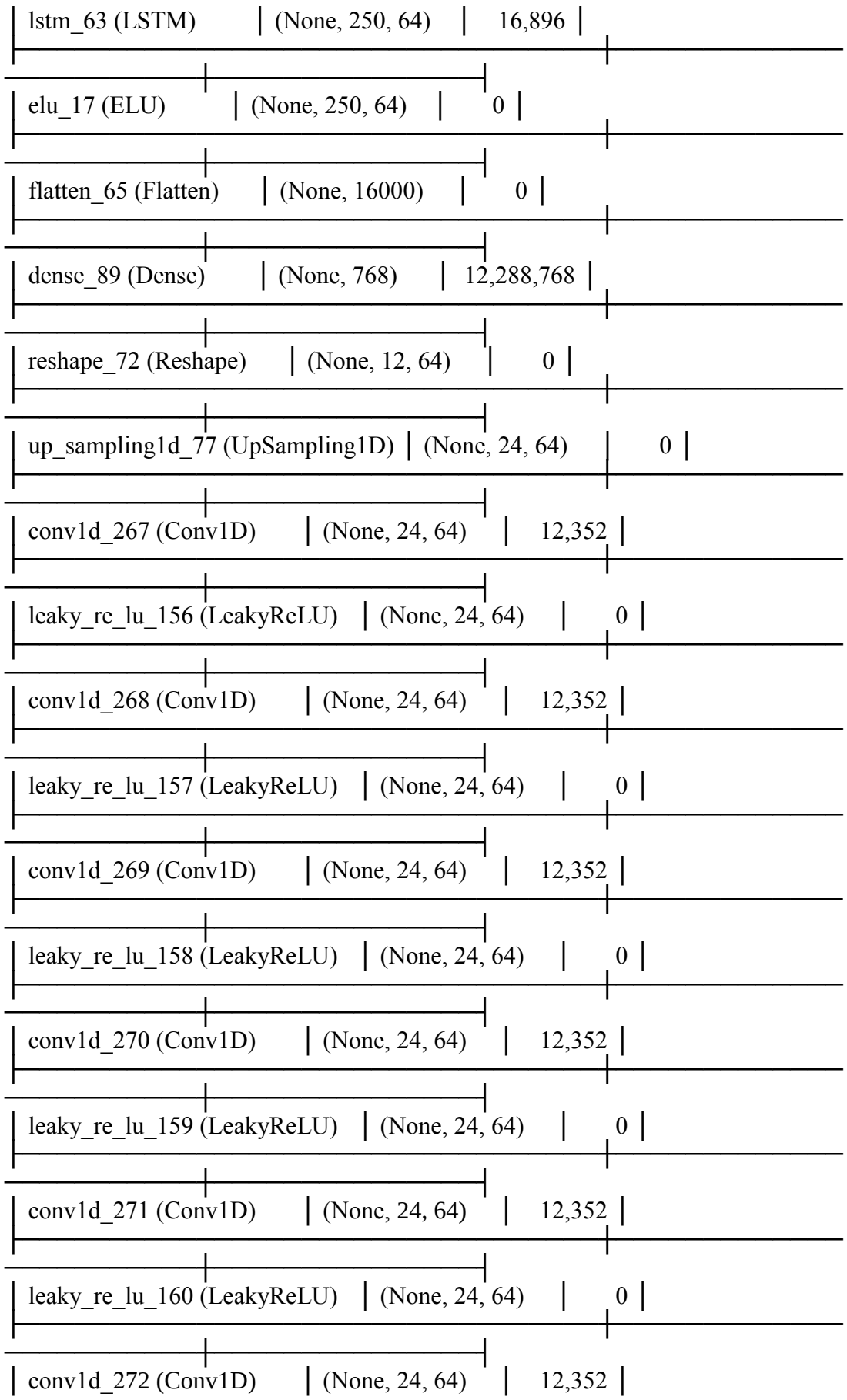
Total params: 13,020,020 (49.67 MB)
Trainable params: 13,020,020 (49.67 MB)
Non-trainable params: 0 (0.00 B)

7 hidden layers of Nadam-Optimized Model

Deep Autoencoder Model with Nadam Optimizer
Model: "sequential_18"

Layer (type)	**Output Shape**	**Param #**
reshape_70 (Reshape)	(None, 25, 10)	0
conv1d_260 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_149 (LeakyReLU)	(None, 25, 64)	0


conv1d_261 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_150 (LeakyReLU) (None, 25, 64) 0
conv1d_262 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_151 (LeakyReLU) (None, 25, 64) 0
conv1d_263 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_152 (LeakyReLU) (None, 25, 64) 0
conv1d_264 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_153 (LeakyReLU) (None, 25, 64) 0
conv1d_265 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_154 (LeakyReLU) (None, 25, 64) 0
conv1d_266 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_155 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_48 (MaxPooling1D) (None, 12, 64) 0
flatten_64 (Flatten) (None, 768) 0
dense_88 (Dense) (None, 250) 192,250
reshape_71 (Reshape) (None, 250, 1) 0


lstm_63 (LSTM) (None, 250, 64) 16,896
elu_17 (ELU) (None, 250, 64) 0
flatten_65 (Flatten) (None, 16000) 0
dense_89 (Dense) (None, 768) 12,288,768
reshape_72 (Reshape) (None, 12, 64) 0
up_sampling1d_77 (UpSampling1D) (None, 24, 64) 0
conv1d_267 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_156 (LeakyReLU) (None, 24, 64) 0
conv1d_268 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_157 (LeakyReLU) (None, 24, 64) 0
conv1d_269 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_158 (LeakyReLU) (None, 24, 64) 0
conv1d_270 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_159 (LeakyReLU) (None, 24, 64) 0
conv1d_271 (Conv1D) (None, 24, 64) 12,352
leaky_re_lu_160 (LeakyReLU) (None, 24, 64) 0
conv1d_272 (Conv1D) (None, 24, 64) 12,352

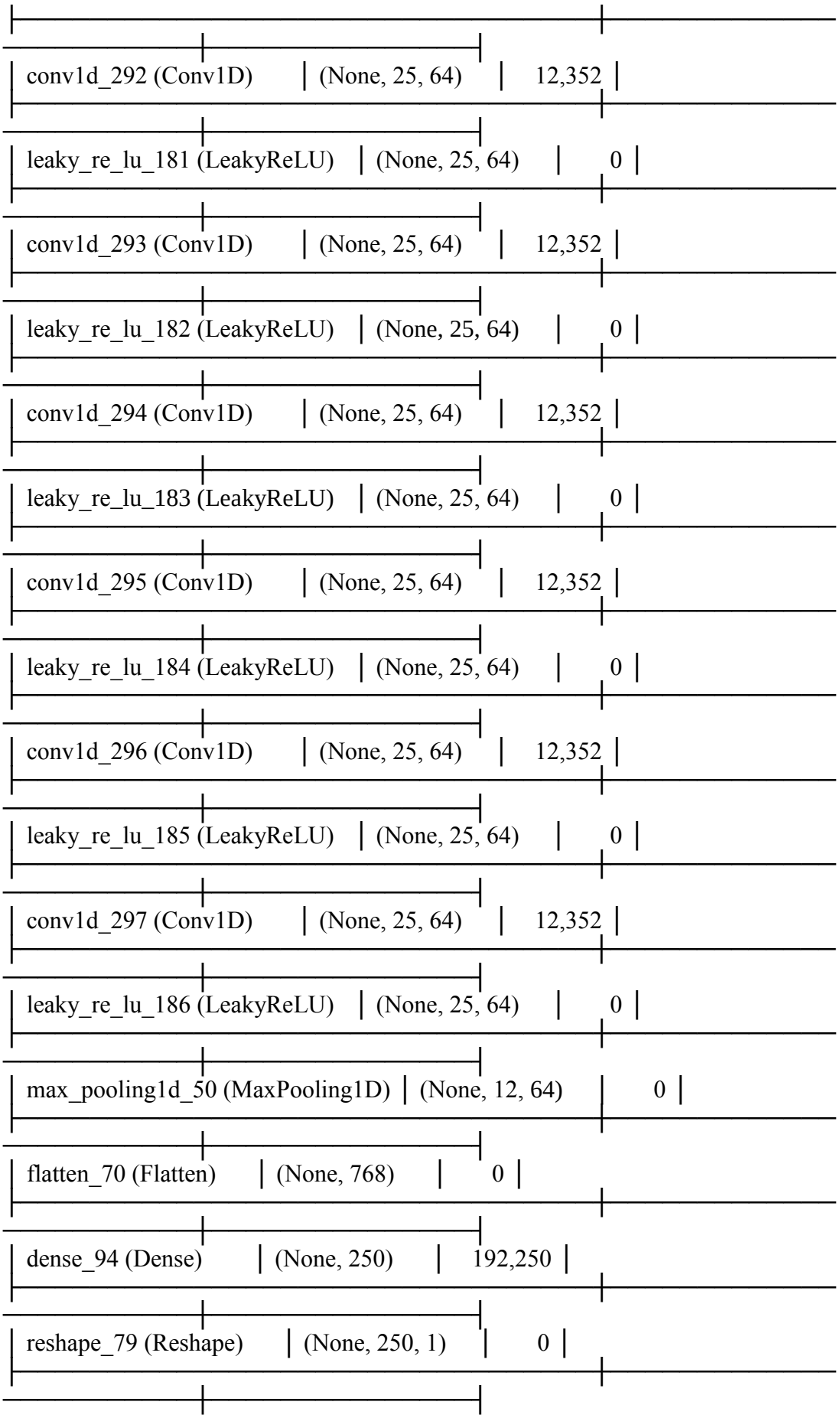
leaky_re_lu_161 (LeakyReLU)	(None, 24, 64)	0
conv1d_273 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_162 (LeakyReLU)	(None, 24, 64)	0
flatten_66 (Flatten)	(None, 1536)	0
dense_90 (Dense)	(None, 250)	384,250
reshape_73 (Reshape)	(None, 25, 10)	0

Total params: 13,044,724 (49.76 MB)
Trainable params: 13,044,724 (49.76 MB)
Non-trainable params: 0 (0.00 B)

8 hidden layers of Nadam-Optimized Model

Deep Autoencoder Model with Nadam Optimizer
Model: "sequential_20"

Layer (type)	**Output Shape**	**Param #**
reshape_78 (Reshape)	(None, 25, 10)	0
conv1d_290 (Conv1D)	(None, 25, 64)	1,984
leaky_re_lu_179 (LeakyReLU)	(None, 25, 64)	0
conv1d_291 (Conv1D)	(None, 25, 64)	12,352
leaky_re_lu_180 (LeakyReLU)	(None, 25, 64)	0


conv1d_292 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_181 (LeakyReLU) (None, 25, 64) 0
conv1d_293 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_182 (LeakyReLU) (None, 25, 64) 0
conv1d_294 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_183 (LeakyReLU) (None, 25, 64) 0
conv1d_295 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_184 (LeakyReLU) (None, 25, 64) 0
conv1d_296 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_185 (LeakyReLU) (None, 25, 64) 0
conv1d_297 (Conv1D) (None, 25, 64) 12,352
leaky_re_lu_186 (LeakyReLU) (None, 25, 64) 0
max_pooling1d_50 (MaxPooling1D) (None, 12, 64) 0
flatten_70 (Flatten) (None, 768) 0
dense_94 (Dense) (None, 250) 192,250
reshape_79 (Reshape) (None, 250, 1) 0

lstm_65 (LSTM)	(None, 250, 64)	16,896
elu_19 (ELU)	(None, 250, 64)	0
flatten_71 (Flatten)	(None, 16000)	0
dense_95 (Dense)	(None, 768)	12,288,768
reshape_80 (Reshape)	(None, 12, 64)	0
up_sampling1d_79 (UpSampling1D)	(None, 24, 64)	0
conv1d_298 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_187 (LeakyReLU)	(None, 24, 64)	0
conv1d_299 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_188 (LeakyReLU)	(None, 24, 64)	0
conv1d_300 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_189 (LeakyReLU)	(None, 24, 64)	0
conv1d_301 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_190 (LeakyReLU)	(None, 24, 64)	0
conv1d_302 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_191 (LeakyReLU)	(None, 24, 64)	0
conv1d_303 (Conv1D)	(None, 24, 64)	12,352

leaky_re_lu_192 (LeakyReLU)	(None, 24, 64)	0
conv1d_304 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_193 (LeakyReLU)	(None, 24, 64)	0
conv1d_305 (Conv1D)	(None, 24, 64)	12,352
leaky_re_lu_194 (LeakyReLU)	(None, 24, 64)	0
flatten_72 (Flatten)	(None, 1536)	0
dense_96 (Dense)	(None, 250)	384,250
reshape_81 (Reshape)	(None, 25, 10)	0

Total params: 13,069,428 (49.86 MB)
Trainable params: 13,069,428 (49.86 MB)
Non-trainable params: 0 (0.00 B)

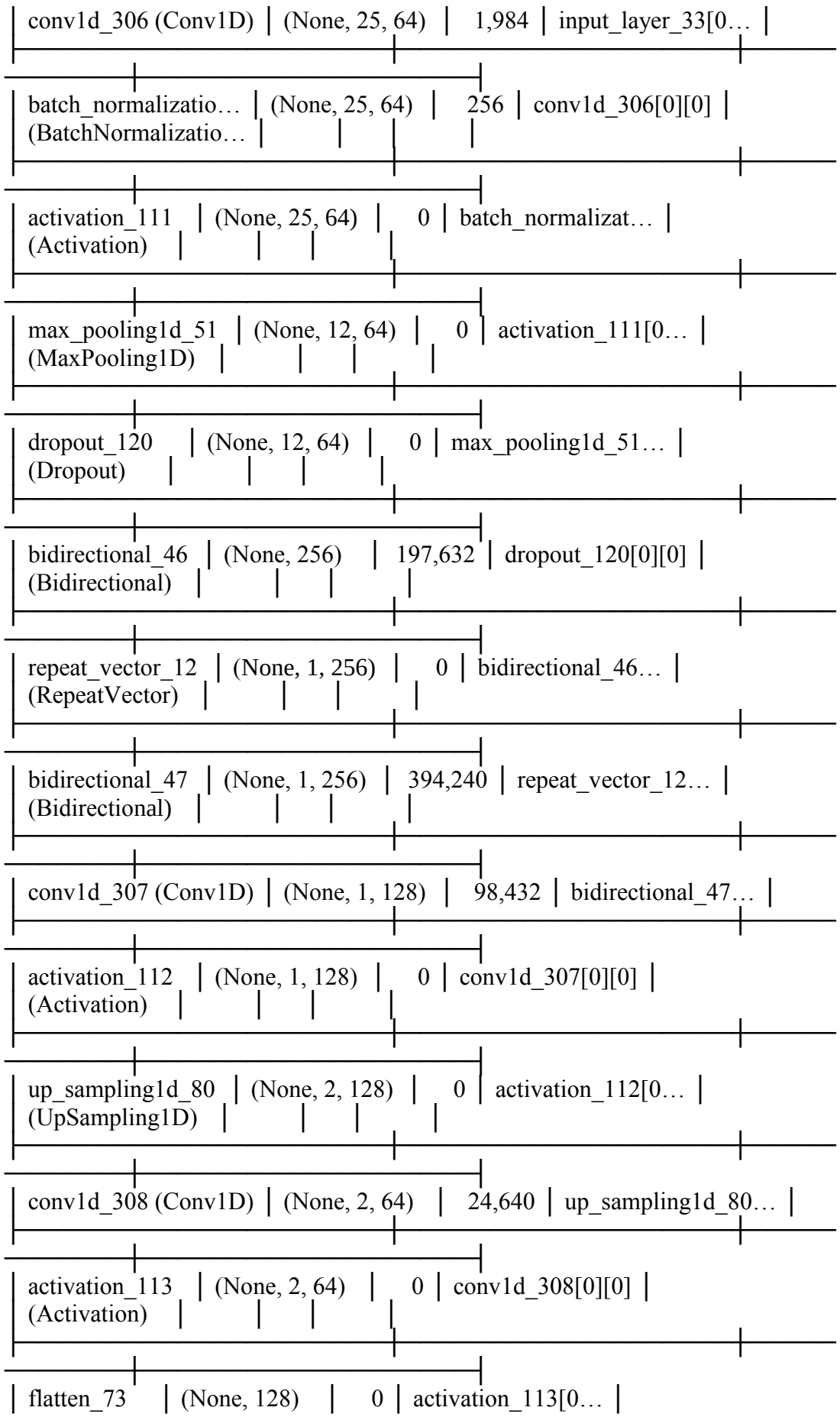
Appendix 4: Nadam-Optimized and MLP-Enhanced Hybrid Model Architectures

1 hidden layer of MLP-Enhanced Model

Deep Autoencoder Model with Nadam Optimizer, Regularization, and Batch Optimization

Model: "functional_687"

Layer (type)	**Output Shape**	**Param #**	**Connected to**
input_layer_33 (InputLayer)	(None, 25, 10)	0	-

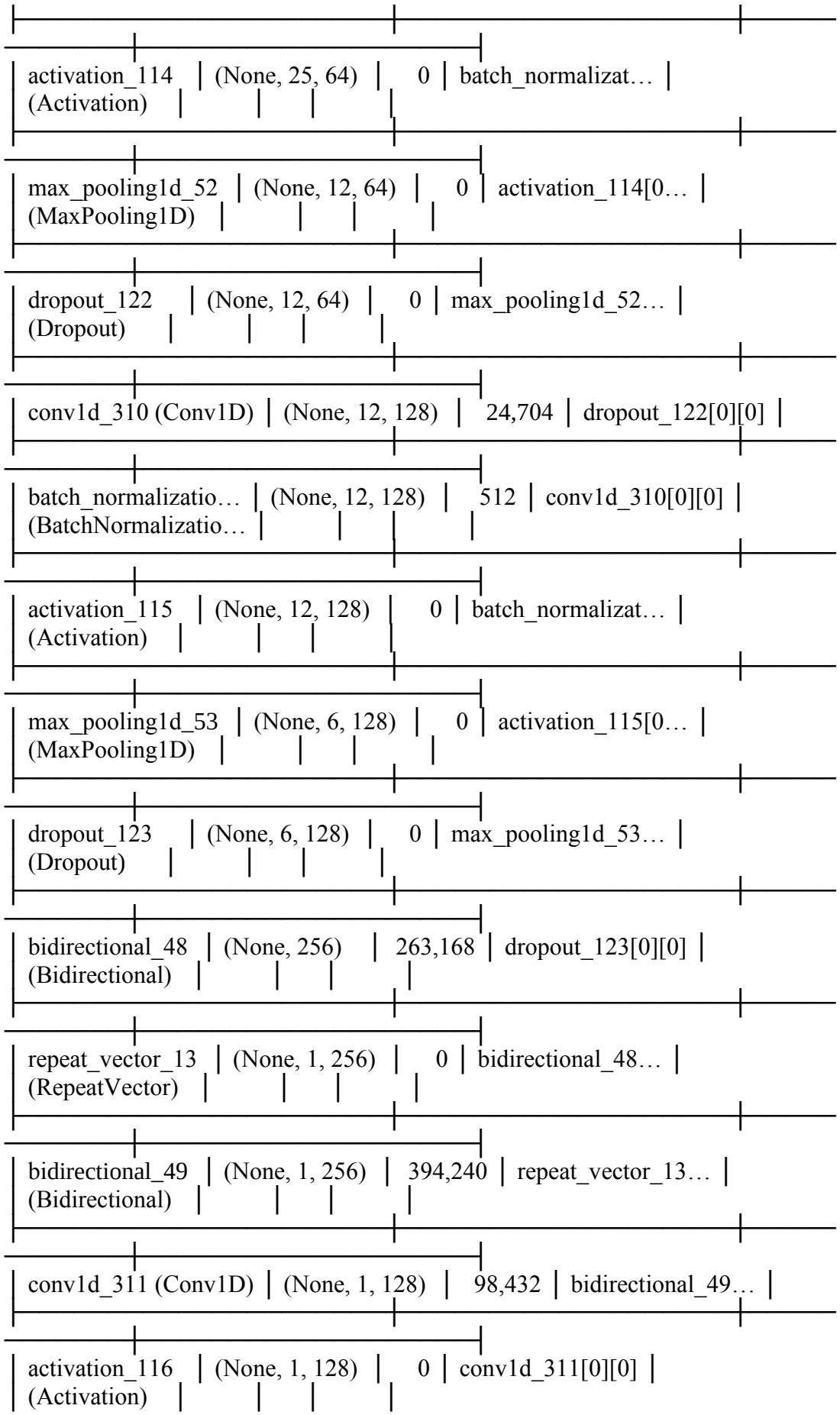

conv1d_306 (Conv1D) (None, 25, 64) 1,984 input_layer_33[0…
batch_normalizatio… (BatchNormalizatio… (None, 25, 64) 256 conv1d_306[0][0]
activation_111 (Activation) (None, 25, 64) 0 batch_normalizat…
max_pooling1d_51 (MaxPooling1D) (None, 12, 64) 0 activation_111[0…
dropout_120 (Dropout) (None, 12, 64) 0 max_pooling1d_51…
bidirectional_46 (Bidirectional) (None, 256) 197,632 dropout_120[0][0]
repeat_vector_12 (RepeatVector) (None, 1, 256) 0 bidirectional_46…
bidirectional_47 (Bidirectional) (None, 1, 256) 394,240 repeat_vector_12…
conv1d_307 (Conv1D) (None, 1, 128) 98,432 bidirectional_47…
activation_112 (Activation) (None, 1, 128) 0 conv1d_307[0][0]
up_sampling1d_80 (UpSampling1D) (None, 2, 128) 0 activation_112[0…
conv1d_308 (Conv1D) (None, 2, 64) 24,640 up_sampling1d_80…
activation_113 (Activation) (None, 2, 64) 0 conv1d_308[0][0]
flatten_73 (None, 128) 0 activation_113[0…

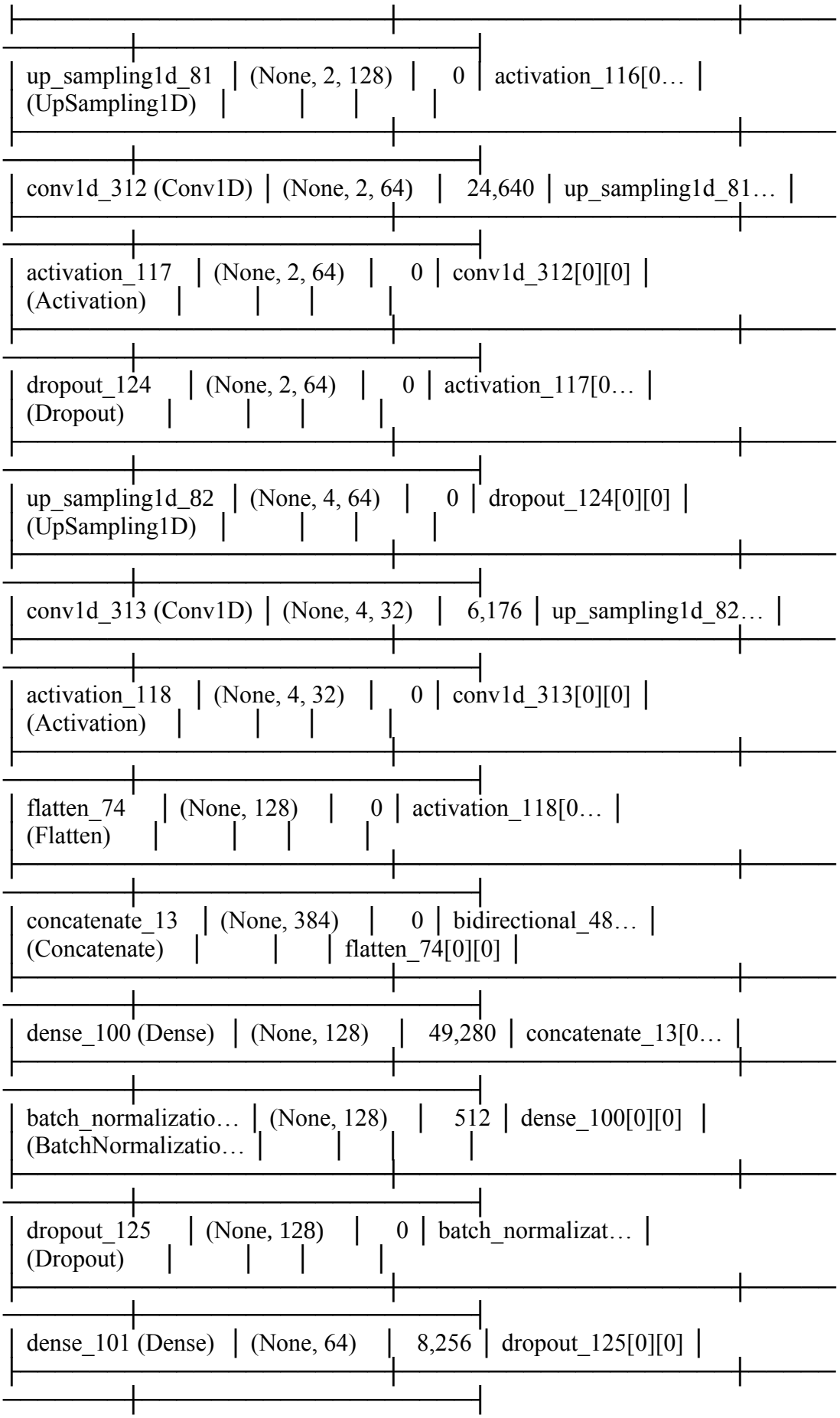
(Flatten)			
concatenate_12 (Concatenate)	(None, 384)	0	bidirectional_46… flatten_73[0][0]
dense_97 (Dense)	(None, 128)	49,280	concatenate_12[0…
batch_normalizatio… (BatchNormalizatio…	(None, 128)	512	dense_97[0][0]
dropout_121 (Dropout)	(None, 128)	0	batch_normalizat…
dense_98 (Dense)	(None, 64)	8,256	dropout_121[0][0]
dense_99 (Dense)	(None, 1)	65	dense_98[0][0]

Total params: 775,297 (2.96 MB)
Trainable params: 774,913 (2.96 MB)
Non-trainable params: 384 (1.50 KB)
2 hidden layers of MLP-Enhanced Model

Deep Autoencoder Model with Nadam Optimizer, Regularization, and Batch Optimization
Model: "functional_688"

Layer (type)	Output Shape	Param #	Connected to
input_layer_34 (InputLayer)	(None, 25, 10)	0	-
conv1d_309 (Conv1D)	(None, 25, 64)	1,984	input_layer_34[0…
batch_normalizatio… (BatchNormalizatio…	(None, 25, 64)	256	conv1d_309[0][0]


activation_114 (Activation) (None, 25, 64) 0 batch_normalizat…
max_pooling1d_52 (MaxPooling1D) (None, 12, 64) 0 activation_114[0…
dropout_122 (Dropout) (None, 12, 64) 0 max_pooling1d_52…
conv1d_310 (Conv1D) (None, 12, 128) 24,704 dropout_122[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 12, 128) 512 conv1d_310[0][0]
activation_115 (Activation) (None, 12, 128) 0 batch_normalizat…
max_pooling1d_53 (MaxPooling1D) (None, 6, 128) 0 activation_115[0…
dropout_123 (Dropout) (None, 6, 128) 0 max_pooling1d_53…
bidirectional_48 (Bidirectional) (None, 256) 263,168 dropout_123[0][0]
repeat_vector_13 (RepeatVector) (None, 1, 256) 0 bidirectional_48…
bidirectional_49 (Bidirectional) (None, 1, 256) 394,240 repeat_vector_13…
conv1d_311 (Conv1D) (None, 1, 128) 98,432 bidirectional_49…
activation_116 (Activation) (None, 1, 128) 0 conv1d_311[0][0]


up_sampling1d_81 (UpSampling1D) (None, 2, 128) 0 activation_116[0…
conv1d_312 (Conv1D) (None, 2, 64) 24,640 up_sampling1d_81…
activation_117 (Activation) (None, 2, 64) 0 conv1d_312[0][0]
dropout_124 (Dropout) (None, 2, 64) 0 activation_117[0…
up_sampling1d_82 (UpSampling1D) (None, 4, 64) 0 dropout_124[0][0]
conv1d_313 (Conv1D) (None, 4, 32) 6,176 up_sampling1d_82…
activation_118 (Activation) (None, 4, 32) 0 conv1d_313[0][0]
flatten_74 (Flatten) (None, 128) 0 activation_118[0…
concatenate_13 (Concatenate) (None, 384) 0 bidirectional_48… flatten_74[0][0]
dense_100 (Dense) (None, 128) 49,280 concatenate_13[0…
batch_normalizatio… (BatchNormalizatio… (None, 128) 512 dense_100[0][0]
dropout_125 (Dropout) (None, 128) 0 batch_normalizat…
dense_101 (Dense) (None, 64) 8,256 dropout_125[0][0]

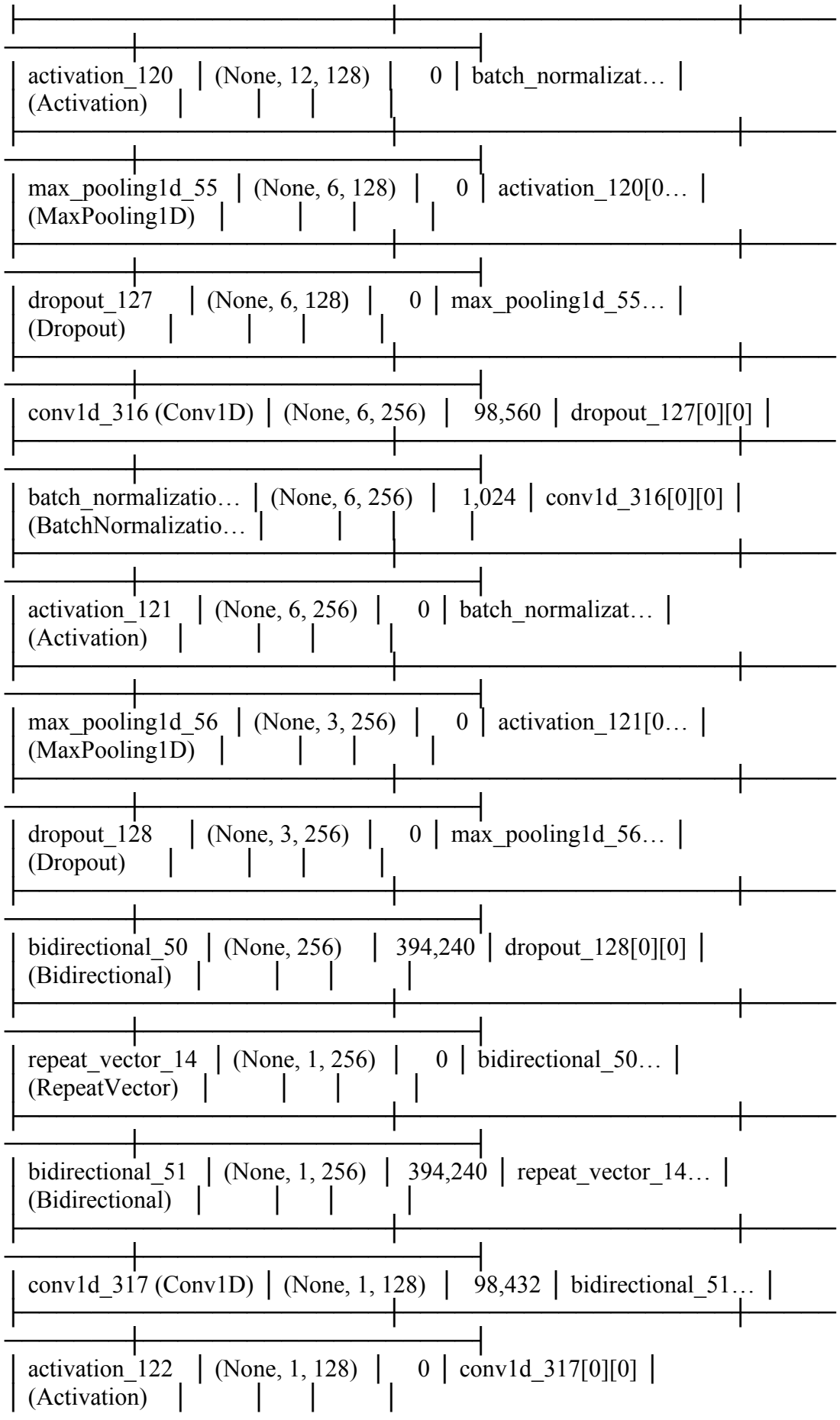
dense_102 (Dense)	(None, 1)	65	dense_101[0][0]

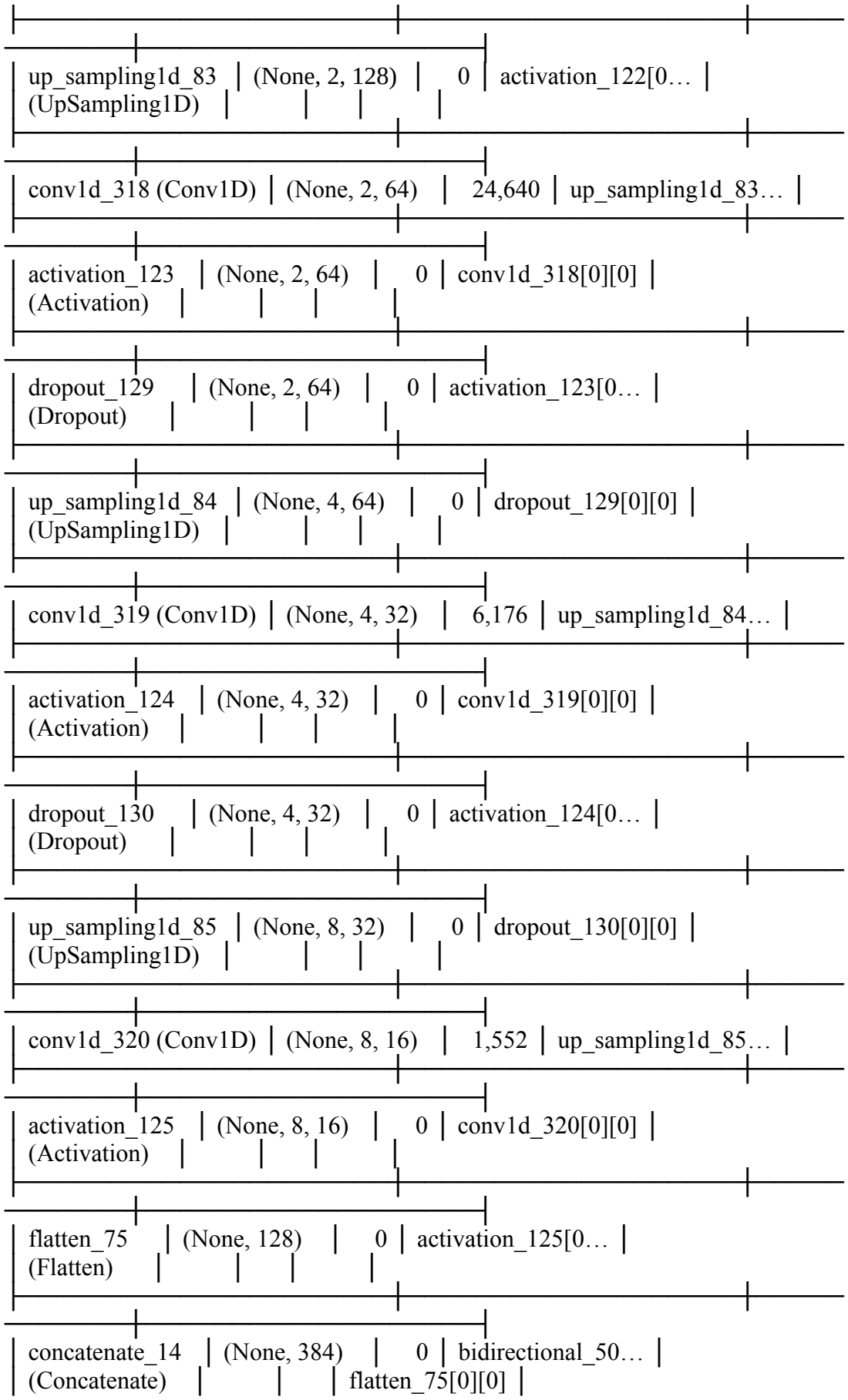
Total params: 872,225 (3.33 MB)
Trainable params: 871,585 (3.32 MB)
Non-trainable params: 640 (2.50 KB)

3 hidden layers of MLP-Enhanced Model

Deep Autoencoder Model with Nadam Optimizer, Regularization, and Batch Optimization
Model: "functional_689"

Layer (type)	Output Shape	Param #	Connected to
input_layer_35 (InputLayer)	(None, 25, 10)	0	-
conv1d_314 (Conv1D)	(None, 25, 64)	1,984	input_layer_35[0…
batch_normalizatio… (BatchNormalizatio…	(None, 25, 64)	256	conv1d_314[0][0]
activation_119 (Activation)	(None, 25, 64)	0	batch_normalizat…
max_pooling1d_54 (MaxPooling1D)	(None, 12, 64)	0	activation_119[0…
dropout_126 (Dropout)	(None, 12, 64)	0	max_pooling1d_54…
conv1d_315 (Conv1D)	(None, 12, 128)	24,704	dropout_126[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 12, 128)	512	conv1d_315[0][0]


activation_120 (Activation) (None, 12, 128) 0 batch_normalizat…
max_pooling1d_55 (MaxPooling1D) (None, 6, 128) 0 activation_120[0…
dropout_127 (Dropout) (None, 6, 128) 0 max_pooling1d_55…
conv1d_316 (Conv1D) (None, 6, 256) 98,560 dropout_127[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 6, 256) 1,024 conv1d_316[0][0]
activation_121 (Activation) (None, 6, 256) 0 batch_normalizat…
max_pooling1d_56 (MaxPooling1D) (None, 3, 256) 0 activation_121[0…
dropout_128 (Dropout) (None, 3, 256) 0 max_pooling1d_56…
bidirectional_50 (Bidirectional) (None, 256) 394,240 dropout_128[0][0]
repeat_vector_14 (RepeatVector) (None, 1, 256) 0 bidirectional_50…
bidirectional_51 (Bidirectional) (None, 1, 256) 394,240 repeat_vector_14…
conv1d_317 (Conv1D) (None, 1, 128) 98,432 bidirectional_51…
activation_122 (Activation) (None, 1, 128) 0 conv1d_317[0][0]

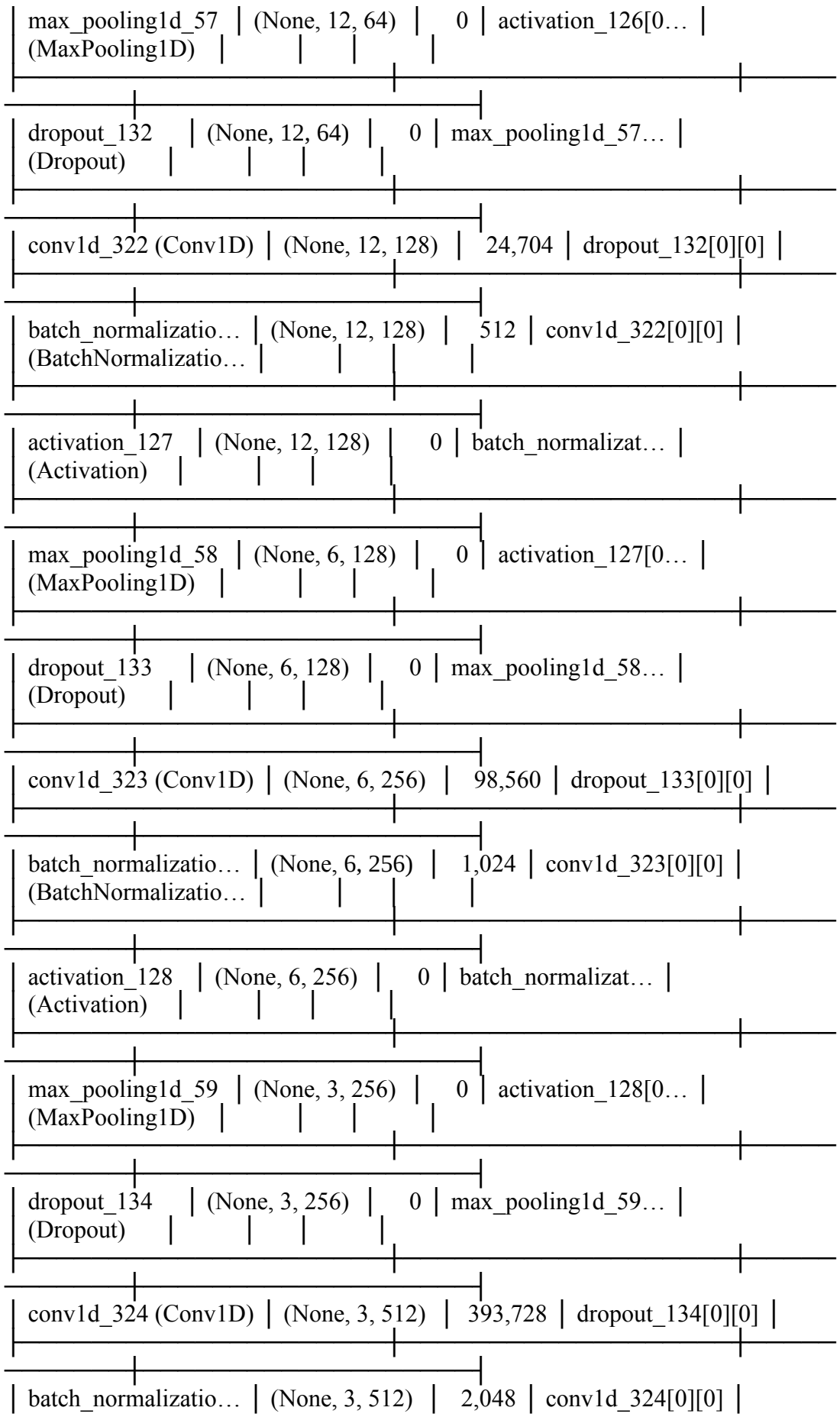

up_sampling1d_83 (UpSampling1D) (None, 2, 128) 0 activation_122[0…
conv1d_318 (Conv1D) (None, 2, 64) 24,640 up_sampling1d_83…
activation_123 (Activation) (None, 2, 64) 0 conv1d_318[0][0]
dropout_129 (Dropout) (None, 2, 64) 0 activation_123[0…
up_sampling1d_84 (UpSampling1D) (None, 4, 64) 0 dropout_129[0][0]
conv1d_319 (Conv1D) (None, 4, 32) 6,176 up_sampling1d_84…
activation_124 (Activation) (None, 4, 32) 0 conv1d_319[0][0]
dropout_130 (Dropout) (None, 4, 32) 0 activation_124[0…
up_sampling1d_85 (UpSampling1D) (None, 8, 32) 0 dropout_130[0][0]
conv1d_320 (Conv1D) (None, 8, 16) 1,552 up_sampling1d_85…
activation_125 (Activation) (None, 8, 16) 0 conv1d_320[0][0]
flatten_75 (Flatten) (None, 128) 0 activation_125[0…
concatenate_14 (Concatenate) (None, 384) 0 bidirectional_50… flatten_75[0][0]

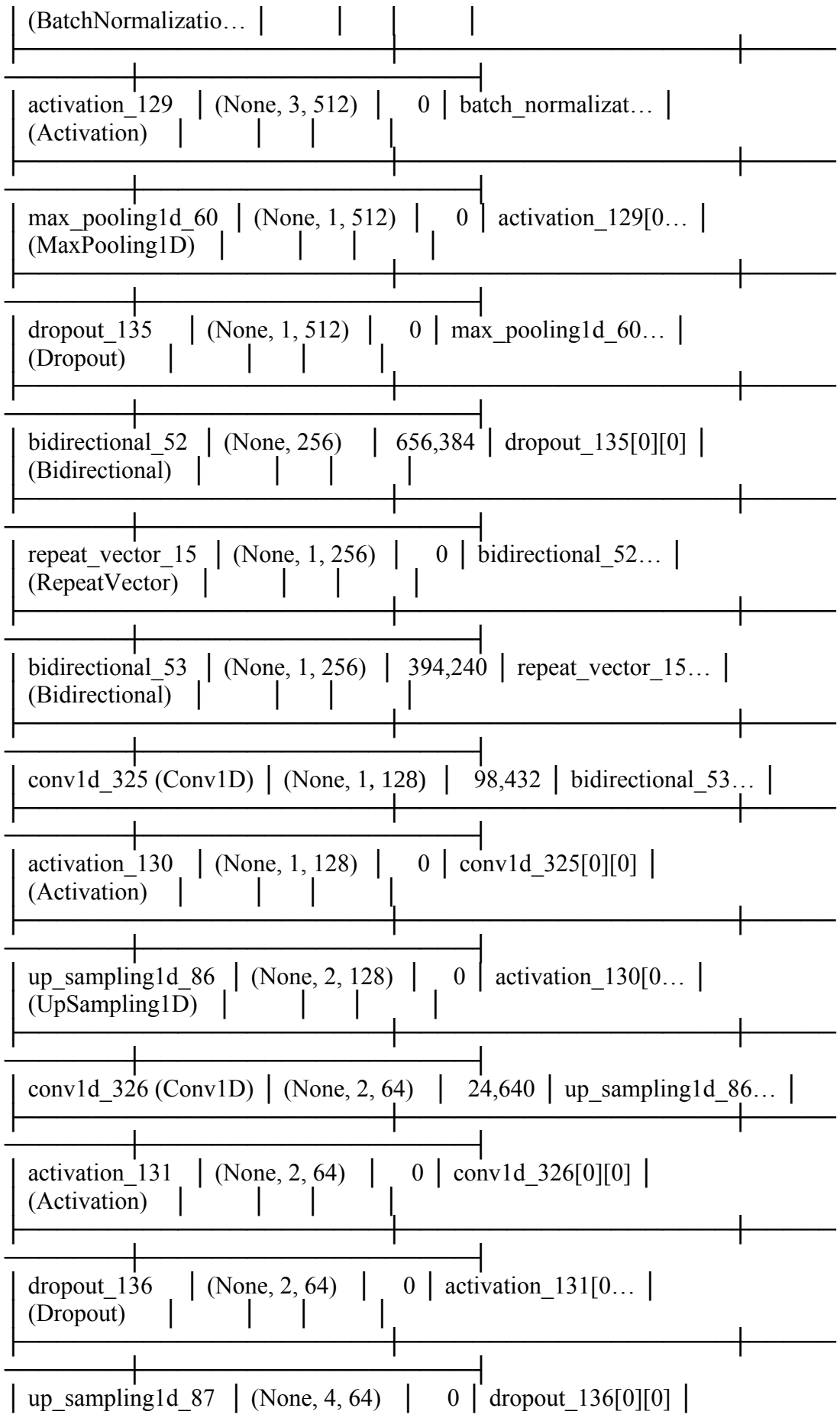
dense_103 (Dense)	(None, 128)	49,280	concatenate_14[0…
batch_normalizatio… (BatchNormalizatio…	(None, 128)	512	dense_103[0][0]
dropout_131 (Dropout)	(None, 128)	0	batch_normalizat…
dense_104 (Dense)	(None, 64)	8,256	dropout_131[0][0]
dense_105 (Dense)	(None, 1)	65	dense_104[0][0]

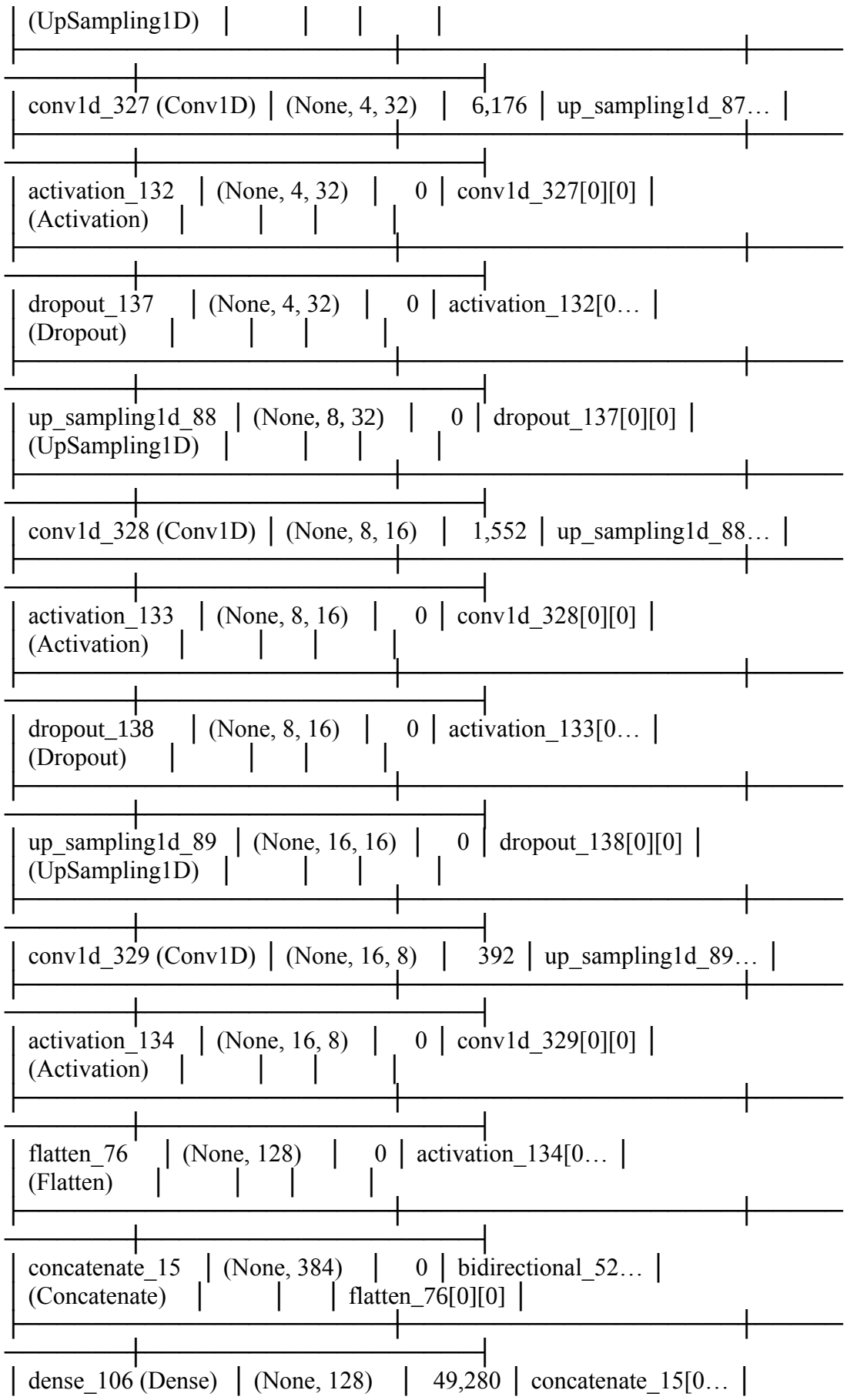
Total params: 1,104,433 (4.21 MB)
Trainable params: 1,103,281 (4.21 MB)
Non-trainable params: 1,152 (4.50 KB)
4 hidden layers of MLP-Enhanced Model

Deep Autoencoder Model with Nadam Optimizer, Regularization, and Batch Optimization
Model: "functional_690"

Layer (type)	**Output Shape**	**Param #**	**Connected to**
input_layer_36 (InputLayer)	(None, 25, 10)	0	-
conv1d_321 (Conv1D)	(None, 25, 64)	1,984	input_layer_36[0…
batch_normalizatio… (BatchNormalizatio…	(None, 25, 64)	256	conv1d_321[0][0]
activation_126 (Activation)	(None, 25, 64)	0	batch_normalizat…


max_pooling1d_57 (MaxPooling1D) (None, 12, 64) 0 activation_126[0…
dropout_132 (Dropout) (None, 12, 64) 0 max_pooling1d_57…
conv1d_322 (Conv1D) (None, 12, 128) 24,704 dropout_132[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 12, 128) 512 conv1d_322[0][0]
activation_127 (Activation) (None, 12, 128) 0 batch_normalizat…
max_pooling1d_58 (MaxPooling1D) (None, 6, 128) 0 activation_127[0…
dropout_133 (Dropout) (None, 6, 128) 0 max_pooling1d_58…
conv1d_323 (Conv1D) (None, 6, 256) 98,560 dropout_133[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 6, 256) 1,024 conv1d_323[0][0]
activation_128 (Activation) (None, 6, 256) 0 batch_normalizat…
max_pooling1d_59 (MaxPooling1D) (None, 3, 256) 0 activation_128[0…
dropout_134 (Dropout) (None, 3, 256) 0 max_pooling1d_59…
conv1d_324 (Conv1D) (None, 3, 512) 393,728 dropout_134[0][0]
batch_normalizatio… (None, 3, 512) 2,048 conv1d_324[0][0]


(BatchNormalizatio…
activation_129 (Activation) (None, 3, 512) 0 batch_normalizat…
max_pooling1d_60 (MaxPooling1D) (None, 1, 512) 0 activation_129[0…
dropout_135 (Dropout) (None, 1, 512) 0 max_pooling1d_60…
bidirectional_52 (Bidirectional) (None, 256) 656,384 dropout_135[0][0]
repeat_vector_15 (RepeatVector) (None, 1, 256) 0 bidirectional_52…
bidirectional_53 (Bidirectional) (None, 1, 256) 394,240 repeat_vector_15…
conv1d_325 (Conv1D) (None, 1, 128) 98,432 bidirectional_53…
activation_130 (Activation) (None, 1, 128) 0 conv1d_325[0][0]
up_sampling1d_86 (UpSampling1D) (None, 2, 128) 0 activation_130[0…
conv1d_326 (Conv1D) (None, 2, 64) 24,640 up_sampling1d_86…
activation_131 (Activation) (None, 2, 64) 0 conv1d_326[0][0]
dropout_136 (Dropout) (None, 2, 64) 0 activation_131[0…
up_sampling1d_87 (None, 4, 64) 0 dropout_136[0][0]


(UpSampling1D)
conv1d_327 (Conv1D) (None, 4, 32) 6,176 up_sampling1d_87…
activation_132 (Activation) (None, 4, 32) 0 conv1d_327[0][0]
dropout_137 (Dropout) (None, 4, 32) 0 activation_132[0…
up_sampling1d_88 (UpSampling1D) (None, 8, 32) 0 dropout_137[0][0]
conv1d_328 (Conv1D) (None, 8, 16) 1,552 up_sampling1d_88…
activation_133 (Activation) (None, 8, 16) 0 conv1d_328[0][0]
dropout_138 (Dropout) (None, 8, 16) 0 activation_133[0…
up_sampling1d_89 (UpSampling1D) (None, 16, 16) 0 dropout_138[0][0]
conv1d_329 (Conv1D) (None, 16, 8) 392 up_sampling1d_89…
activation_134 (Activation) (None, 16, 8) 0 conv1d_329[0][0]
flatten_76 (Flatten) (None, 128) 0 activation_134[0…
concatenate_15 (Concatenate) (None, 384) 0 bidirectional_52… flatten_76[0][0]
dense_106 (Dense) (None, 128) 49,280 concatenate_15[0…

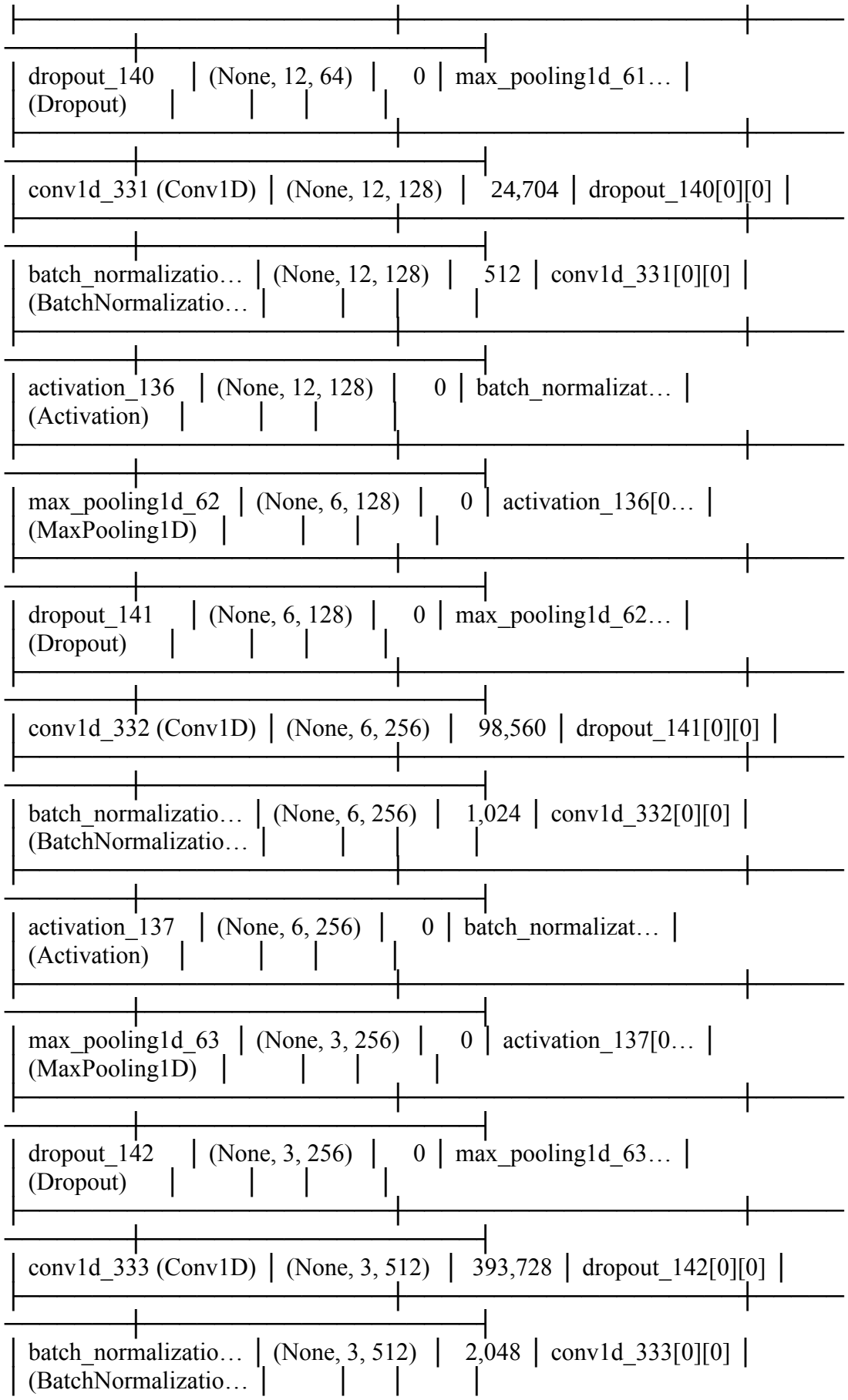
batch_normalizatio… (BatchNormalizatio…	(None, 128)	512	dense_106[0][0]
dropout_139 (Dropout)	(None, 128)	0	batch_normalizat…
dense_107 (Dense)	(None, 64)	8,256	dropout_139[0][0]
dense_108 (Dense)	(None, 1)	65	dense_107[0][0]

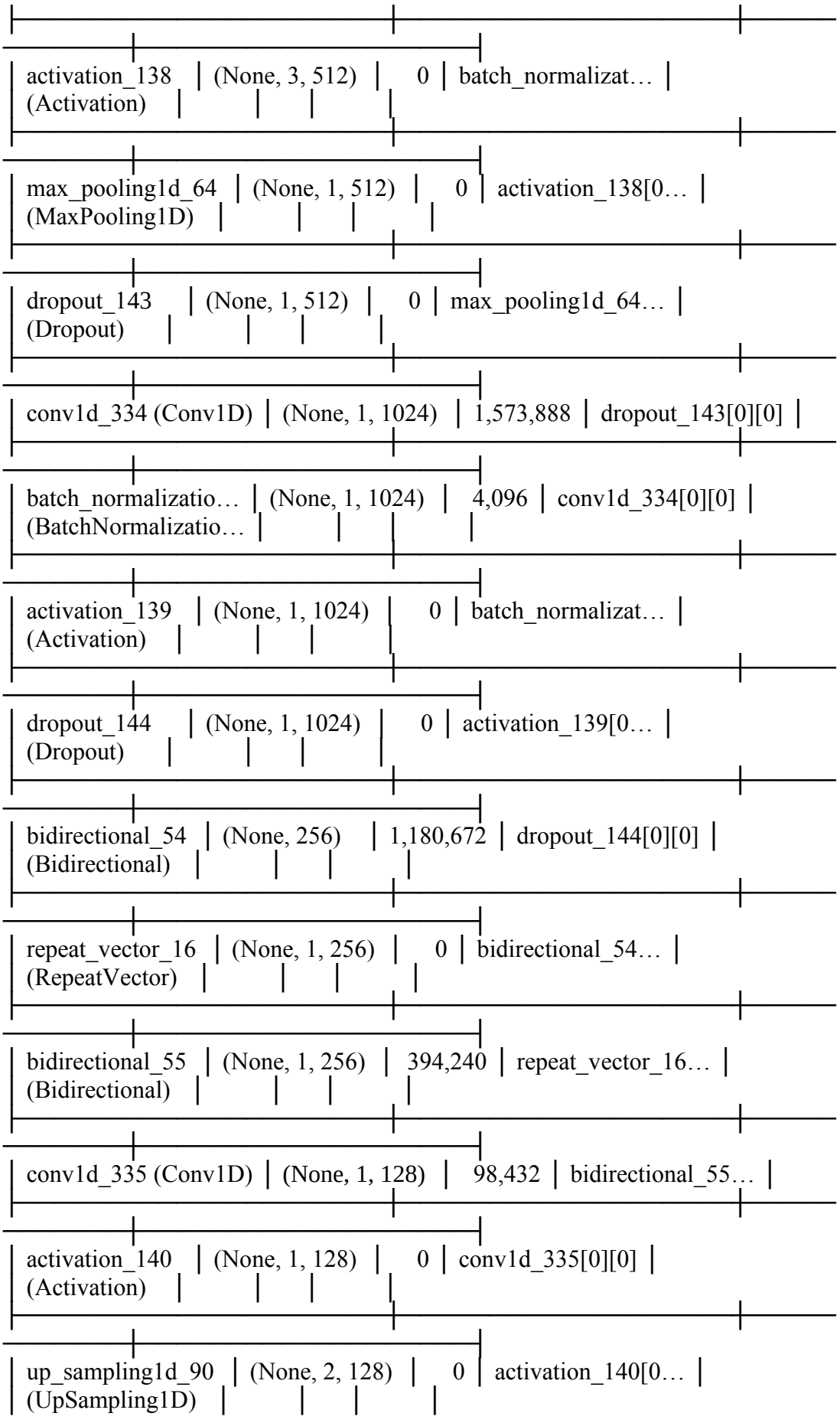
Total params: 1,762,745 (6.72 MB)
Trainable params: 1,760,569 (6.72 MB)
Non-trainable params: 2,176 (8.50 KB)

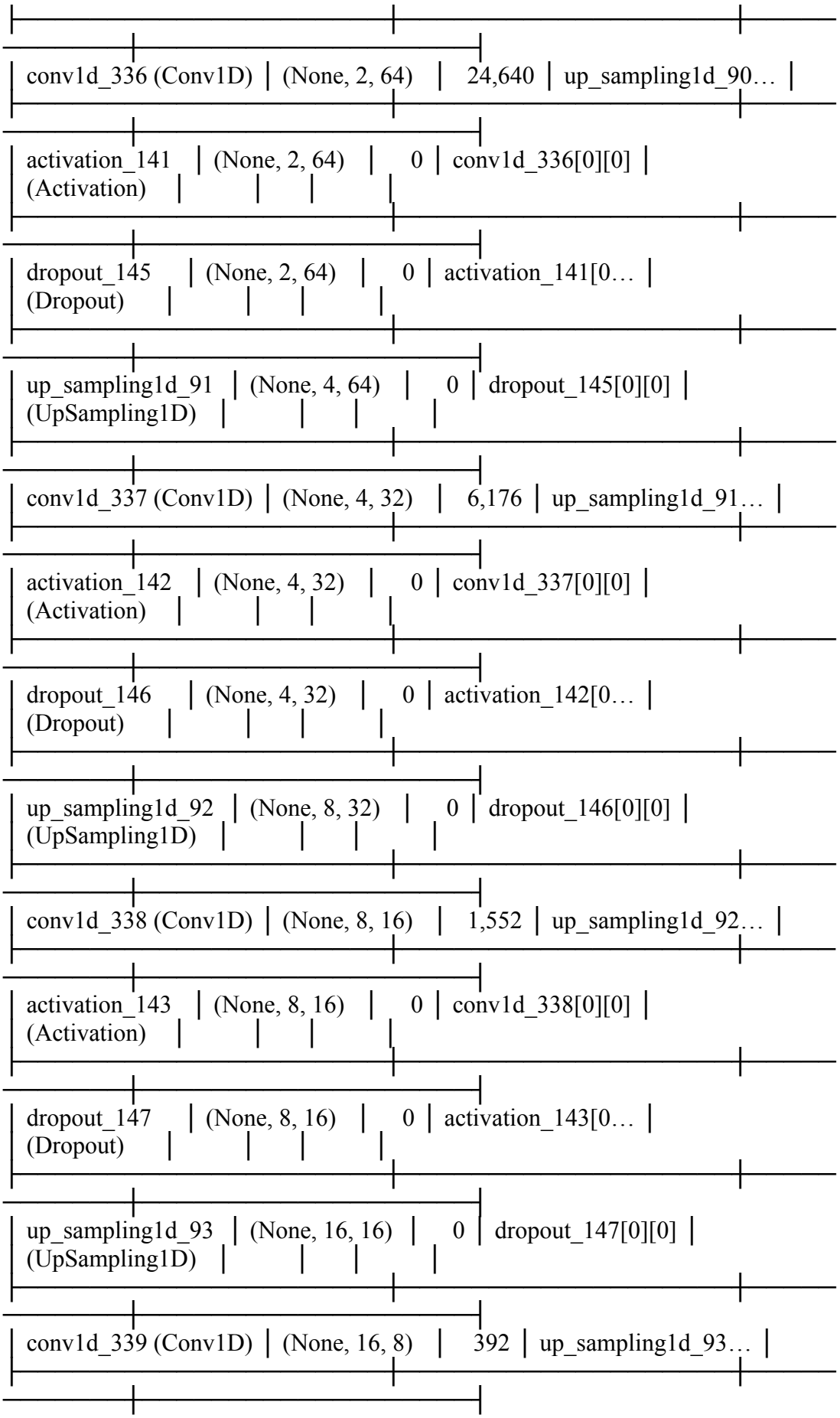
5 hidden layers of MLP-Enhanced Model

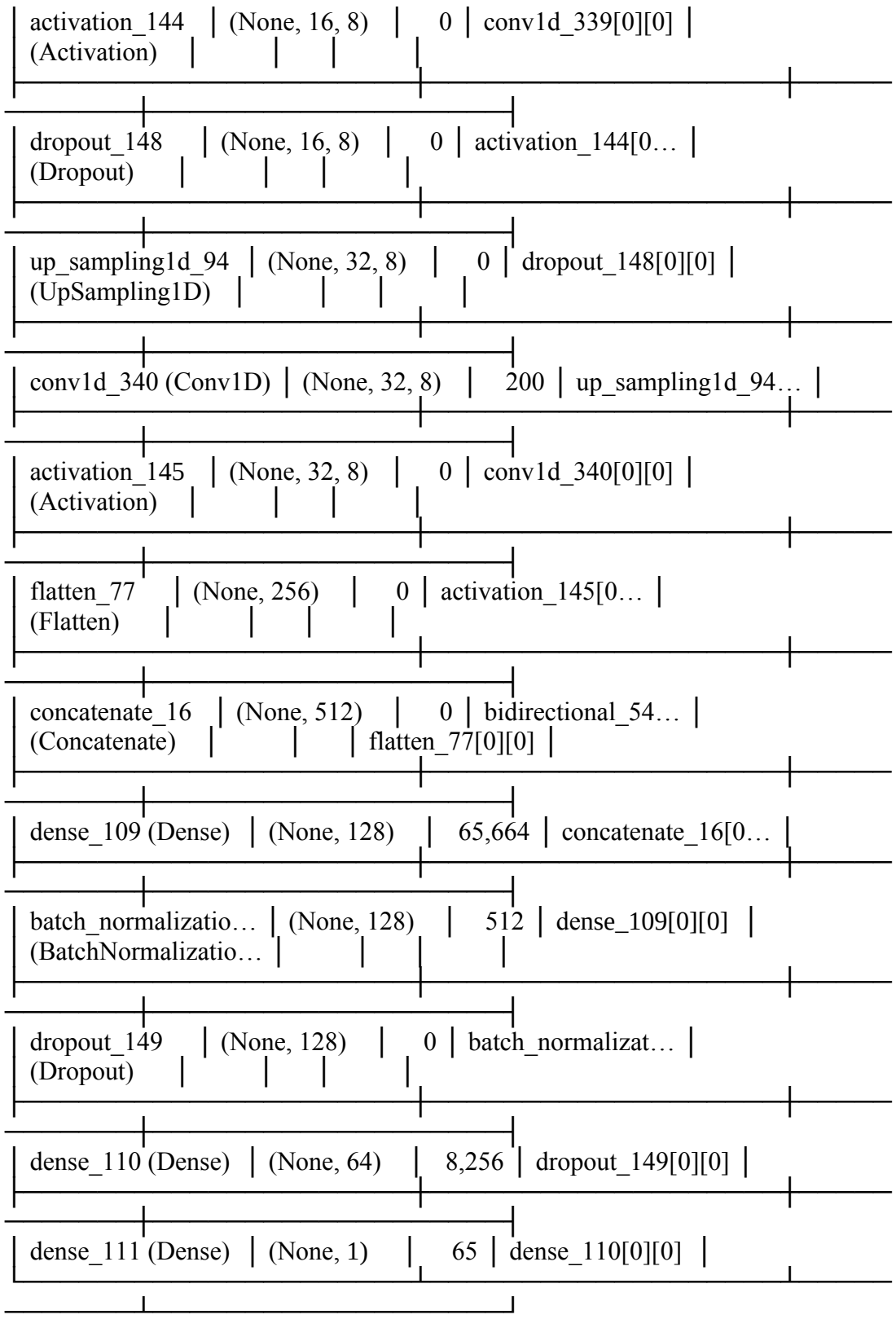
Deep Autoencoder Model with Nadam Optimizer, Regularization, and Batch Optimization
Model: "functional_691"

Layer (type)	Output Shape	Param #	Connected to
input_layer_37 (InputLayer)	(None, 25, 10)	0	-
conv1d_330 (Conv1D)	(None, 25, 64)	1,984	input_layer_37[0…
batch_normalizatio… (BatchNormalizatio…	(None, 25, 64)	256	conv1d_330[0][0]
activation_135 (Activation)	(None, 25, 64)	0	batch_normalizat…
max_pooling1d_61 (MaxPooling1D)	(None, 12, 64)	0	activation_135[0…


dropout_140 (Dropout) | (None, 12, 64) | 0 | max_pooling1d_61…
conv1d_331 (Conv1D) | (None, 12, 128) | 24,704 | dropout_140[0][0]
batch_normalizatio… (BatchNormalizatio… | (None, 12, 128) | 512 | conv1d_331[0][0]
activation_136 (Activation) | (None, 12, 128) | 0 | batch_normalizat…
max_pooling1d_62 (MaxPooling1D) | (None, 6, 128) | 0 | activation_136[0…
dropout_141 (Dropout) | (None, 6, 128) | 0 | max_pooling1d_62…
conv1d_332 (Conv1D) | (None, 6, 256) | 98,560 | dropout_141[0][0]
batch_normalizatio… (BatchNormalizatio… | (None, 6, 256) | 1,024 | conv1d_332[0][0]
activation_137 (Activation) | (None, 6, 256) | 0 | batch_normalizat…
max_pooling1d_63 (MaxPooling1D) | (None, 3, 256) | 0 | activation_137[0…
dropout_142 (Dropout) | (None, 3, 256) | 0 | max_pooling1d_63…
conv1d_333 (Conv1D) | (None, 3, 512) | 393,728 | dropout_142[0][0]
batch_normalizatio… (BatchNormalizatio… | (None, 3, 512) | 2,048 | conv1d_333[0][0]


activation_138 (Activation) (None, 3, 512) 0 batch_normalizat…
max_pooling1d_64 (MaxPooling1D) (None, 1, 512) 0 activation_138[0…
dropout_143 (Dropout) (None, 1, 512) 0 max_pooling1d_64…
conv1d_334 (Conv1D) (None, 1, 1024) 1,573,888 dropout_143[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 1, 1024) 4,096 conv1d_334[0][0]
activation_139 (Activation) (None, 1, 1024) 0 batch_normalizat…
dropout_144 (Dropout) (None, 1, 1024) 0 activation_139[0…
bidirectional_54 (Bidirectional) (None, 256) 1,180,672 dropout_144[0][0]
repeat_vector_16 (RepeatVector) (None, 1, 256) 0 bidirectional_54…
bidirectional_55 (Bidirectional) (None, 1, 256) 394,240 repeat_vector_16…
conv1d_335 (Conv1D) (None, 1, 128) 98,432 bidirectional_55…
activation_140 (Activation) (None, 1, 128) 0 conv1d_335[0][0]
up_sampling1d_90 (UpSampling1D) (None, 2, 128) 0 activation_140[0…


conv1d_336 (Conv1D) | (None, 2, 64) | 24,640 | up_sampling1d_90…
activation_141 (Activation) | (None, 2, 64) | 0 | conv1d_336[0][0]
dropout_145 (Dropout) | (None, 2, 64) | 0 | activation_141[0…
up_sampling1d_91 (UpSampling1D) | (None, 4, 64) | 0 | dropout_145[0][0]
conv1d_337 (Conv1D) | (None, 4, 32) | 6,176 | up_sampling1d_91…
activation_142 (Activation) | (None, 4, 32) | 0 | conv1d_337[0][0]
dropout_146 (Dropout) | (None, 4, 32) | 0 | activation_142[0…
up_sampling1d_92 (UpSampling1D) | (None, 8, 32) | 0 | dropout_146[0][0]
conv1d_338 (Conv1D) | (None, 8, 16) | 1,552 | up_sampling1d_92…
activation_143 (Activation) | (None, 8, 16) | 0 | conv1d_338[0][0]
dropout_147 (Dropout) | (None, 8, 16) | 0 | activation_143[0…
up_sampling1d_93 (UpSampling1D) | (None, 16, 16) | 0 | dropout_147[0][0]
conv1d_339 (Conv1D) | (None, 16, 8) | 392 | up_sampling1d_93…



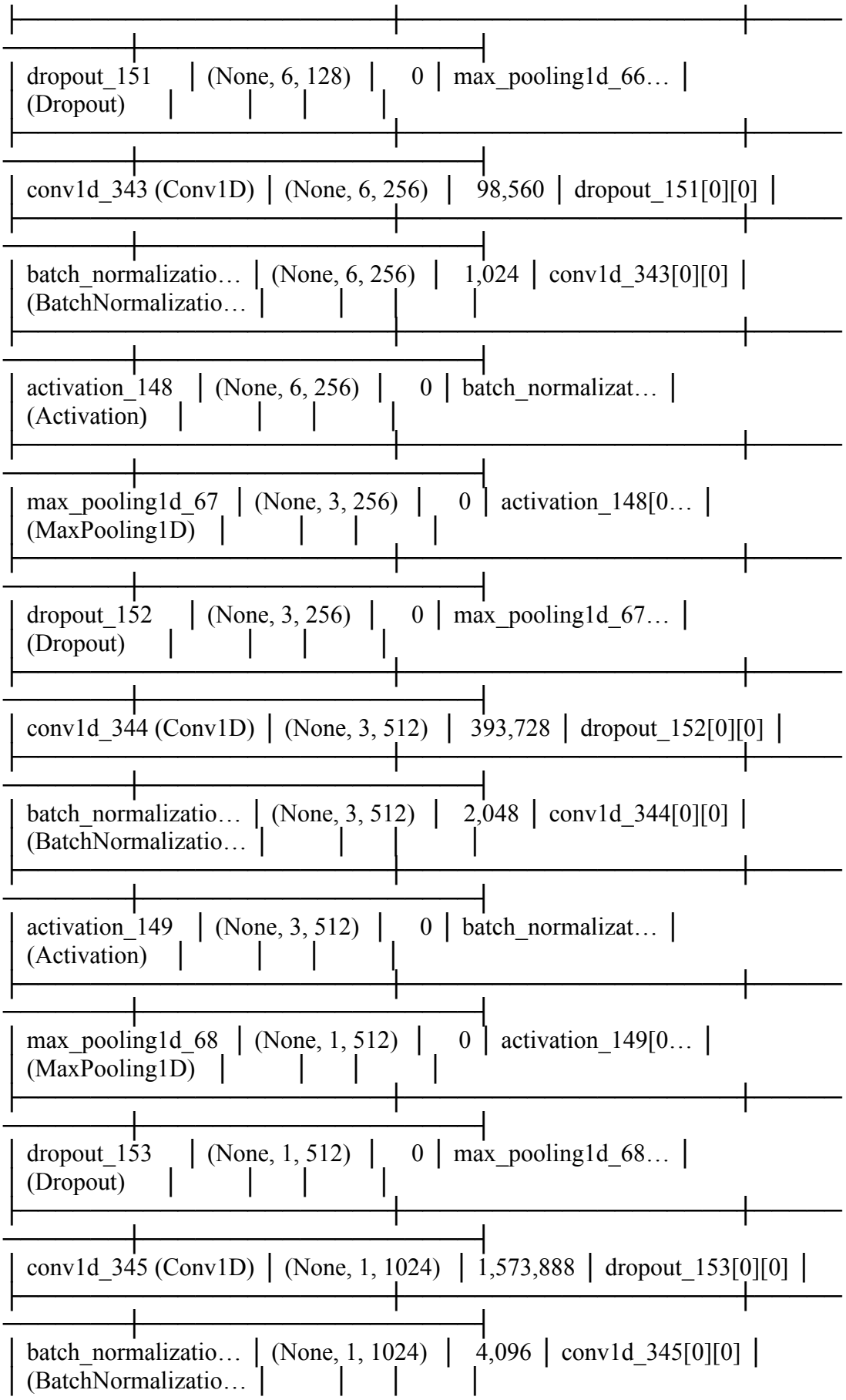
Total params: 3,881,601 (14.81 MB)
Trainable params: 3,877,377 (14.79 MB)
Non-trainable params: 4,224 (16.50 KB)

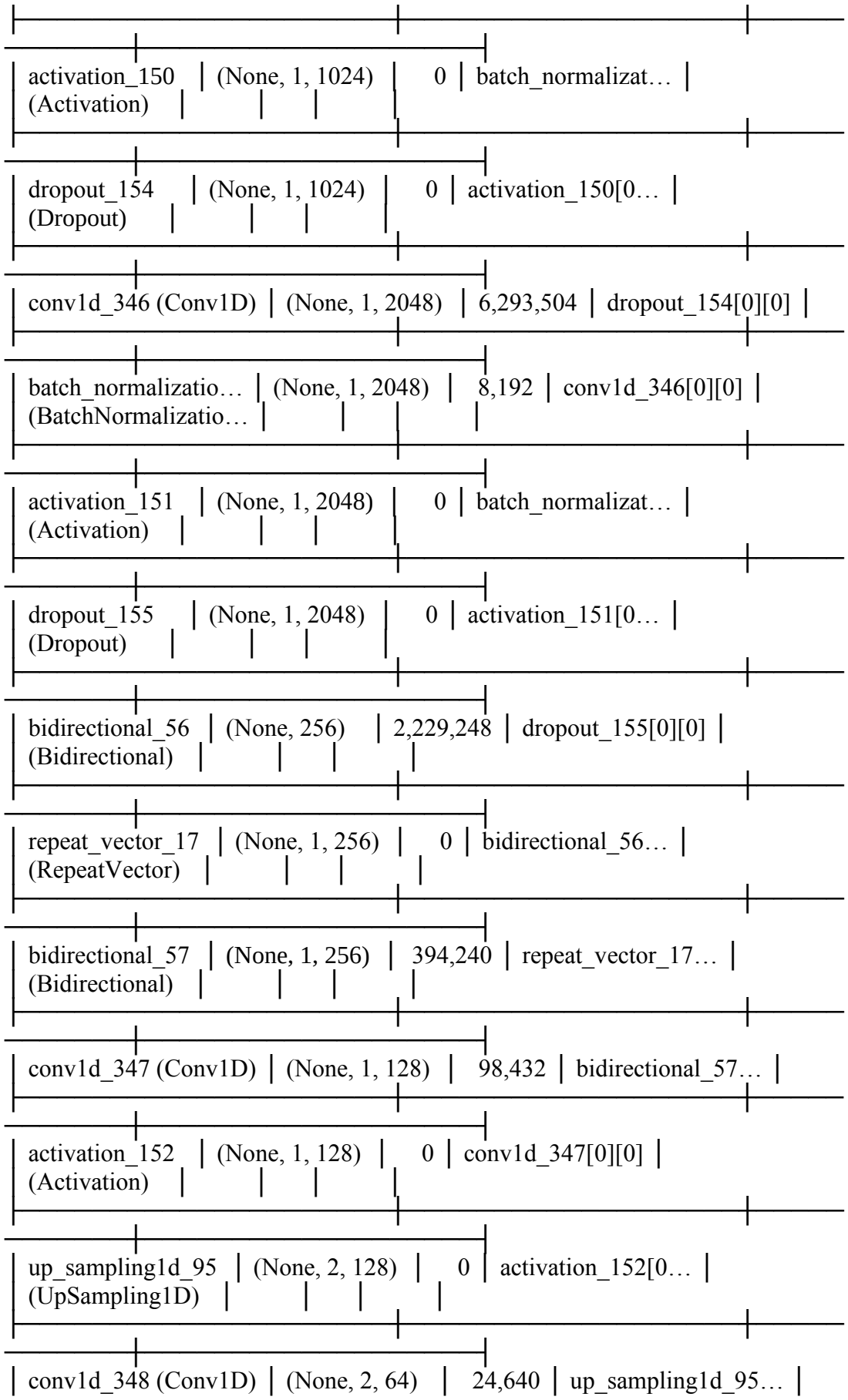
6 hidden layers of MLP-Enhanced Model

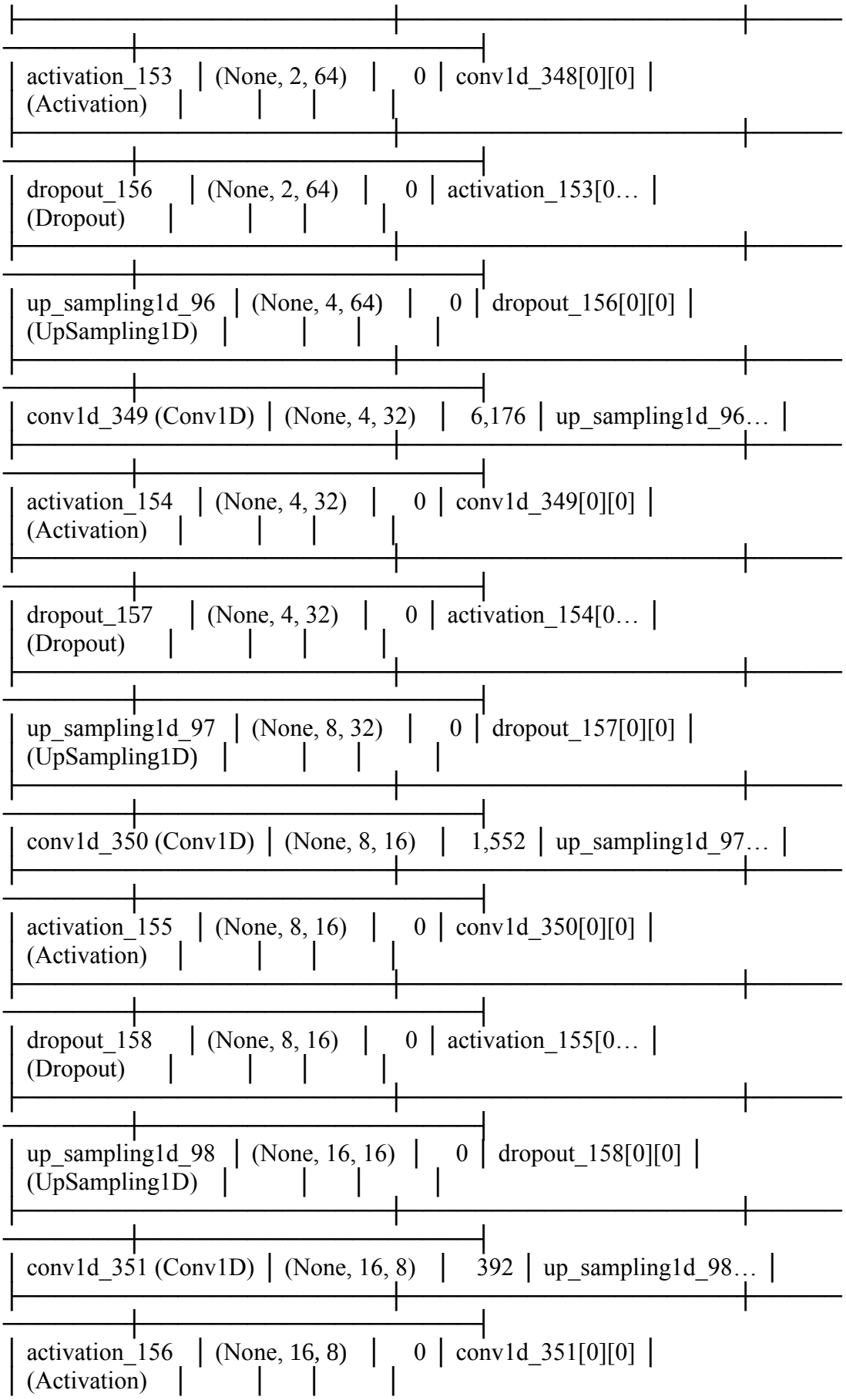
Deep Autoencoder Model with Nadam Optimizer, Regularization, and Batch Optimization

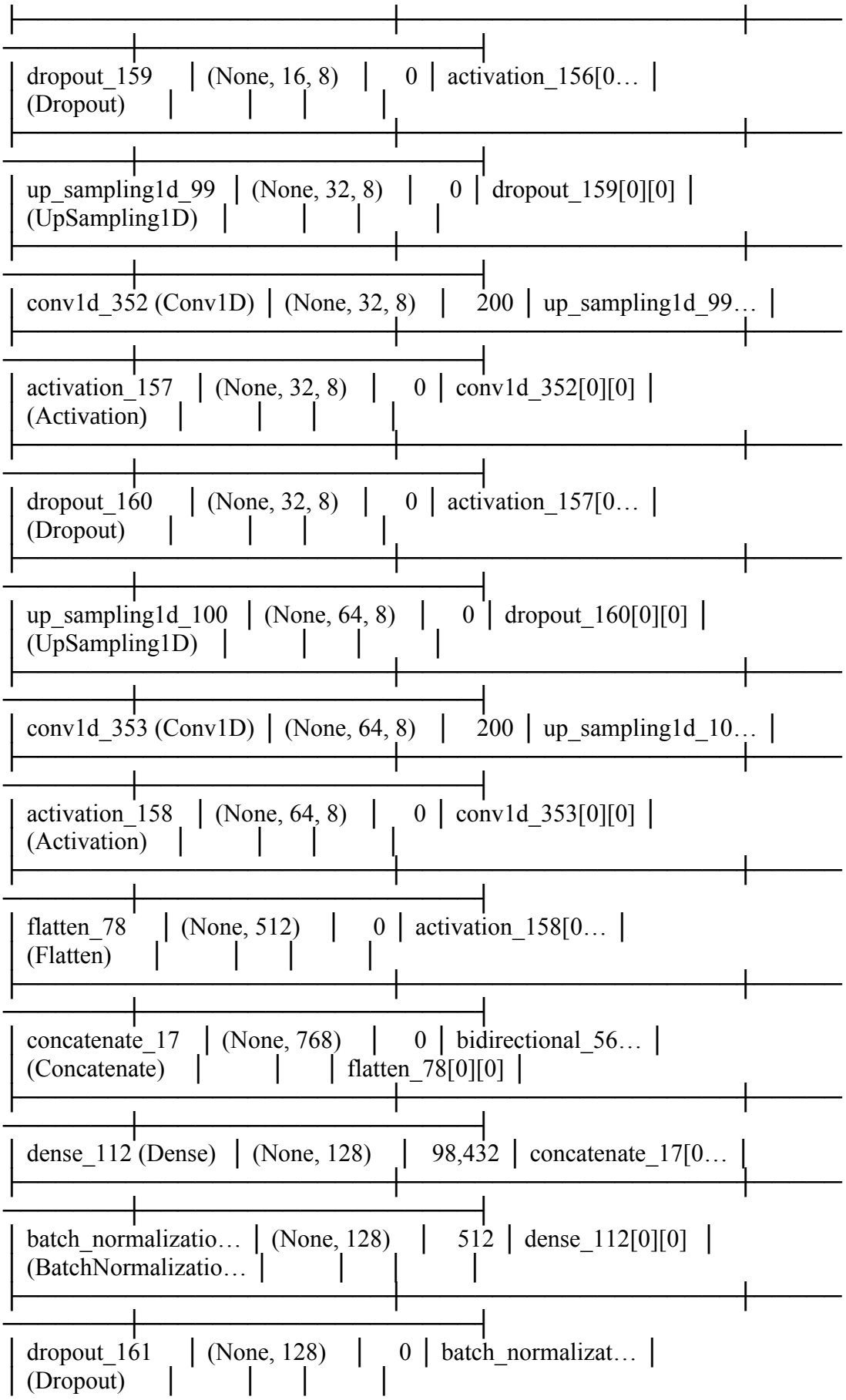
Model: "functional_692"

Layer (type)	Output Shape	Param #	Connected to
input_layer_38 (InputLayer)	(None, 25, 10)	0	-
conv1d_341 (Conv1D)	(None, 25, 64)	1,984	input_layer_38[0…
batch_normalizatio… (BatchNormalizatio…	(None, 25, 64)	256	conv1d_341[0][0]
activation_146 (Activation)	(None, 25, 64)	0	batch_normalizat…
max_pooling1d_65 (MaxPooling1D)	(None, 12, 64)	0	activation_146[0…
dropout_150 (Dropout)	(None, 12, 64)	0	max_pooling1d_65…
conv1d_342 (Conv1D)	(None, 12, 128)	24,704	dropout_150[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 12, 128)	512	conv1d_342[0][0]
activation_147 (Activation)	(None, 12, 128)	0	batch_normalizat…
max_pooling1d_66 (MaxPooling1D)	(None, 6, 128)	0	activation_147[0…


dropout_151 (Dropout) (None, 6, 128) 0 max_pooling1d_66…
conv1d_343 (Conv1D) (None, 6, 256) 98,560 dropout_151[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 6, 256) 1,024 conv1d_343[0][0]
activation_148 (Activation) (None, 6, 256) 0 batch_normalizat…
max_pooling1d_67 (MaxPooling1D) (None, 3, 256) 0 activation_148[0…
dropout_152 (Dropout) (None, 3, 256) 0 max_pooling1d_67…
conv1d_344 (Conv1D) (None, 3, 512) 393,728 dropout_152[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 3, 512) 2,048 conv1d_344[0][0]
activation_149 (Activation) (None, 3, 512) 0 batch_normalizat…
max_pooling1d_68 (MaxPooling1D) (None, 1, 512) 0 activation_149[0…
dropout_153 (Dropout) (None, 1, 512) 0 max_pooling1d_68…
conv1d_345 (Conv1D) (None, 1, 1024) 1,573,888 dropout_153[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 1, 1024) 4,096 conv1d_345[0][0]


activation_150 (Activation) (None, 1, 1024) 0 batch_normalizat…
dropout_154 (Dropout) (None, 1, 1024) 0 activation_150[0…
conv1d_346 (Conv1D) (None, 1, 2048) 6,293,504 dropout_154[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 1, 2048) 8,192 conv1d_346[0][0]
activation_151 (Activation) (None, 1, 2048) 0 batch_normalizat…
dropout_155 (Dropout) (None, 1, 2048) 0 activation_151[0…
bidirectional_56 (Bidirectional) (None, 256) 2,229,248 dropout_155[0][0]
repeat_vector_17 (RepeatVector) (None, 1, 256) 0 bidirectional_56…
bidirectional_57 (Bidirectional) (None, 1, 256) 394,240 repeat_vector_17…
conv1d_347 (Conv1D) (None, 1, 128) 98,432 bidirectional_57…
activation_152 (Activation) (None, 1, 128) 0 conv1d_347[0][0]
up_sampling1d_95 (UpSampling1D) (None, 2, 128) 0 activation_152[0…
conv1d_348 (Conv1D) (None, 2, 64) 24,640 up_sampling1d_95…


activation_153 (Activation) | (None, 2, 64) | 0 | conv1d_348[0][0]
dropout_156 (Dropout) | (None, 2, 64) | 0 | activation_153[0…
up_sampling1d_96 (UpSampling1D) | (None, 4, 64) | 0 | dropout_156[0][0]
conv1d_349 (Conv1D) | (None, 4, 32) | 6,176 | up_sampling1d_96…
activation_154 (Activation) | (None, 4, 32) | 0 | conv1d_349[0][0]
dropout_157 (Dropout) | (None, 4, 32) | 0 | activation_154[0…
up_sampling1d_97 (UpSampling1D) | (None, 8, 32) | 0 | dropout_157[0][0]
conv1d_350 (Conv1D) | (None, 8, 16) | 1,552 | up_sampling1d_97…
activation_155 (Activation) | (None, 8, 16) | 0 | conv1d_350[0][0]
dropout_158 (Dropout) | (None, 8, 16) | 0 | activation_155[0…
up_sampling1d_98 (UpSampling1D) | (None, 16, 16) | 0 | dropout_158[0][0]
conv1d_351 (Conv1D) | (None, 16, 8) | 392 | up_sampling1d_98…
activation_156 (Activation) | (None, 16, 8) | 0 | conv1d_351[0][0]

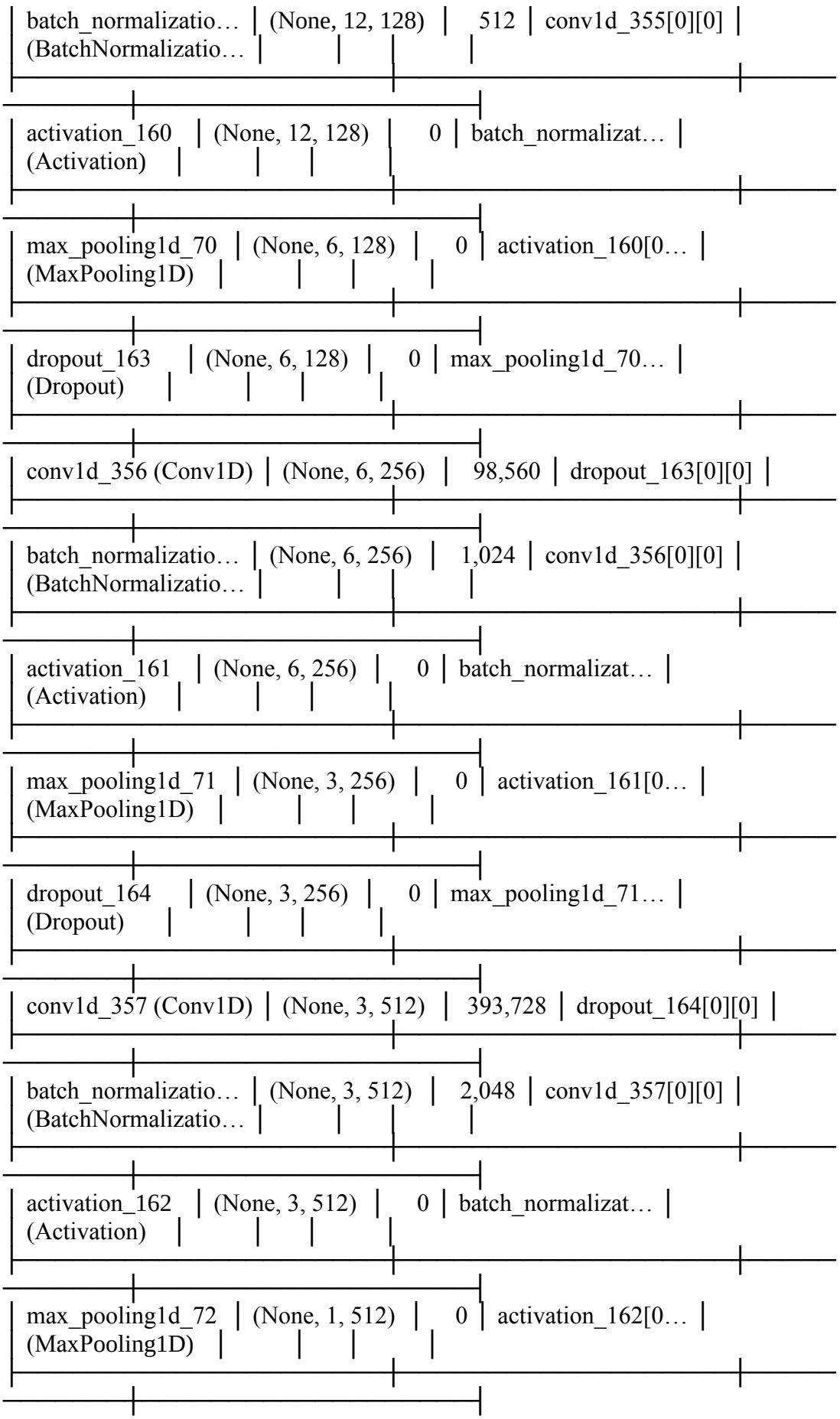

dropout_159 (Dropout) | (None, 16, 8) | 0 | activation_156[0…
up_sampling1d_99 (UpSampling1D) | (None, 32, 8) | 0 | dropout_159[0][0]
conv1d_352 (Conv1D) | (None, 32, 8) | 200 | up_sampling1d_99…
activation_157 (Activation) | (None, 32, 8) | 0 | conv1d_352[0][0]
dropout_160 (Dropout) | (None, 32, 8) | 0 | activation_157[0…
up_sampling1d_100 (UpSampling1D) | (None, 64, 8) | 0 | dropout_160[0][0]
conv1d_353 (Conv1D) | (None, 64, 8) | 200 | up_sampling1d_10…
activation_158 (Activation) | (None, 64, 8) | 0 | conv1d_353[0][0]
flatten_78 (Flatten) | (None, 512) | 0 | activation_158[0…
concatenate_17 (Concatenate) | (None, 768) | 0 | bidirectional_56… | flatten_78[0][0]
dense_112 (Dense) | (None, 128) | 98,432 | concatenate_17[0…
batch_normalizatio… (BatchNormalizatio… | (None, 128) | 512 | dense_112[0][0]
dropout_161 (Dropout) | (None, 128) | 0 | batch_normalizat…

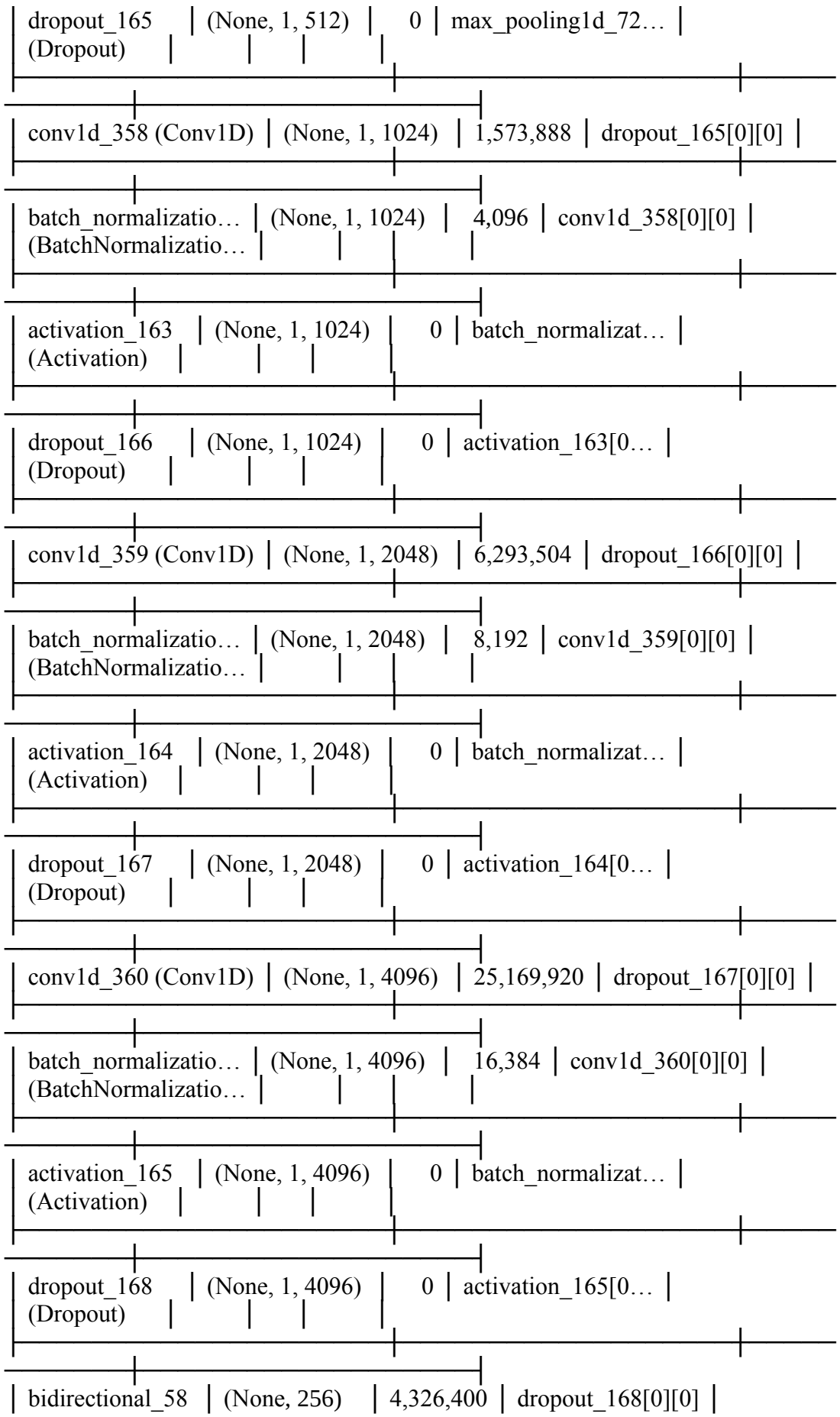
dense_113 (Dense)	(None, 64)	8,256	dropout_161[0][0]
dense_114 (Dense)	(None, 1)	65	dense_113[0][0]

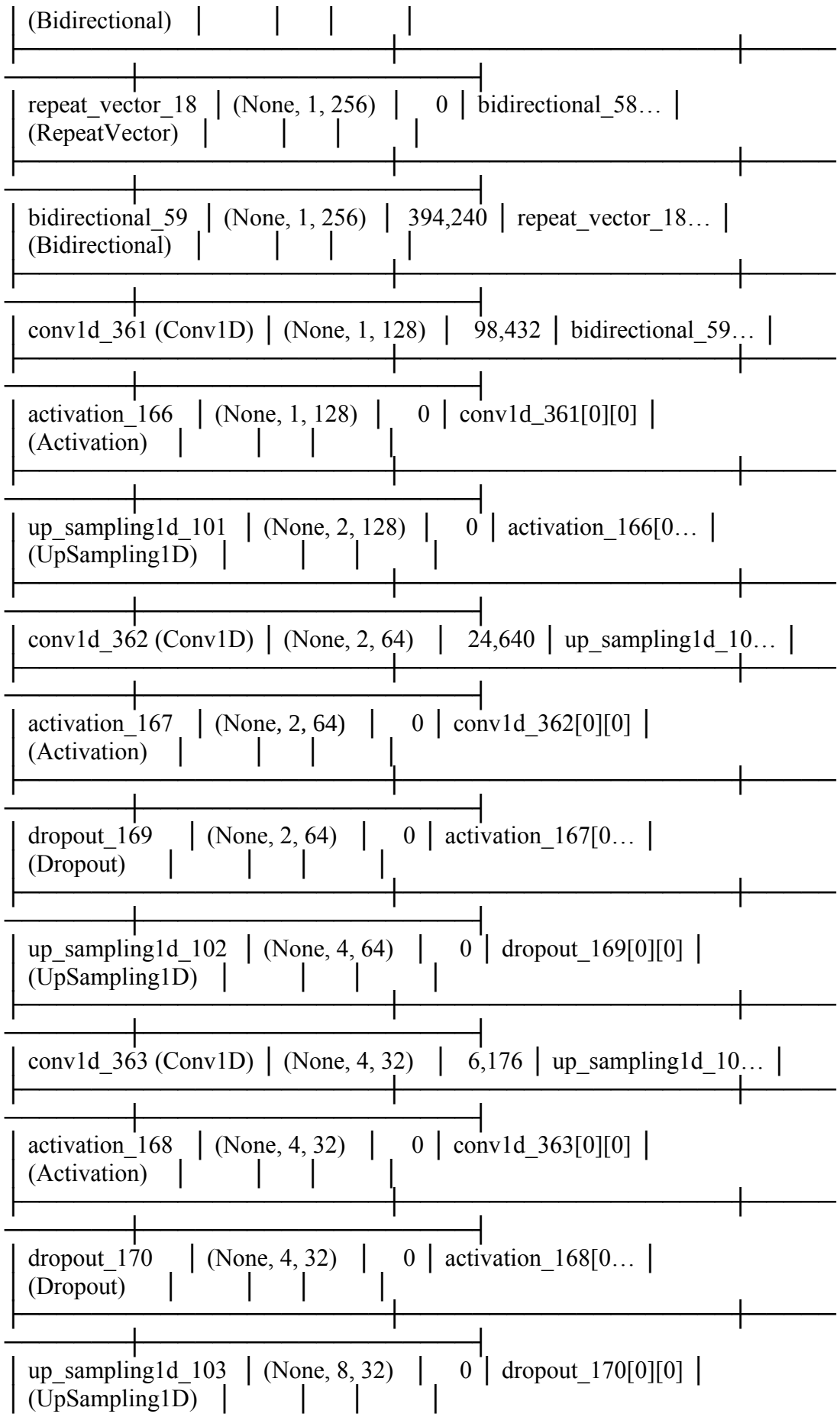
Total params: 11,264,841 (42.97 MB)
Trainable params: 11,256,521 (42.94 MB)
Non-trainable params: 8,320 (32.50 KB)

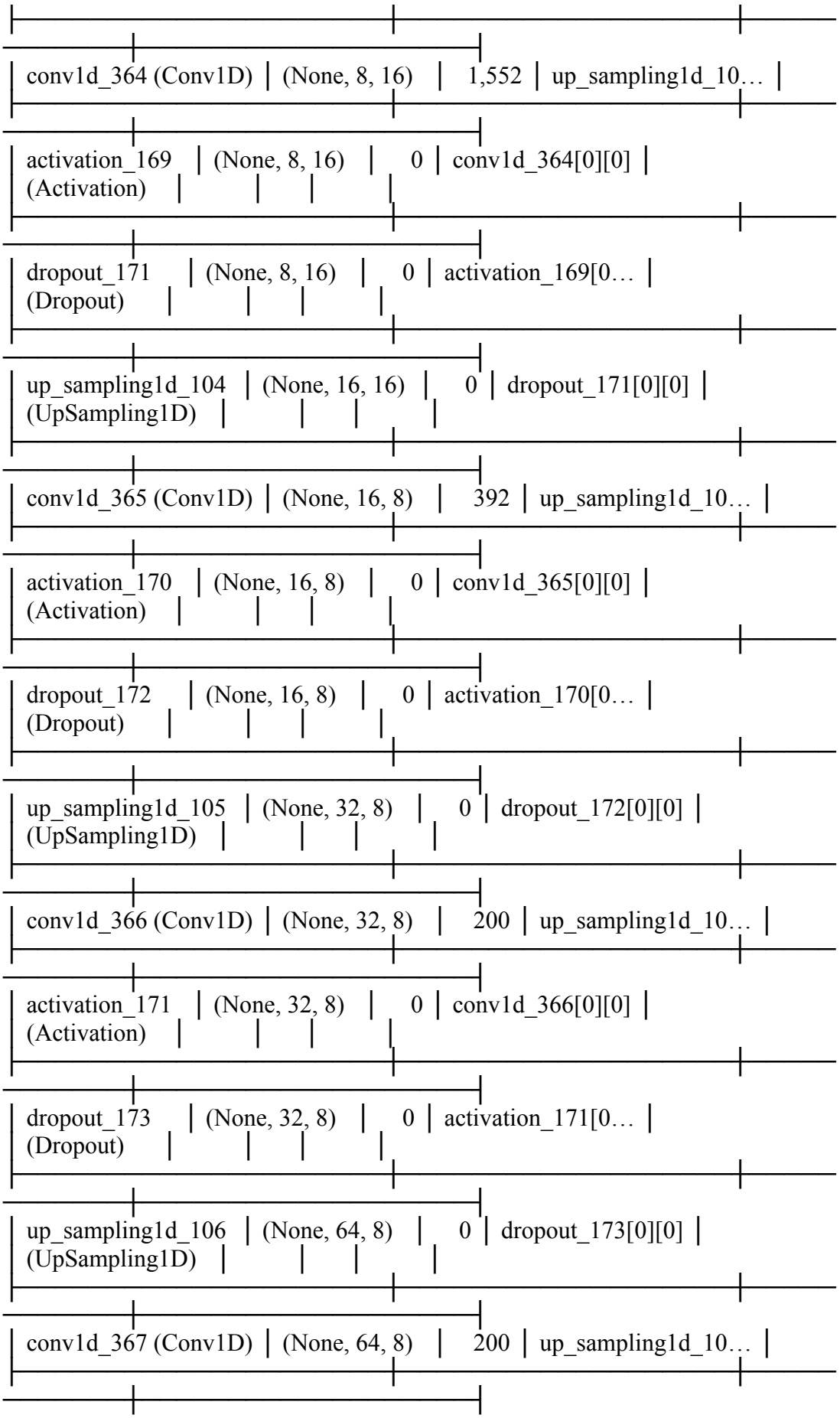
7 hidden layers of MLP-Enhanced Model
Deep Autoencoder Model with Nadam Optimizer, Regularization, and Batch Optimization
Model: "functional_693"

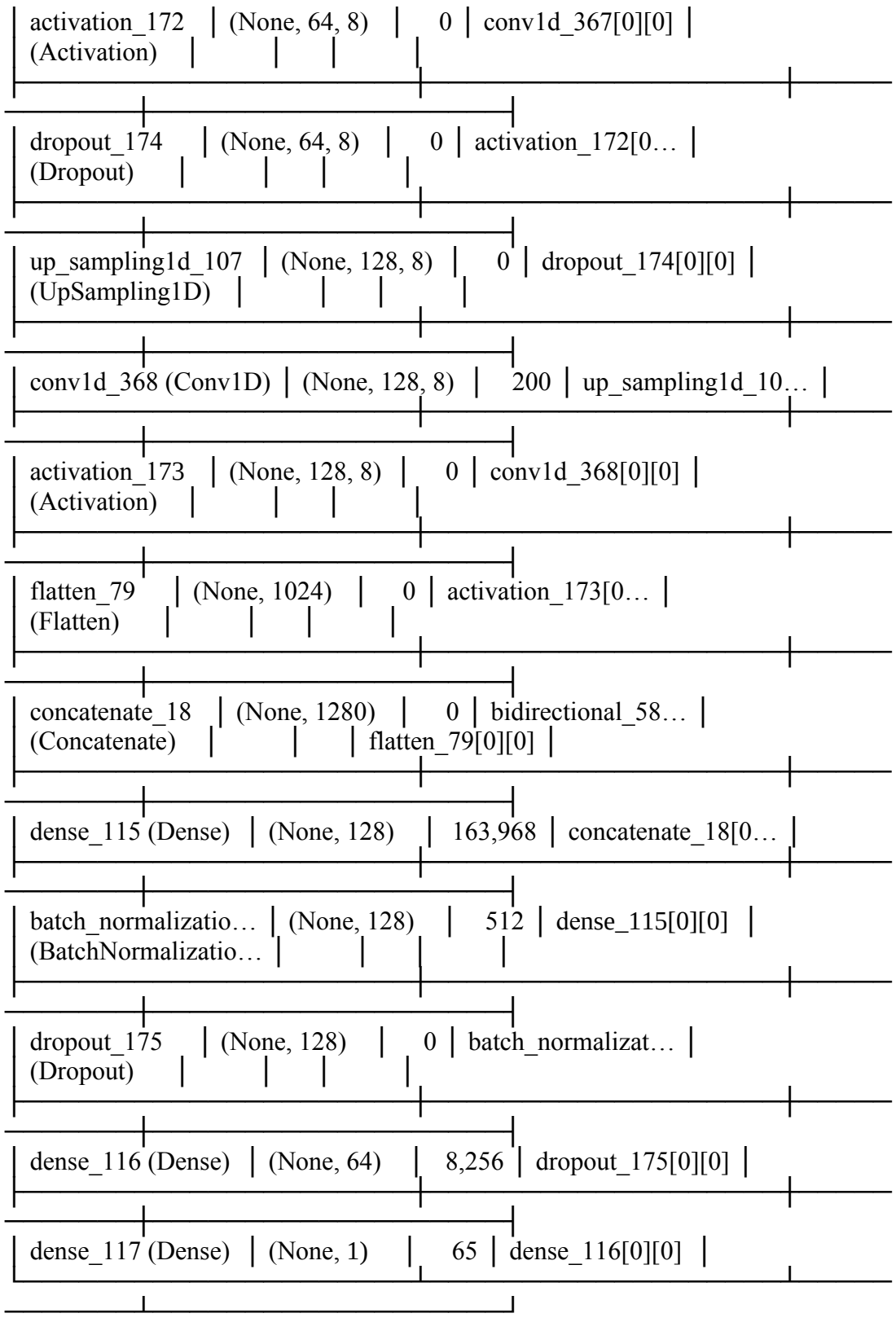
Layer (type)	Output Shape	Param #	Connected to
input_layer_39 (InputLayer)	(None, 25, 10)	0	-
conv1d_354 (Conv1D)	(None, 25, 64)	1,984	input_layer_39[0…
batch_normalizatio… (BatchNormalizatio…	(None, 25, 64)	256	conv1d_354[0][0]
activation_159 (Activation)	(None, 25, 64)	0	batch_normalizat…
max_pooling1d_69 (MaxPooling1D)	(None, 12, 64)	0	activation_159[0…
dropout_162 (Dropout)	(None, 12, 64)	0	max_pooling1d_69…
conv1d_355 (Conv1D)	(None, 12, 128)	24,704	dropout_162[0][0]


batch_normalizatio… (BatchNormalizatio… (None, 12, 128) 512 conv1d_355[0][0]
activation_160 (Activation) (None, 12, 128) 0 batch_normalizat…
max_pooling1d_70 (MaxPooling1D) (None, 6, 128) 0 activation_160[0…
dropout_163 (Dropout) (None, 6, 128) 0 max_pooling1d_70…
conv1d_356 (Conv1D) (None, 6, 256) 98,560 dropout_163[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 6, 256) 1,024 conv1d_356[0][0]
activation_161 (Activation) (None, 6, 256) 0 batch_normalizat…
max_pooling1d_71 (MaxPooling1D) (None, 3, 256) 0 activation_161[0…
dropout_164 (Dropout) (None, 3, 256) 0 max_pooling1d_71…
conv1d_357 (Conv1D) (None, 3, 512) 393,728 dropout_164[0][0]
batch_normalizatio… (BatchNormalizatio… (None, 3, 512) 2,048 conv1d_357[0][0]
activation_162 (Activation) (None, 3, 512) 0 batch_normalizat…
max_pooling1d_72 (MaxPooling1D) (None, 1, 512) 0 activation_162[0…


dropout_165 (Dropout) | (None, 1, 512) | 0 | max_pooling1d_72…
conv1d_358 (Conv1D) | (None, 1, 1024) | 1,573,888 | dropout_165[0][0]
batch_normalizatio… (BatchNormalizatio… | (None, 1, 1024) | 4,096 | conv1d_358[0][0]
activation_163 (Activation) | (None, 1, 1024) | 0 | batch_normalizat…
dropout_166 (Dropout) | (None, 1, 1024) | 0 | activation_163[0…
conv1d_359 (Conv1D) | (None, 1, 2048) | 6,293,504 | dropout_166[0][0]
batch_normalizatio… (BatchNormalizatio… | (None, 1, 2048) | 8,192 | conv1d_359[0][0]
activation_164 (Activation) | (None, 1, 2048) | 0 | batch_normalizat…
dropout_167 (Dropout) | (None, 1, 2048) | 0 | activation_164[0…
conv1d_360 (Conv1D) | (None, 1, 4096) | 25,169,920 | dropout_167[0][0]
batch_normalizatio… (BatchNormalizatio… | (None, 1, 4096) | 16,384 | conv1d_360[0][0]
activation_165 (Activation) | (None, 1, 4096) | 0 | batch_normalizat…
dropout_168 (Dropout) | (None, 1, 4096) | 0 | activation_165[0…
bidirectional_58 | (None, 256) | 4,326,400 | dropout_168[0][0]


(Bidirectional)
repeat_vector_18 (RepeatVector) (None, 1, 256) 0 bidirectional_58…
bidirectional_59 (Bidirectional) (None, 1, 256) 394,240 repeat_vector_18…
conv1d_361 (Conv1D) (None, 1, 128) 98,432 bidirectional_59…
activation_166 (Activation) (None, 1, 128) 0 conv1d_361[0][0]
up_sampling1d_101 (UpSampling1D) (None, 2, 128) 0 activation_166[0…
conv1d_362 (Conv1D) (None, 2, 64) 24,640 up_sampling1d_10…
activation_167 (Activation) (None, 2, 64) 0 conv1d_362[0][0]
dropout_169 (Dropout) (None, 2, 64) 0 activation_167[0…
up_sampling1d_102 (UpSampling1D) (None, 4, 64) 0 dropout_169[0][0]
conv1d_363 (Conv1D) (None, 4, 32) 6,176 up_sampling1d_10…
activation_168 (Activation) (None, 4, 32) 0 conv1d_363[0][0]
dropout_170 (Dropout) (None, 4, 32) 0 activation_168[0…
up_sampling1d_103 (UpSampling1D) (None, 8, 32) 0 dropout_170[0][0]


conv1d_364 (Conv1D) (None, 8, 16) 1,552 up_sampling1d_10…
activation_169 (Activation) (None, 8, 16) 0 conv1d_364[0][0]
dropout_171 (Dropout) (None, 8, 16) 0 activation_169[0…
up_sampling1d_104 (UpSampling1D) (None, 16, 16) 0 dropout_171[0][0]
conv1d_365 (Conv1D) (None, 16, 8) 392 up_sampling1d_10…
activation_170 (Activation) (None, 16, 8) 0 conv1d_365[0][0]
dropout_172 (Dropout) (None, 16, 8) 0 activation_170[0…
up_sampling1d_105 (UpSampling1D) (None, 32, 8) 0 dropout_172[0][0]
conv1d_366 (Conv1D) (None, 32, 8) 200 up_sampling1d_10…
activation_171 (Activation) (None, 32, 8) 0 conv1d_366[0][0]
dropout_173 (Dropout) (None, 32, 8) 0 activation_171[0…
up_sampling1d_106 (UpSampling1D) (None, 64, 8) 0 dropout_173[0][0]
conv1d_367 (Conv1D) (None, 64, 8) 200 up_sampling1d_10…



Total params: 38,614,033 (147.30 MB)
Trainable params: 38,597,521 (147.24 MB)
Non-trainable params: 16,512 (64.50 KB)

Appendix 5: Research Authorization Letter

CHUKA UNIVERSITY

Knowledge is Wealth *(Sapientia divitia est)* Akili ni Mali

CHUKA UNIVERSITY INSTITUTIONAL ETHICS REVIEW COMMITTEE

Telephones: 020-2310512/18
Direct Line: 0772894438
Email: info@chuka.ac.ke,
P. O. Box 109-60400, Chuka
Website: www.chuka.ac.ke

1st April, 2025

REF: CUIERC/ NACOSTI/698
TO: **James Kirimi**

RE: Hybrid of Deep Autoencoders, Convolutional Neural Network and Long Short-Term Memory for Intrusion Detection in Cloud-based implantable Medical Devices

This is to inform you that *Chuka University IERC* has reviewed and approved your above research proposal. Your application approval number is *NACOSTI/NBC/AC-0812.* The approval period is 1st April, 2025 – 1st April, 2026.

This approval is subject to compliance with the following requirements;

i. Only approved documents including (informed consents, study instruments, MTA) will be used
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by *Chuka University IERC.*
iii. Death and life threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to *Chuka University IERC* within 72 hours of notification
iv. Any changes, anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to *Chuka University IERC* within 72 hours
v. Clearance for export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days upon completion of the study to *Chuka University IERC.*

Prior to commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology and Innovation (NACOSTI) https://oris.nacosti.go.ke and also obtain other clearances needed.

Yours sincerely

Dr. Benjamin Kanga
SECRETARY

INSTITUTIONAL ETHICS REVIEW COMMITTEE
01 APR 2025
P.O. Box 109-60400, CHUKA
CHUKA UNIVERSITY

Appendix 6: National Commission for Science, Technology and Innovation (NACOSTI)Research License

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: **977806**

Date of Issue: **21/June/2025**

RESEARCH LICENSE

This is to Certify that Mr.. James Kirimi of Chuka University, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act, 2013 (Rev.2014) in Tharaka-Nithi on the topic: Hybrid of deep autoencoders, convolutional neural network, and long short-term memory for intrusion detection in cloud-based implantable medical devices for the period ending : 21/June/2026.

License No: **NACOSTI/P/25/4175582**

977806

Applicant Identification Number

Deputy Director
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

See overleaf for conditions