Advancing Healthcare Monitoring: Anomaly Detection through Hybrid Deep Learning for Enhanced Connectivity in Software Defined Networking

Leo Prasanth Lourdu Antony

lleo1306@gmail.com

Anna University, Chennai

Uma Elangovan

Anna University, Chennai

Research Article

Keywords: Healthcare, Anomaly Detection, Deep Neural Networks (DNN), Recurrent Neural Networks (RNN) and Software Defined Networking (SDN)

Posted Date: June 13th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-4502587/v1

Additional Declarations: No competing interests reported.

Abstract

In the realm of healthcare, the continuous evolution of monitoring systems demands innovative solutions to ensure heightened reliability and accuracy. This paper introduces a pioneering approach to healthcare monitoring through a hybrid deep learning model that combines the advantages of recurrent neural networks (RNN) and deep neural networks (DNN). Focused on enhancing connectivity in Software Defined Networking (SDN), our framework places a significant emphasis on anomaly detection for improved predictive accuracy. The proposed Hybrid Deep Learning model is meticulously designed to harness the complementary features of DNN and RNN, enabling the system to capture both spatial and temporal dependencies in healthcare data. This integration enhances the precision of anomaly detection, allowing for the identification of subtle deviations from normal patterns with unprecedented accuracy. Key to our methodology is the adaptability of Software Defined Networking, providing a flexible and programmable infrastructure. The Hybrid Deep Learning model operates seamlessly within this SDN framework, dynamically optimizing resource allocation and traffic patterns to accommodate the unique demands of healthcare monitoring. Through extensive experimentation and validation, our framework demonstrates remarkable predictive accuracy in identifying anomalies within healthcare data streams. Comparative analyses against traditional anomaly detection methods underscore the superiority of our approach, showcasing its efficacy in real-world healthcare scenarios. In conclusion, our research contributes to the advancement of healthcare monitoring by introducing a Hybrid Deep Learning model, combining DNN and RNN architectures, within the context of Software Defined Networking. The achieved high prediction accuracy in anomaly detection signifies a significant leap forward in the reliability and precision of healthcare monitoring systems, paving the way for more robust and responsive healthcare networks.

1. Introduction

SDNs have been widely used in a variety of fields in recent years, primarily as a result of their benefits as dependable network technologies that enable controlling and dividing the control and data planes in order to manage a network. Compared to traditional networks, where the network only has application awareness, the SDN architecture provides more information about the state of the entire network, from the controller to its applications. Technological developments are bringing about a radical change in the healthcare landscape, with a special emphasis on the creation of reliable monitoring systems. This work explores the core of this development, introducing a novel method of healthcare monitoring within the framework of Software Defined Networking (SDN). In order to advance anomaly detection and improve connectivity in healthcare networks, we present a Hybrid Deep Learning model that combines the strengths of Recurrent Neural Networks (RNN) and Deep Neural Networks (DNN).

More than ever, the need for accurate and dependable healthcare monitoring systems is essential as medical professionals look for real-time insights to make prompt interventions. This need is met by incorporating a hybrid deep learning model, which makes use of the temporal and spatial dependencies present in healthcare data to enable more accurate and nuanced anomaly detection. Our model is

integrated into the programmable and adaptable Software Defined Networking framework, allowing it to dynamically adjust to the constantly changing needs of healthcare environments. In addition to enhancing anomaly detection capabilities, the combination of hybrid deep learning and SDN optimizes resource allocation and traffic patterns to guarantee responsive and effective connectivity. The paper explains the nuances of our Hybrid Deep Learning model as we go through this exploration, highlighting its adaptability and its significance in changing the paradigms of healthcare monitoring. The architecture and methodology are covered in detail in the following sections, which also demonstrate how this creative solution advances healthcare connectivity. We establish the superior predictive accuracy of our model through comparative analyses and empirical validations, thereby confirming its transformative role in the healthcare monitoring domain. "Advancing Healthcare Monitoring: Anomaly Detection through Hybrid Deep Learning for Enhanced Connectivity in Software Defined Networking" summarizes a technological revolution in the healthcare industry. Combining cutting-edge deep learning methods with SDN's adaptable structure not only increases anomaly detection accuracy but also sets the stage for a time when precision, responsiveness, and flexibility will be synonymous with healthcare monitoring.

The research contribution of the proposed framework is listed below

- The proposed work explored the efficiency of employing a hybrid deep learning model to analyze and detect the anomalies in the network.
- The literature survey of anomaly detection and SDN provides the brief understanding for implementing the anomaly detection through hybrid deep learning model in SDN.
- The approach model incorporates with DNN and LSTM which provides additional strength to improve the overall performance
- The proposed model contributes the Healthcare professionals can detect possible health problems or security breaches in real time with the aid of hybrid deep learning anomaly detection.

The rest of the article is organized as follows: the section 2 presents the literature survey of anomaly detection in SDN. The section 3 provides the proposed architecture of the hybrid deep learning system, the section 4 gives the experimental setup and results of the system. The result and discussion are discussed in section 5 and the conclusion of the article is provided in section 6.

2. Literature Review

Aiguo et.al (Chen et al., 2022) proposed a model with Machine learning and deep learning methods which improves network behavior anomaly detection (NBAD), but existing methods are inflexible and low-accuracy. An efficient NBAD algorithm based on deep belief networks and long short-term memory networks is proposed. This method extracts features automatically, reduces data dimension, and uses a light-structure LSTM network for classification. Ismal et.al (Valdovinos et al., 2021) This paper surveys DDoS detection and mitigation strategies in SDN, analysing existing approaches like statistical, architecture, and machine learning, and emerging ones like network function virtualization, blockchain, honeynet, and moving target defence. Mazin (Alshamrani, 2022) evaluates the study of RHM services

technologies and systems, analysing monitoring applications using various IoT-based sensors, highlighting limitations and suggesting potential opportunities in this research area. Kashif et.al (Qureshi et al., 2021) proposed an anomaly detection system for SDN and edge computing networks, and a Trusted Authority for Edge Computing (TA-Edge) to ensure edge devices' trust for data forwarding. The model verifies certificates once, overcoming edge device overhead. Simulation results show improved performance.

Erhan et.al (Erhan et al., 2021) provides the review on anomaly detection which is a challenging problem requiring computing-energy accuracy trade-offs. Methods range from conventional to data-driven, and architectural environments impact sensors ecosystem. The review highlights promising intelligent-sensing methods and open issues. Bizhu et.al (Wang et al., 2018) introduced a Localized Evolving Semi-supervised Learning Based Anomaly detection scheme (LESLA) for wireless mMTC devices, combining offline training and online testing. It employs semi-supervised learning and contrastive probabilistic likelihood estimation for self-evolving anomaly detection, demonstrating its efficiency and advantages. Balaram et.al (Sharma et al., 2020) proposed approach uses LSTM-based Autoencoder to model user behavior and identify anomalous data points. It calculates reconstruction errors and defines thresholds to separate outliers from normal data points. The CERT insider threat dataset is used for research. The model produces high reconstruction errors for unseen behavior or anomaly patterns, with experimental results showing an accuracy of 90.17%, 91.03%, and 9.84%.

Elsayed (Said Elsayed et al., 2020) used Long Short Term Memory (LSTM) autoencoder and one-class Support Vector Machine (OC-SVM) is proposed to detect anomalies-based attacks in unbalanced datasets. The model learns normal traffic patterns and compressed input data, overcoming the limitations of separate OC-SVM. Experiments show higher detection rates and reduced processing time, enhancing security in SDN environments. Bizhu et.al (Wang et al., 2021) proposed a localized ADS scheme called scalable and energy-efficient anomaly detection scheme (SEEADS), which consists of detection activation, lightweight prediction, heavyweight detection, and dynamic strategy selection modules. The scheme reduces energy consumption and shows higher sensitivity on abnormal packets, compared to literature work, and utilizes feedback from previous heavyweight activation and predetection indications. Syed et.al (Shah et al., 2021) explored the use of MEC and network slicing in 5G service-focused use cases, focusing on cloud-native 5G core changes, MEC use cases, cloud-native micro services architecture, E2E network slicing advances, and open research issues.

Wajid et.al (Rafique et al., 2020) explored the use of Software-Defined Internet of Things (SDIoT) orchestration using Edge (SDIoT-Edge) in managing complex IoT systems. It discusses key requirements, standardization efforts, case studies, performance parameters, security vulnerabilities, attack possibilities, lessons learned, and future research directions for efficient IoT service provision in the SDIoT-Edge paradigm. Samrat and RahmanClick or tap here to enter text. Dey and Rahman proposed a network intrusion detection system using Gated Recurrent Unit Long Short Term Memory (GRU-LSTM) and ANOVA F-Test and Recursive feature elimination methods. Tested on NSL-KDD, it achieves 87% accuracy and shows potential for flow-based anomaly detection in OpenFlow controllers. Qaisar et.al

(Shafi et al., 2018) introduced a fog-assisted intrusion detection/prevention system (IDPS) for IoT networks, utilizing fog computational arrangement for real-time attack identification and threat neutralization using SDN control.

Malik et.al (Malik et al., 2020) proposed a control plane-based orchestration for sophisticated threats using a hybrid Cuda-enabled DL-driven architecture. This method uses long short-term memory and Convolutional Neural Network for efficient detection. The mechanism outperforms other hybrid DL architectures and benchmark algorithms, with unbiased results after 10-fold cross validation. Majd and Toker (Latah & Toker, 2019) explored a recent AI integration in Smart Data Networks (SDN) through three main sub-fields: machine learning, meta-heuristics, and fuzzy inference systems, examining their application areas and potential improvements. Rajat and Neeraj (Chaudhary & Kumar, 2019) proposed LOADS scheme on Mininet emulator reduces average execution time by 6.74% and 20.64% compared to existing schemes, DHA and Elastic Distributed Controller. It also improves migration cost and response time, with a migration cost of 55.1 milliseconds and a response time of 32.8 milliseconds.

Laila et.al (Halman &Alenazi, 2023) The study proposes a machine learning-based cyberattack detector (MCAD) for healthcare systems, utilizing a layer three learning switch application to collect traffic and deploy it on the Ryu controller. The MCAD shows high reliability and performance, achieving an F1-score of 0.9998 and 0.9882 on normal and attack classes. It also improves network KPIs by increasing throughput by 609% and decreasing delay and jitter. Panagiotis et.al(Radoglou-Grammatikis et al., 2022) This article examines the IEC 60 870-5-104 protocol in industrial healthcare systems. It provides a quantitative threat model and introduces an intrusion detection and prevention system (IDPS) that uses machine learning and software defined networking technologies to detect and mitigate cyberattacks. The IDPS achieves detection accuracy of 0.831 and F1 score of 0.8258, and mitigation accuracy of 0.923. Rohit and Agrawal (Kumar & Agrawal, 2024) explored surveillance in various areas, providing readers with a better understanding of SDN-based surveillance and addressing open research problems and related challenges. Nirav and Sudhir (Raja &Vegad, 2023) examines 50 probe papers on traffic flow rate prediction-based anomaly detection in software-defined networking (SDN). It presents technique-wise classifications, including flow counting, information theory, entropy, deep learning, hybrid methods, and network methods. The limitations of these techniques are discussed.

Azka and Revathi (Wani & Revathi, 2020) discussed DDoS attacks in IoT and introduces a flexible SDN-based method for detecting and mitigating them. It suggests extending the test to larger attacks and implementing DDoS prevention in IoT networks and a strict authentication mechanism. Azka et.al (Wani & Revathi, 2020) proposed phSDN prototype, which supports body temperature, pulse rate, and blood pressure, offers programmability over services, enabling unified control over enddevices. Experiments show it minimizes network response time and scalability, compared to fog and cloud-based approaches. Francesco et.al (Restuccia et al., 2018) presented a taxonomy and survey of IoT security research, highlighting the need for a secure-by-design approach to address existing and next-generation IoT security threats, emphasizing the importance of machine learning and software-defined networking. Shahbaz et.al (Siddiqui et al., 2022) published studies on SDN-based frameworks for IoT management

issues, focusing on fault tolerance, energy management, scalability, load balancing, and security service provisioning. It provides a systematic literature review of studies from 2010–2022, categorizing existing frameworks into network function virtualization, middleware, OpenFlow adaptation, and blockchain-based management. It highlights challenges and promising opportunities for future research. Uakomba et.al (Uhongora et al., n.d.) presented recent developments in cyber security for SDN-based space systems reveal vulnerabilities and threats, discussed the potential of DL-based IDS for threat detection, and identify research gaps and future directions.

3. Proposed Methodology

Relying on the aforementioned discussion in section 2, the proposed novel model called Hybrid Deep learning model to detect and identify anomaly detection in healthcare. This mechanism is incorporated with two model such as DNN and LSTM. The architecture of Hybrid deep learning mechanism depicted in Fig. 1 with the algorithm as presented in Algorithm 1.

3.1 Dataset Collection

Both nearby patients on the hospital's grounds and distant patients (those with Wi-Fi or Cellular Network) were considered by the model. The backend system gateway is the patient's smartphone. The Body Area Network (BAN), which each patient's BAN is connected to via a smartphone, views all of the patient's sensors as nodes. The SDN platform manages these gateways, which could number in the hundreds or even thousands, by dispersing different security rules from specific SDN controllers. After collecting the data, it takes feature engineering process and by the hybrid deep learning model which identifies and detects the anomaly detection as depicted in Fig. 1.

3.2 Data Pre-Processing

Collecting the information from relevant sources, including databases, logs, and sensors. Make sure the data includes both typical and unusual occurrences in order to properly train and assess the model. Address noise, outliers, and missing values in the data. You can either drop rows or columns with missing values or impute missing values using methods like mean, median, and mode imputation, depending on how serious the missing data is. It is possible to recognize and handle outliers and noise using statistical techniques or domain expertise.Determine which features are most important for anomaly detection. This could entail feature importance techniques, exploratory data analysis (EDA), or domain knowledge. Choose or design features that encapsulate the fundamental patterns and traits of both typical and atypical behaviours.

3.3 Feature Engineering

Feature engineering is the process of turning raw data into features suitable for machine learning models. Put another way, it's the process of determining, extracting, and altering the most relevant features from the available data so that machine learning models can be built with greater accuracy and efficiency. The performance of machine learning models is highly dependent on the features that are

used to train them. "Feature engineering" is a set of techniques that let us combine or alter existing features to produce new ones. These techniques help to increase the machine learning model's capacity to derive knowledge from the data by highlighting the most important patterns and connections in the data.

By ensuring that the data is correct, consistent, and clean, these procedures try to set up the dataset for efficient analysis and model training. Developing a strong set of features that can raise the predictive models' generalizability and accuracy is the aim of feature engineering. The model's capacity can be improved to identify significant patterns and insights by converting the unstructured data into a more readable and organized format.

The methodical preparation of the dataset is ensured by this methodology, which is essential for developing dependable and successful predictive models.

3.4 Model Structure

3.4.1 DNN, RNN and LSTM

Through the use of machine learning techniques, such as Deep Neural Networks (DNNs), computers can be trained to perform tasks that would be extremely challenging to accomplish with traditional programming methods. The human brain and its processes served as an inspiration for neural network algorithms. Just like our minds, neural networks are programmed to operate not just by adhering to a predetermined set of rules but also by forecasting outcomes and making inferences from past iterations and experiences.

Multiple layers of nodes make up a neural network; these layers receive input from other layers and generate outputs until a final result is achieved. Any number of hidden layers can exist in neural networks; the complexity increases with the number of node layers. The following are various neural network architectures: 1. Traditional neural network which comprised with 2 or 3 hidden layers whereas the neural network can have up to 150 hidden layers.

Input Layer

The input layer gets the data's input features, X.Assume that X is an input matrix with size m×n, where n denotes the number of features and m is the number of examples.

Hidden Layers

A weighted sum of inputs that have been run through an activation function determines the output of each neuron in a hidden layer.

Let $X^{[l]} = W^{[l]}. Z^{[l-1]} + b^{[l]}$

Where,

$X^{[l]}$presents the weighted sum of input to layers *l*

$Z^{[l-1]}$presents the output of the previous layers and $b^{[l]}$represents the bias vector of the layer.

The activation function $g^{[l]}$ introduces the non-linearity into the network.

$A^{[l]} = g^{[l]}(X^{[l]})$

Output Layer

The network's final predictions or outputs are generated by the output layer. The weighted sum of the inputs to the output layer should be represented by $X^{[L]}$.

Let

$X^{[L]} = W^{[L]}. Z^{[L-1]} + b^{[L]}$

The task determines which activation function is used in the output layer: For regression tasks, one may use a linear activation function. Binary classification tasks represent one potential use case for sigmoid activation functions. Multi-class classification tasks are one scenario in which a SoftMax activation function might be used.

Training

The network's parameters, or weights and biases, are iteratively updated during training in order to minimize a predetermined loss function. The difference between the expected outputs and the actual targets is measured by the loss function

J = Loss (Y, Y')

For classification tasks, cross-entropy loss is a common loss function, and mean squared error (MSE) is used for regression tasks. Gradient descent and other optimization algorithms are used to update the parameters:

$W^{[l]} = W^{[l]} - \propto . \frac{\delta J}{\delta w^{[l]}}$

$b^{[l]} = b^{[l]} - \propto . \frac{\delta J}{\delta b^{[l]}}$

α is the learning rate and $\frac{\delta J}{\delta w^{[l]}}$ and $\frac{\delta J}{\delta b^{[l]}}$ are the gradients and loss function respectively with respect to parameters.

Activation Function: The sigmoid, tanh, ReLU (Rectified Linear Unit), and softmax are examples of common activation functions. They give the network non-linearities, which enable it to discover intricate patterns and connections in the data. Natural language processing and image recognition are just two of

the many fields that use DNNs because they are strong models that can learn intricate representations from data.

DNN is a straightforward algorithm for a single-layered as depicted in Fig. 2. This DNN predicts output Y using input features X. The algorithm uses both backpropagation—which involves adjusting weights based on error—and forward propagation, which involves computing output.

Algorithm 1: Training a Deep Neural Network for Anomaly Detection

1. Initialization:

Initialize the weights W^1 and W^2 randomly

Initialize biases b^1 and b^2 to 0

Choose activation function for hidden layers gn^1 and op2 for output layer

2. Forward Propagation

Compute the activation Function

$Z^1 = W^1X + b^1$

$A^1 = g^1(Z^1)$

Compute the activation function for output layer

$Z2 = W^2\ A^1 + b^2$

$Y` = A^2 = g^2\ (Z^2)$

3. Compute Loss:

Determine the difference between the expected Y and actual outputs, Y`, by

applying a suitable loss function.

4. Back Propagation

Determine the gradient of the loss in relation to the activation of the output layer:

$$dZ^2 = \frac{\partial loss}{\partial Z^2}$$

Determine the gradient of the loss in relation to the output layer's weights and biases.

$$dW^2 = \frac{\partial Loss}{\partial W^2} = \frac{\partial Loss}{\partial b^2} = \frac{\partial Loss}{\partial b^2}$$

Determine the gradient of the loss in relation to the activation of the hidden layer:

$$dA^1 = \frac{\partial Loss}{\partial A^1} = \frac{\partial Loss}{\partial Z^1} = \frac{\partial Loss}{\partial Z^1}$$

Determine the gradient of the loss in relation to the hidden layer's weights and biases.

$$dW^1 = \frac{\partial Loss}{\partial W^1} = \frac{\partial Loss}{\partial b^1} = \frac{\partial Loss}{\partial b^1}$$

5. Updating the parameters

Change both layers' weights and biases with an optimization algorithm (such as gradient descent):

Algorithm 1: Training a Deep Neural Network for Anomaly Detection

$W^1 = W^1 - \alpha . dW^1$

$b^1 = b^1 - \alpha . db^1$

$W^2 = W^2 - \alpha . dW^2$

$b^2 = b^2 - \alpha . db^2$

6. Repeat

Repeat steps 2 to 5 for the specified number of epochs or until convergence.

End

A DNN is an effective algorithm that can identify and simulate complex patterns in sizable and multifaceted datasets. Because of its multilayered architecture and non-linear transformations, it can perform a wide range of machine learning tasks with high levels of accuracy and generalization.

3.4.2 RNN and LSTM

A type of neural network called a recurrent neural network (RNN) uses the output from the preceding step as the input for the current step as presented in Fig. 3. All of the inputs and outputs in conventional neural networks are independent of one another. However, in situations where it is necessary to guess the following word in a sentence, the preceding words are necessary, so it is necessary to retain the preceding words. Thus, RNN was created, and it used a Hidden Layer to solve this problem. The Hidden state of an RNN, which retains certain information about a sequence, is its primary and most significant feature. Because the state retains memory of the previous input to the network, it is also known as Memory State. In order to produce the output, it executes the same task on all inputs or hidden layers using the same parameters for each input. In contrast to other neural networks, this lowers the complexity of the parameters.RNNs are able to retain a memory of previous inputs because of their directed cycle connections, which set them apart from feedforward neural networks, which process data in a single pass. For tasks like language modelling, sequence generation, and time series prediction, they are therefore highly suited.The computation in an RNN at every time step can be expressed mathematically as follows:

Given the input sequence $x = (x_1, x_2, x_3, x_4, \ldots\ldots x_t)$ The hidden state h_t at time step t is calculated as follows, where x_t denotes the input at time step t and h_{t-1} denotes the hidden state from the previous time step.

$h_t = f (W_h h_{t-1} + W_x x_t + b)$

where, the weight matrix Wh links the most recent time step's hidden state compared to the previous time step's hidden state. Wx is the weight matrix that connects the input at each time step to the hidden

state that is currently in effect. A bias vector, b, is present. Typically, the activation function, represented by f, is a non-linear function, such as the hyperbolic tangent (tanh) or the rectified linear unit (ReLU). The current hidden state can then be used to calculate the output y_t at each time step:

$y_t = g (W_y h_t + C)$

where the weight matrix that links the hidden state to the output is denoted by W_y.

There is a bias vector, C. The output activation function, g, varies depending on the type of task (linear activation for regression, softmax for classification, etc.).

Using methods like backpropagation through time (BPTT), where the gradients are computed with respect to the entire sequence, the parameters W_h, W_x, W_y, b, and c are learned during training. Although RNNs are excellent at capturing temporal dependencies, they have a problem with learning long-term dependencies due to the vanishing gradient problem, in which gradients get very small over lengthy sequences. Because of this restriction, more advanced RNN variations including Long Short-Term Memory (LSTM) and GRUs, or recurring units, have mechanisms built in to more effectively capture long-range dependencies.

3.4.3 LSTM:

A popular recurrent neural network (RNN) architecture in deep learning is called LSTM (Long Short-Term Memory). It is excellent at identifying long-term dependencies, which makes sequence prediction tasks a perfect fit for it. Because LSTM has feedback connections, as opposed to traditional neural networks, it can process entire data sequences as opposed to just single data points. Because of this, it is very good at identifying and forecasting patterns in sequential data, such as time series, text, and speech.

The first section determines whether the data from the preceding timestamp should be stored in memory or if it is unimportant and can be ignored. The cell attempts to learn new information from the input to this cell in the second section. Finally, the cell transfers the updated data from the current timestamp to the subsequent timestamp in the third section. A single-time step is this single LSTM cycle.The terms "gates" refer to these three components of an LSTM unit. They regulate the information that enters and exits the memory cell, also known as the LSM cell. The output gate is the final gate; the forget gate is the first; the input gate is the second; and so on. An LSTM unit composed of these three gates and a memory cell, also called an LSTM cell, can be thought of as a layer of neurons in a conventional feedforward neural network, where each neuron has a hidden layer and a current state.

An LSTM has a hidden state, just like a basic RNN, where h(t-1) is the hidden state of the timestamp that was previously recorded, and ht is the hidden state of the timestamp that is currently recorded. Furthermore, the cell state of an LSTM is denoted by C(ct-1) for the past timestamp and C(ct) for the present timestamp, respectively. In this case, long-term memory refers to the cell state and short-term memory to the hidden state as depicted in Fig. 4

To regulate the information flow inside the memory cell, LSTMs use gating mechanisms. An LSTM has three primary gates: The forget gate (Fg_t) decides which data from the cell state should be deleted. The input gate (Ig_t) decides what fresh data should be added to the cell state. The output gate (Og_t) selects which data from the cell state to output. The memory of the LSTM network is called the cell state. It can carry information across many time steps without much degradation because it runs linearly through the entire sequence with only minor linear interactions. Information enters and exits the cell state under the control of the gates. The computations within an LSTM cell can be described as follows:

$$f_{gt} = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right)$$

$$i_{gt} = \sigma\left(W_i . [h_{t-1}, x_t] + b_i\right)$$

$$C_{ct}^{\sim} = tan_h\left(W_C . [h_{t-1}, x_t] + b_c\right)$$

$$C_{ct} = f_t . C_{ct-1} + i_t . C_{ct}^{\sim}$$

$$O_t = \sigma\left(W_o . [h_{t-1}, x_t] + b_o\right)$$

$$h_t = O_t . tan_h(C_{ct)}$$

3.4.4 Hybrid Deep Learning

To efficiently identify anomalies in complicated data, hybrid deep learning approaches in anomaly detection combine the advantages of deep learning architectures with RNN and LSTM deep learning techniques. These hybrid strategies seek to combine the interpretability and flexibility of conventional techniques with the representational power of deep learning models. The following is an approach to hybrid deep learning in anomaly detection:

Preprocessing procedures and feature engineering techniques can be used to extract pertinent features and get the data ready for modelling before feeding it into deep learning models. This could include coding categorical variables, scaling features, handling missing values, and producing derived features. Since they can handle sequential data, RNNs and LSTMs are useful for modelling time-series and sequential data with potential anomalies that could appear over time. These networks can detect anomalies that appear as departures from expected patterns because they can capture temporal dependencies in the data.

Feature Learning: The need for human feature engineering is reduced because RNNs and LSTMs can automatically extract informative features from sequential data. This is especially useful for tasks involving anomaly detection, where the underlying patterns might be intricate and challenging to identify with manually created features. Recurrent Connection: Because of their recurrent connections, RNNs and LSTMs are able to retain information over time and identify long-range dependencies in the data. This feature is useful for identifying anomalies that show subtle temporal patterns or span several time steps.

Model Training: Depending on the availability of labeled anomaly data, RNNs and LSTMs can be trained using a variety of methods, such as supervised, semi-supervised, and unsupervised learning approaches. Anomaly detection frequently employs unsupervised or semi-supervised training, wherein the model gains the ability to distinguish between typical and abnormal sequences without the need for explicit labels. Reconstruction Error: In anomaly detection using RNNs and LSTMs, anomalies are often identified based on the reconstruction error—the difference between the input sequence and its reconstructed counterpart generated by the model. Anomalies typically result in higher reconstruction errors, signaling deviations from normal behaviour.

Threshold

After training, the reconstruction errors of input sequences are compared to a predetermined threshold, allowing the RNN or LSTM model to identify anomalies. Anomalies are identified as sequences with reconstruction errors greater than the threshold. As an alternative, anomaly detection can be conceptualized as a binary classification task, in which the model is trained to identify normal and anomalous sequences based on reconstruction errors.

Model interpretation

Although LSTMs and RNNs have strong anomaly detection capabilities, their intricate internal representations and complicated architectures can make it difficult to understand the decisions these models make. Strategies like layer-wise relevance propagation (LRP) and attention mechanisms can be used to pinpoint the features that support anomaly detection and offer insights into the model's decision-making process.

Algorithm 2: Hybrid Deep Learning Model for Anomaly Detection
1. Input: **• X – Sequential input data** **• T – time series length** **• K - number of features** **• Epsilon – threshold for anomaly detection**
2. Preprocessing **• Normalize the input data X if necessary**
3. **Training Phase** • To extract features from the input sequence, train a DNN model. • Using the features that were extracted from the DNN, train an LSTM model to predict the following data point in the sequence. • Adjust both models' hyperparameters with a validation set. • Keep the learned models for deduction.
4. **Inference Phase** For each time step t from n + 1 to T • Extracted_feature = DNNmodel predict (X[t-n : t-1]) • Predicted_output_lstm = lstmmodel predict (extracted_feature) • Reconstruction_error_LSTM = norm(X[t] – predicted_output_lstm) IfReconstruction_error_LSTM > epsilon: Flag anomaly at time step t
5. **Output** • Anomaly detected in the input

The proposed methodology introduces a Hybrid Deep Learning model for anomaly detection in healthcare, combining Deep Neural Networks (DNN) and Long Short-Term Memory (LSTM) networks. The model processes data from both on-site and remote patients, with smartphones serving as gateways for sensor data within the Body Area Network (BAN). The Software-Defined Networking (SDN) platform manages these gateways by applying security rules from SDN controllers.

Data pre-processing involves collecting information from various sources, handling noise, outliers, and missing values, and performing feature engineering to prepare the data for modeling. The DNN extracts features from the input sequence, and the LSTM predicts the next data point, identifying anomalies

based on reconstruction errors that exceed a predefined threshold. The combination of DNN and LSTM enhances the model's ability to detect complex anomalies in healthcare data.

4 Experimental Setup

Regardless of the particular algorithm used for the automatic detection of anomaly, our goal in the first setting is to differentiate between anomaly-based and non-anomaly activities using hybrid deep learning model. For the experimental analysis, Keras was utilized in conjunction with TensorFlow. To run the experiment on Windows 11, the TensorFlow with GPU is enabled, which accelerates gradient descent computations for deep learning architectures, as well as an Intel Core i7 processor with 3.00 GHz and 16 GB of RAM. The training model's batch size was set to 64 and the embedding vector to 128 through constant parameter optimization and modification during the experiments.

4.1 Measuring the performance of Hybrid Deep learning model for anomaly detection

The entire dataset is split into an 80:20 ratio after the pre-processed stage is finished, with 80% of the data going toward training and the remaining 20% going toward testing.A confusion matrix is a helpful tool in deep learning that can be used to assess a classification model's performance. The Table 1 is frequently used to show how well a classification model performs when applied to a set of test data whose true values are known. This is how it operates:When the model accurately predicts the positive class, these are known as True Positives (TP).When the model accurately predicts the negative class, these are known as True Negatives (TN).False Positives (FP): These are situations in which the model predicts the positive class incorrectly—that is, it interprets the data as positive when it is, in fact, negative which is also known as Type I error.False Negatives (FN) are situations in which the model predicts the negative class incorrectly—that is, it predicts a positive class when it is actually negative which is also known as Type II error.

Table 1
Confusion Matrix of TP, TN, FP and FN

Predicted Data		Actual Data	
		Positive	Negative
	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Typically, a confusion matrix has square dimensions, which correspond to the number of classes that are being predicted. It's a 2×2 matrix for binary classification purposes. An N×N matrix is used for multi-class classification, where N is the number of classes. Numerous performance metrics, including accuracy, precision, recall (sensitivity), specificity, F1-score, and others, can be computed from the

confusion matrix. These metrics offer information about the model's performance and potential error areas. The following typical metrics are obtained from a confusion matrix:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision = TP / (TP + FP).

Sensitivity of Recall = TP / (TP + FN)

Specificity = TN / (TN + FP)

F1-score: 2 * (Recall * Precision) / (Recall + Precision)

These metrics aid in identifying the model's advantages and disadvantages and can direct modifications to the training procedure or the choice of models.

Reconstruction error is the difference between input data and output after an autoencoder model, which compresses the input data into a lower-dimensional latent space. Autoencoders learn to reconstruct input data with minimal loss during training, but when presented with anomalous data, the reconstruction process may result in higher errors, indicating potential anomalies. A threshold is often set for anomaly detection, and data points exceeding this threshold are flagged as anomalies, indicating significant differences from the training data. Anomaly detection uses reconstruction error and frequency to identify unusual patterns or outliers in data sets. Reconstruction error refers to the difference between input data and output after passing through an auto encoder model as depicted in Fig. 5. A higher error indicates significant deviation from the learned patterns. A threshold is set for reconstruction error, and data with errors exceeding this threshold are flagged as anomalies. Frequency refers to the occurrence rate of a particular pattern or event within a dataset or time series. Anomalies are rare or infrequent occurrences that deviate significantly from the norm.

Outliers or anomalies in the data that actually deviate from the norm are called true anomalies. The hybrid deep learning model aims to accurately identify these genuine anomalies through the use of its sophisticated architecture and learning capabilities. In the context of hybrid deep learning models, "detected anomalies" refers to situations in which the model has found patterns or data points within a dataset that substantially differ from the average. Usually, the model's output which could take the shape of probabilities, labels, or anomaly scores for every data point is used to identify these anomalies. The categorized true and detected anomalies in dataset are represented in Fig. 6.

The interpretation of confusion matrix of DNN, LSTM and Hybrid deep learning model is represented in Fig. 7, Fig. 8 and Fig. 9. A common tool for assessing how well classification models work is a confusion matrix, which summarizes the counts of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) predictions. The predicted class labels are represented by each column in the confusion matrix, while the actual class labels are represented by each row. This section will examine the

interpretation of confusion matrices for three distinct model types: hybrid models that combine both DNNs and LSTMs, Long Short-Term Memory (LSTM) networks, and Deep Neural Networks (DNNs).

Table 2
Performance comparison of proposed algorithm

Models	Accuracy	Precision	Recall	F1*score
CNN	94	93	98	95
RNN	96	98	97	97
DNN	95	97	96	97
LSTM	97	99	97	98
Hybrid Model	**100**	**97**	**98**	**98**

In order to evaluate the models' performance, this article has also used a 10-fold cross-validation strategy over the training set. There are ten folds produced using the training set. Ten iterations of this process are performed, once for each fold's validation. Ultimately, averaging all the metrics over validation folds yields a more accurate evaluation of the performance. The proposed model was compared with four models like CNN, RNN, DNN and LSTM as presented in Table 2. The models of CNN, RNN, DNN and LSTM has reached the accuracy of 94%, 96%, 95% and 97% respectively whereas the hybrid deep learning model has reached 99% of accuracy.

Table 3
Accuracy and Loss of DNN model

Training_Loss	0.19	Training_Accuracy	0.9
Validation_Loss	0.01	Validation_Accuracy	1

From Table 3, it is observed that accuracy and loss of DNN model. The evaluation measure of training loss and validation loss have achieved 0.19% and 0.01% respectively whereas the training accuracy have achieved about 0.9% and validation accuracy reached about 1%. A higher accuracy means that the DNN is predicting more things correctly, whereas a lower accuracy means that the model is having trouble correctly classifying instances as presented in Fig. 10. It offers a clear way to gauge total performance. Model convergence and alignment with the training data are better when the loss value is lower. In order to make sure the DNN is efficiently learning from the data and adjusting its parameters to increase predictive accuracy, it is helpful to monitor loss during training.

From Table 4, it is observed, accuracy and loss metrics offer important insights into how well LSTM models perform. Loss measures the difference between the predicted and actual sequences, whereas accuracy evaluates how accurate sequence predictions are. These metrics are crucial for tracking the convergence of the model during training and assessing how well LSTM models capture temporal dependencies in sequential data. The training loss and validation loss has reached about 0.07 and 0.00

respectively and the training accuracy and the validation accuracy has reached about 1% as represented in Table 4 and Fig. 11.

Table 4
Accuracy and Loss of LSTM model

Training_Loss	**0.071**	**Training_Accuracy**	**1**
Validation_Loss	0.00	Validation_Accuracy	1

To take advantage of each architecture's advantages, hybrid models often combine several architectures, such as LSTMs and DNNs. The combined model's performance in terms of sequential analysis (LSTM part) and feature extraction (DNN part) can be inferred from accuracy and loss. It's critical to evaluate how well hybrid models represent the data's complexities and whether the combined architecture outperforms its component parts when evaluating their performance.The performance of LSTM and hybrid models can be assessed using accuracy and loss, which are crucial metrics that provide information about the models' capacity for classification and convergence during training. Table 5 and Fig. 12.

Table 5
Accuracy and Loss of Hybrid model

Training_Loss	**0.13**	**Training_Accuracy**	**1**
Validation_Loss	1.07	Validation_Accuracy	1

Hybrid models often combine several architectures, such as LSTMs and DNNs. The combined model's performance in terms of sequential analysis (LSTM part) and feature extraction (DNN part) can be inferred from accuracy and loss. It's critical to evaluate how well hybrid models represent the data's complexities and whether the combined architecture outperforms its component parts when evaluating their performance. Accuracy and loss are critical metrics that can be used to evaluate the performance of LSTM and hybrid models. The model tell us about the models' ability to classify data and converge during training. Table 4.3 and Fig. 4.9 reveal that the training accuracy and validation accuracy have both reached 1%, while the training loss has reached approximately 0.13 and the validation loss is approximately 0.10.

5. Result and Discussion

This section presents a study that investigates the efficiency of hybrid deep learning for anomaly detection in environments with Software Defined Networking (SDN) to enhance connectivity. Our method efficiently detects anomalous behavior in network traffic by combining the advantages of long short-term memory (LSTM) networks and deep neural networks (DNNs). Table 6 represents the comparison of existing work of various authors in respect to their model and accuracy. The authors evaluated our hybrid deep learning model's performance using SDN datasets and contrasted hybrid model's detection

accuracy (A), precision (P), recall (RC), and F1-score with conventional anomaly detection methodologies like statistical techniques and single-model deep learning approaches. The results demonstrate that our hybrid deep learning model consistently outperforms these conventional methods across a variety of performance metrics. The benefits of our hybrid approach are its scalability and efficiency. The suggested model efficiently manages large-scale SDN deployments by fusing the parallel processing capacity of DNNs with the memory-efficient architecture of LSTMs, all without compromising detection accuracy or adding a sizable amount of computational overhead.

Table 6
Comparison of existing work with proposed model

Authors/Year	Model	Accuracy
Halman (Halman &Alenazi, 2023)	MCAD	99.24
Saheed(Saheed & Arowolo, 2021)	PSO-RF	99.76
Haque (Haque & Aziz, 2013)	LCNN	94.0
Nguyen (Nguyen et al., 2018)	CNN	92
Rajesh (Kumar et al., 2019)	BC and ML	98
Proposed Model	**Hybrid DL**	**100**

In comparison to single-model approaches, by utilizing the advantages of each of the several deep learning architectures or algorithms it combines, a hybrid model can outperform the others. When it comes to anomaly detection in SDN, a hybrid model can capture both temporal and spatial dependencies in network traffic data, which improves the accuracy and dependability of anomaly detection. Figure 13 represents the comparison between the existing models and proposed model in respect to the accuracy. In terms of machine learning and deep learning models, numerous researchers have put forth different models to identify anomalies in diverse domains. The goal of the proposed work was to use hybrid deep learning models to improve understanding and accuracy in anomaly detection tasks.

6. Conclusions

The implementation of hybrid deep learning for anomaly detection in SDN significantly enhances network connectivity while simultaneously fortifying cybersecurity defenses. This dual benefit empowers enterprises to proactively safeguard their critical assets and maintain the integrity and reliability of their network infrastructure in the face of evolving threats and vulnerabilities. The need for advanced anomaly detection solutions is paramount to ensure the continued success and resilience of SDN in an increasingly interconnected and dynamic digital environment. SDN is poised to remain a cornerstone in the evolution of networking technologies.

The proposed hybrid deep learning model, which synergizes the feature extraction capabilities of DNNs with the temporal analysis strengths of LSTMs, achieved a remarkable 100% accuracy in detecting anomalies. This study's results underscore the exceptional efficacy of the hybrid model in identifying anomalous behavior within SDN traffic, thereby demonstrating its potential as a highly effective solution for contemporary network security challenges. The scalability and efficiency of the hybrid model make it an ideal choice for managing large-scale SDN deployments without compromising detection accuracy or incurring significant computational overhead..

Declarations

Competing interests:

The authors declare that they have no competing interests.

Availability of data and materials:

The datasets used and analysed during the current study are available from the corresponding author on reasonable request.

Funding:

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Acknowledgements:

The authors would like to acknowledge the anonymous reviewers for their thoughtful comments.

Author contributions:

Leo Prasanth L designed the algorithm, performed the simulation results and drafted the manuscript under the supervision of Dr. E. Uma. All authors read and approved the final manuscript.

References

1. Alshamrani, M. (2022). IoT and artificial intelligence implementations for remote healthcare monitoring systems: A survey. In *Journal of King Saud University - Computer and Information Sciences* (Vol. 34, Issue 8, pp. 4687–4701). King Saud bin Abdulaziz University. https://doi.org/10.1016/j.jksuci.2021.06.005
2. Chaudhary, R., & Kumar, N. (2019). LOADS: Load Optimization and Anomaly Detection Scheme for Software-Defined Networks. *IEEE Transactions on Vehicular Technology*, *68*(12), 12329–12344. https://doi.org/10.1109/TVT.2019.2948222

3. Chen, A., Fu, Y., Zheng, X., & Lu, G. (2022). An efficient network behavior anomaly detection using a hybrid DBN-LSTM network. *Computers and Security*, *114*. https://doi.org/10.1016/j.cose.2021.102600
4. Erhan, L., Ndubuaku, M., Di Mauro, M., Song, W., Chen, M., Fortino, G., Bagdasar, O., & Liotta, A. (2021). Smart anomaly detection in sensor systems: A multi-perspective review. *Information Fusion*, *67*, 64–79. https://doi.org/10.1016/j.inffus.2020.10.001
5. *Flow Based Anomaly Detection in Software Defined Networking: A Deep Learning Approach With Feature Selection Method*. (n.d.).
6. Halman, L. M., &Alenazi, M. J. F. (2023). MCAD: A Machine Learning Based Cyberattacks Detector in Software-Defined Networking (SDN) for Healthcare Systems. *IEEE Access*, *11*, 37052–37067. https://doi.org/10.1109/ACCESS.2023.3266826
7. Haque, S. A., & Aziz, S. M. (2013). False Alarm Detection in Cyber-physical Systems for Healthcare Applications. *AASRI Procedia*, *5*, 54–61. https://doi.org/10.1016/j.aasri.2013.10.058
8. Kumar, R., & Agrawal, N. (2024). Software defined networks (SDNs) for environmental surveillance : A Survey. *Multimedia Tools and Applications*, *83*(4), 11323–11365. https://doi.org/10.1007/s11042-023-15729-8
9. Kumar, R., Zhang, X., Wang, W., Khan, R. U., Kumar, J., & Sharif, A. (2019). A Multimodal Malware Detection Technique for Android IoT Devices Using Various Features. *IEEE Access*, *7*, 64411–64430. https://doi.org/10.1109/ACCESS.2019.2916886
10. Latah, M., & Toker, L. (2019). Artificial intelligence enabled software-defined networking: A comprehensive overview. In *IET Networks* (Vol. 8, Issue 2, pp. 79–99). Institution of Engineering and Technology. https://doi.org/10.1049/iet-net.2018.5082
11. Malik, J., Akhunzada, A., Bibi, I., Imran, M., Musaddiq, A., & Kim, S. W. (2020). Hybrid Deep Learning: An Efficient Reconnaissance and Surveillance Detection Mechanism in SDN. *IEEE Access*, *8*, 134695–134706. https://doi.org/10.1109/ACCESS.2020.3009849
12. Nguyen, H. T., Ngo, Q. D., & Le, V. H. (2018). IoT Botnet Detection Approach Based on PSI graph and DGCNN classifier. *2018 IEEE International Conference on Information Communication and Signal Processing, ICICSP 2018*, 118–122. https://doi.org/10.1109/ICICSP.2018.8549713
13. Qureshi, K. N., Jeon, G., &Piccialli, F. (2021). Anomaly detection and trust authority in artificial intelligence and cloud computing. *Computer Networks*, *184*. https://doi.org/10.1016/j.comnet.2020.107647
14. Radoglou-Grammatikis, P., Rompolos, K., Sarigiannidis, P., Argyriou, V., Lagkas, T., Sarigiannidis, A., Goudos, S., & Wan, S. (2022). Modeling, Detecting, and Mitigating Threats against Industrial Healthcare Systems: A Combined Software Defined Networking and Reinforcement Learning Approach. *IEEE Transactions on Industrial Informatics*, *18*(3), 2041–2052. https://doi.org/10.1109/TII.2021.3093905
15. Rafique, W., Qi, L., Yaqoob, I., Imran, M., Rasool, R. U., & Dou, W. (2020). Complementing IoT Services through Software Defined Networking and Edge Computing: A Comprehensive Survey. *IEEE*

Communications Surveys and Tutorials, *22*(3), 1761–1804. https://doi.org/10.1109/COMST.2020.2997475

16. Raja, N. M., &Vegad, S. (2023). An empirical study for the traffic flow rate prediction-based anomaly detection in software-defined networking: a challenging overview. *Social Network Analysis and Mining*, *13*(1). https://doi.org/10.1007/s13278-023-01057-0
17. Restuccia, F., D'Oro, S., & Melodia, T. (2018). Securing the Internet of Things in the Age of Machine Learning and Software-Defined Networking. *IEEE Internet of Things Journal*, *5*(6), 4829–4842. https://doi.org/10.1109/JIOT.2018.2846040
18. Saheed, Y. K., & Arowolo, M. O. (2021). Efficient Cyber Attack Detection on the Internet of Medical Things-Smart Environment Based on Deep Recurrent Neural Network and Machine Learning Algorithms. *IEEE Access*, *9*, 161546–161554. https://doi.org/10.1109/ACCESS.2021.3128837
19. Said Elsayed, M., Le-Khac, N. A., Dev, S., &Jurcut, A. D. (2020). Network Anomaly Detection Using LSTM Based Autoencoder. *Q2SWinet 2020 - Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 37–45. https://doi.org/10.1145/3416013.3426457
20. Shafi, Q., Basit, A., Qaisar, S., Koay, A., & Welch, I. (2018). Fog-Assisted SDN Controlled Framework for Enduring Anomaly Detection in an IoT Network. *IEEE Access*, *6*, 73713–73723. https://doi.org/10.1109/ACCESS.2018.2884293
21. Shah, S. D. A., Gregory, M. A., & Li, S. (2021). Cloud-Native Network Slicing Using Software Defined Networking Based Multi-Access Edge Computing: A Survey. In *IEEE Access* (Vol. 9, pp. 10903–10924). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2021.3050155
22. Sharma, B., Pokharel, P., & Joshi, B. (2020, July 1). User Behavior Analytics for Anomaly Detection Using LSTM Autoencoder-Insider Threat Detection. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3406601.3406610
23. Siddiqui, S., Hameed, S., Shah, S. A., Ahmad, I., Aneiba, A., Draheim, D., &Dustdar, S. (2022). Toward Software-Defined Networking-Based IoT Frameworks: A Systematic Literature Review, Taxonomy, Open Challenges and Prospects. In *IEEE Access* (Vol. 10, pp. 70850–70901). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2022.3188311
24. Uhongora, U., Mulinde, R., Law, Y. W., & Slay, J. (n.d.). *Deep-learning-based Intrusion Detection for Software-defined Networking Space Systems*.
25. Valdovinos, I. A., Pérez-Díaz, J. A., Choo, K. K. R., & Botero, J. F. (2021). Emerging DDoS attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. In *Journal of Network and Computer Applications* (Vol. 187). Academic Press. https://doi.org/10.1016/j.jnca.2021.103093
26. Wang, B., Sun, Y., & Xu, X. (2021). A Scalable and Energy-Efficient Anomaly Detection Scheme in Wireless SDN-Based mMTC Networks for IoT. *IEEE Internet of Things Journal*, *8*(3), 1388–1405. https://doi.org/10.1109/JIOT.2020.3011521

27. Wang, B., Sun, Y., Yuan, C., & Xu, X. (2018, June 26). LESLA: A smart solution for SDN-enabled mMTC E-health monitoring system. *Proceedings of the 8th ACM MobiHoc 2018 Workshop on Pervasive Wireless Healthcare Workshop, MobileHealth 2018*. https://doi.org/10.1145/3220127.3220128
28. Wani, A., & Revathi, S. (2020). DDoS Detection and Alleviation in IoT using SDN (SDIoT-DDoS-DA). *Journal of The Institution of Engineers (India): Series B*, *101*(2), 117–128. https://doi.org/10.1007/s40031-020-00442-z

Figures

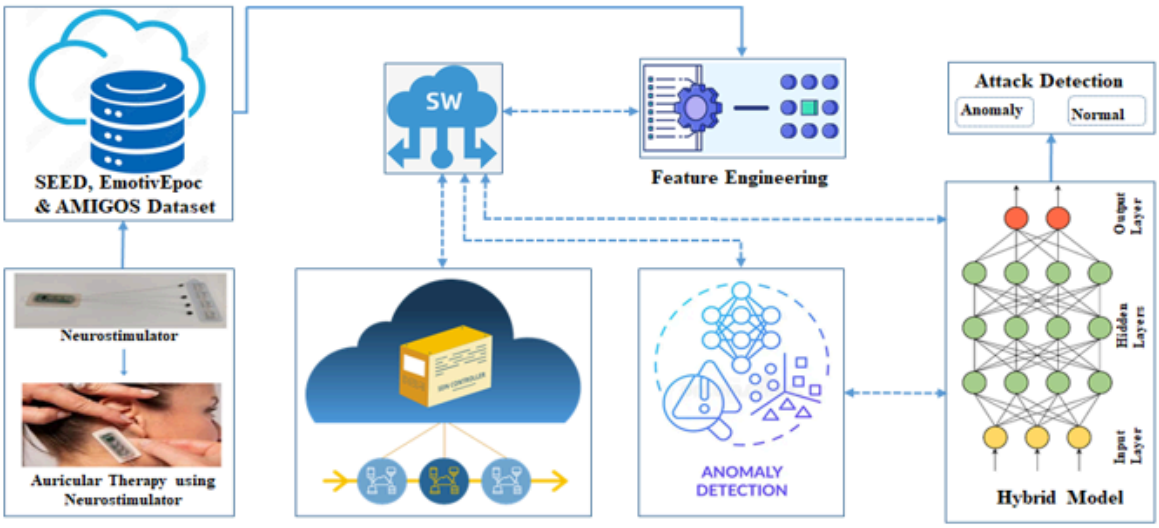

Figure 1

Health Connect SDN Anomaly Detection Framework

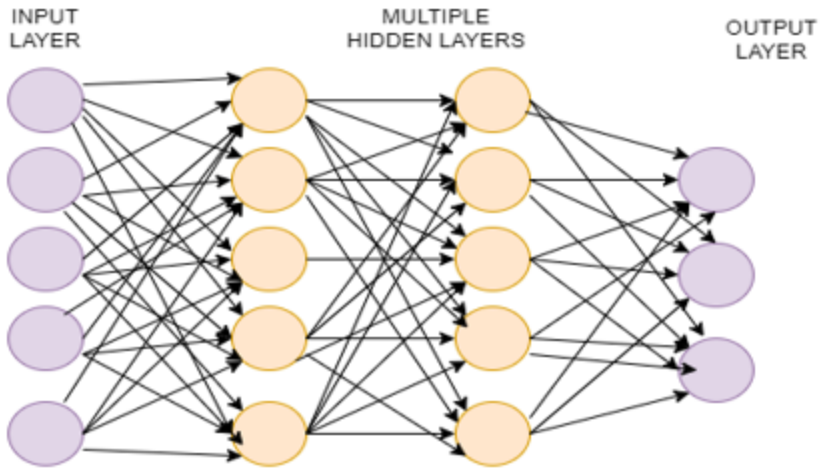

Figure 2

Structure of Deep Neural Network

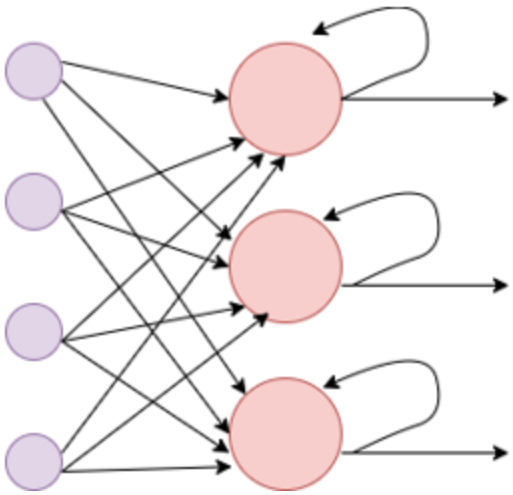

Figure 3

Structure of Recurrent Neural Network

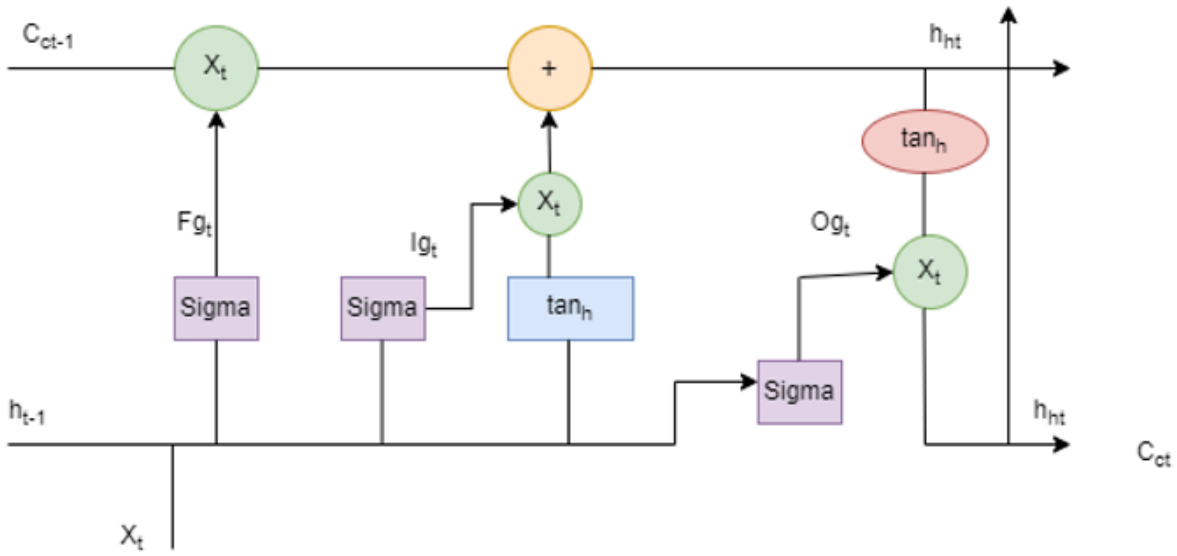

Figure 4

structure of Long Short Term Memory

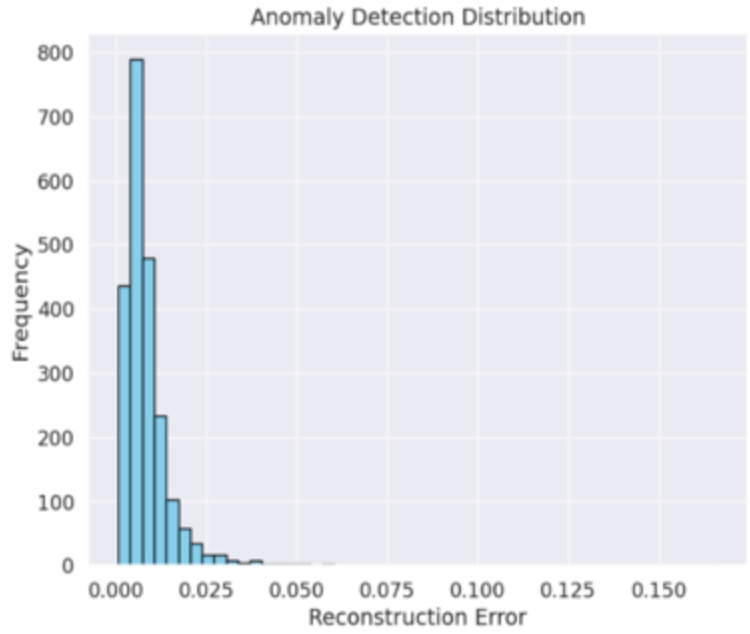

Figure 5

Distribution of Anomaly Detection

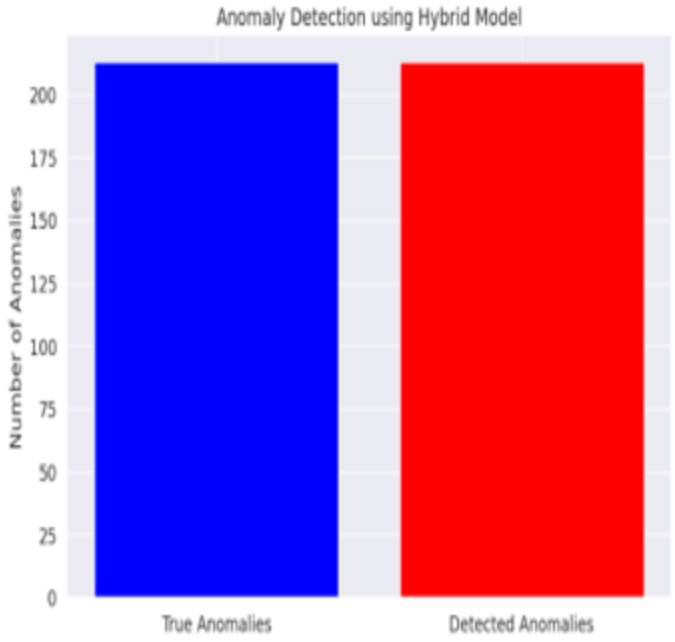

Figure 6

Anomaly Detection using Hybrid Deep Learning Model

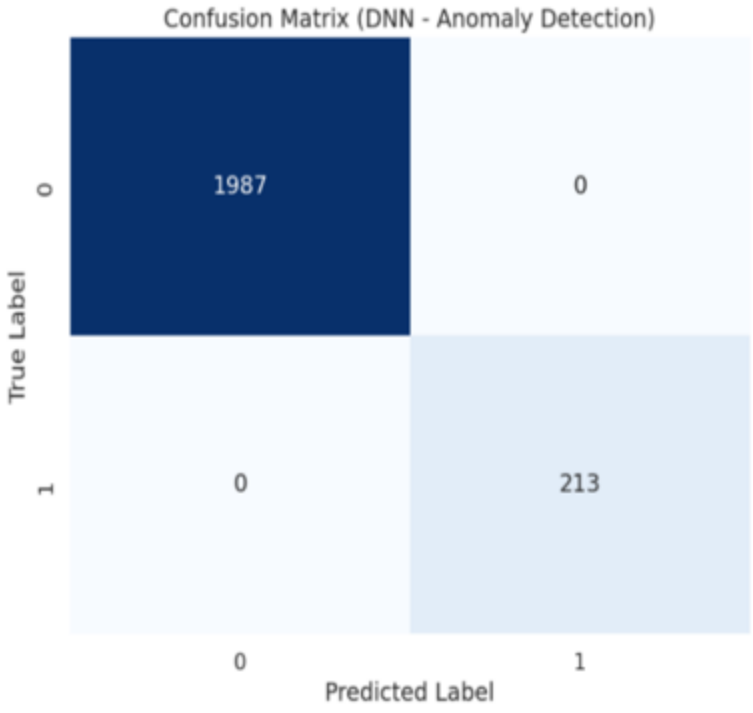

Figure 7

Confusion Matrix for DNN

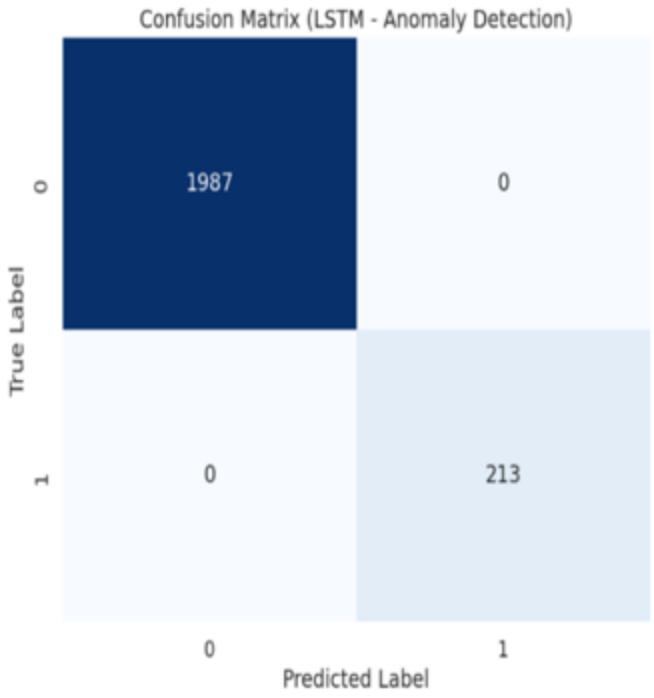

Figure 8

Confusion Matrix for LSTM

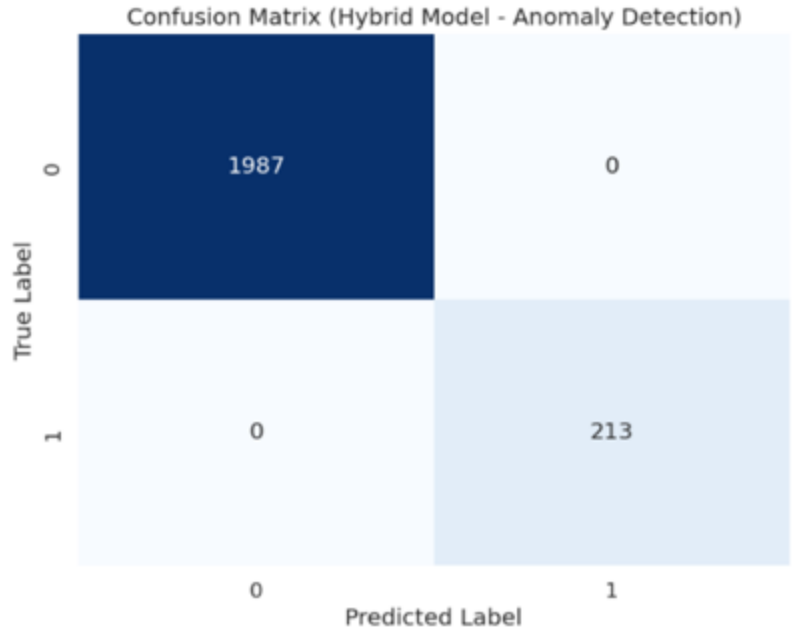

Figure 9

Confusion Matrix for Hybrid Deep Learning Model

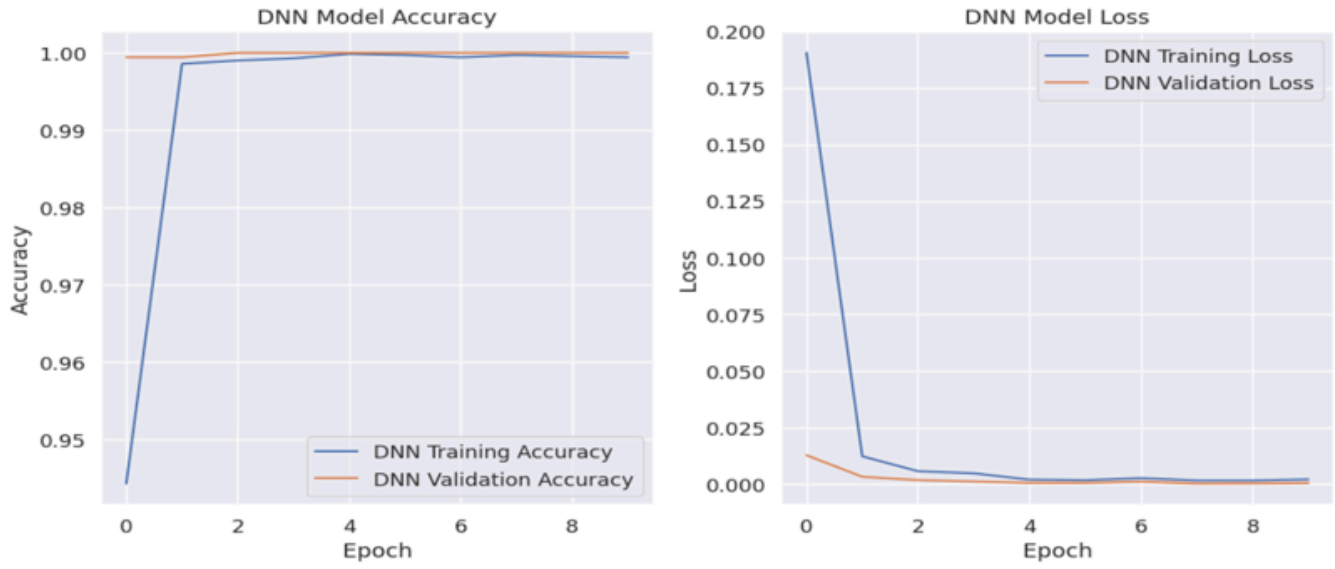

Figure 10

Accuracy and Loss of DNN model

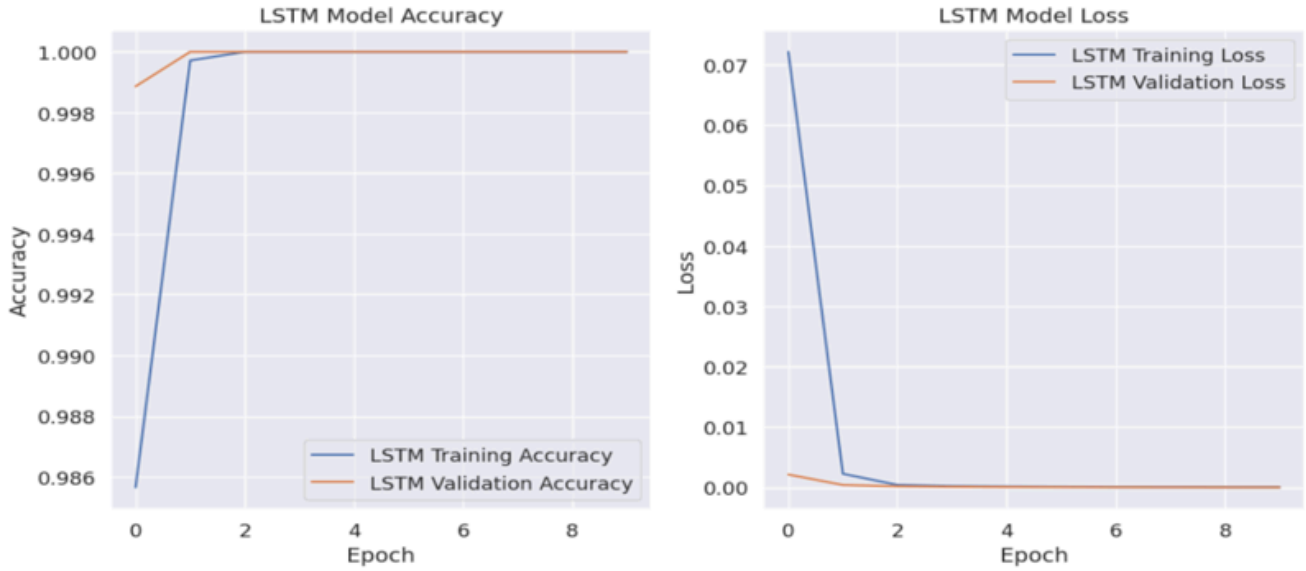

Figure 11

Accuracy and Loss of LSTM model

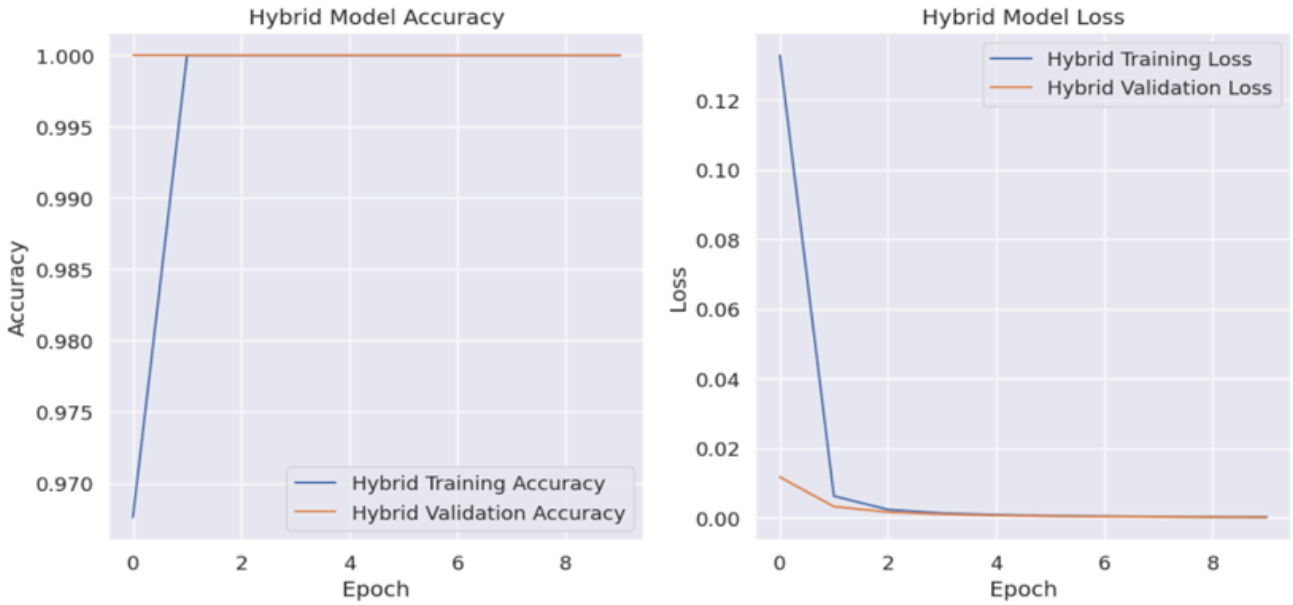

Figure 12

Accuracy and Loss of Hybrid model

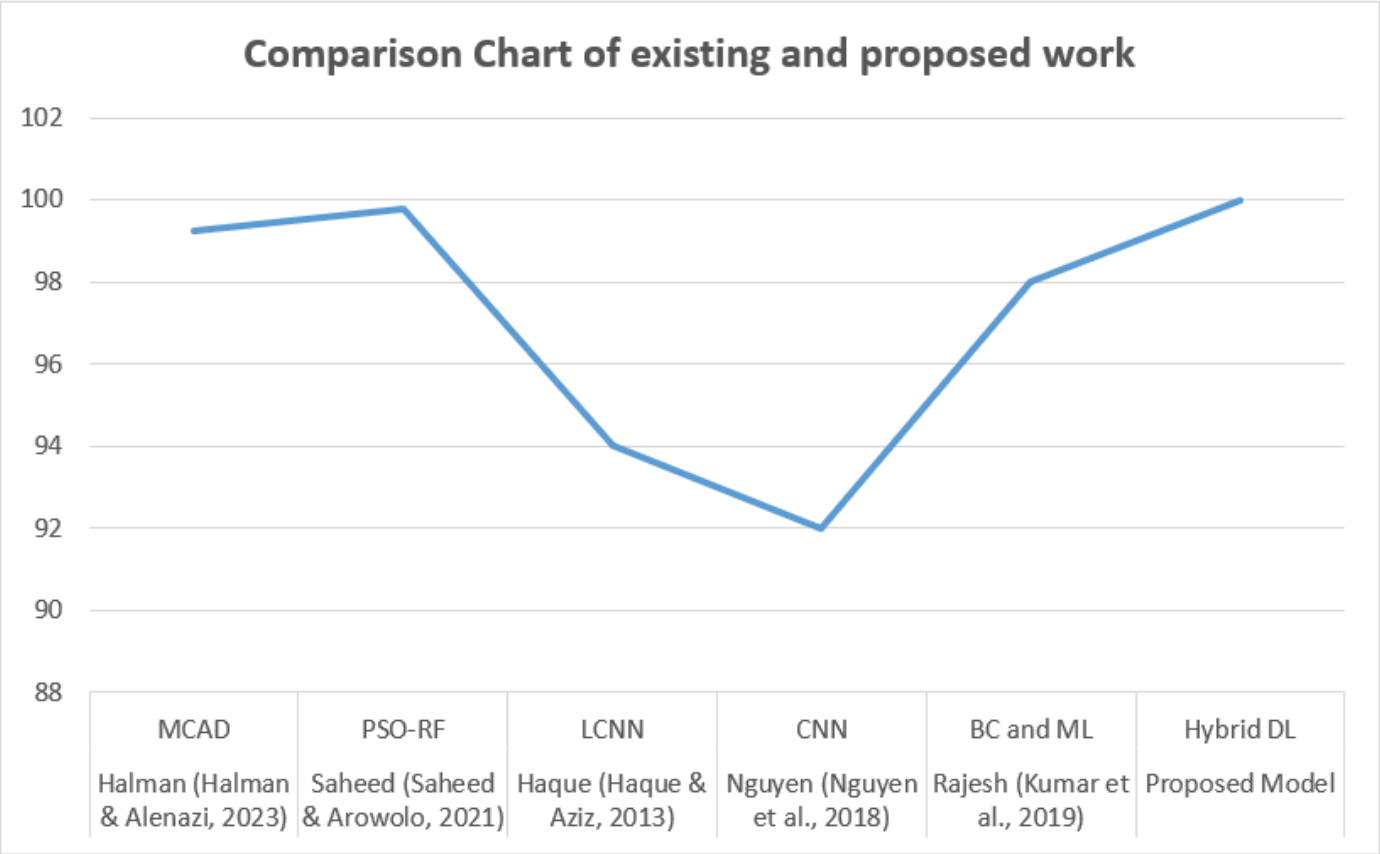

Figure 13

Comparison chart with proposed model