Journal of Theoretical and Applied Information Technology
15th March 2026. Vol.104. No.5
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THREATSCAN - CYBERTHREAT PREDICTION SYSTEM USING TEMPORAL CONVOLUTIONAL NETWORKS

THONDEPU ADILAKSHMI[1], MALLANNAGARI SUNITHA[2] , MADDARAPU PAVAN DURGA NIVAS[3], HARSHITHA PALLAPOLU[4]

[1]Head Of Department and Professor, Department of Computer Science Engineering, Vasavi College of Engineering, Telangana, India.
[2]Asscociate Professor, Department of Computer Science Engineering, Vasavi College of Engineering, Telangana, India.
[3]Student, Department of Computer Science Engineering, Vasavi College of Engineering, Telangana, India.
[4]Student, Department of Computer Science Engineering, Vasavi College of Engineering, Telangana, India.
E-mail: [1]t_adilakshmi@staff.vce.ac.in, [2]m.sunitha@staff.vce.ac.in, [3]pavanmaddarapu@gmail.com, [4]pallapoluharshitha@gmail.com

ABSTRACT

The growing digitization of the public and private sectors increases attack surfaces for cyber threats including cyber denial-of-service (DoS) attacks and more advanced zero-day exploits. Unlike established Intrusion Detection Systems (IDSs) that utilize rule-based signatures and conventional machine learning, the proposed AI-based systems which utilize Temporal Convolutional Networks (TCNs) anticipates threats by scrutinizing multivariate time-series network traffic and system log data of an organization. The proposed model leverages the Cybersecurity Threat Detection and Awareness Program dataset which includes real-world network event data associated with flow logs, IDS alerts, firewall telemetry, and threat intelligence feeds from India. The model evaluates threats and anticipates cyber-attack vectors within the scope of multi-class classification and anomaly detection. The choice of TCNs is due to their ability to model long-range temporal features that is essential in recognizing and predicting attacks that evolve over time. The model predicts and classifies attack types, and signals a detection of an anomaly using five levels (No Threat, Low, Medium, High, and Critical). TCN provides security analytics and threat intelligence convergence which is instrumental to achieving a higher efficacy of incident response, real-time anomaly detection and risk scoring help security teams prioritize responses and mitigate risks before threats escalate.

Keywords: *Cyberthreat, Intrusion Detection System, Dual Stacked Temporal Convolutional Network, Temporal Convolution Network, Machine Learning.*

1. INTRODUCTION

With industries and critical infrastructures built around digital systems, cyber threats become more advanced and more pervasive. Adversaries are using automation by means of AI to design complex and multiscale attacks, which include advanced persistent threats, AI-enhanced phishing, and sophisticated ransomware attacks that through extortion, escalate beyond mere data encryption. At the same time, DDoS attacks are more sophisticated and more extensive, using application-layer floods that get around standard DDoS front-line defenses [1].

The increase in velocity and complexity of cyber attacks and the DDoS attacks means that signature or rule-based systems and security systems focused on reactive defenses are become more obsolete and ineffective [2]. Security systems focused on maintaining the 'known' threat signature are a sitting duck to zero-day attacks and polymorphic malware. It is estimated that more than 80% of successful cyber attacks bypass traditional defenses which is a major concern given the average eCrime breach time is now 62 minutes [3][4]. This offensive gap underscores the need more intelligent security.

Deep Learning (DL) and Machine Learning (ML) technologies have proven useful in threat detection by analyzing large datasets and flagging unusual patterns for real-time escalation [5]. However, technologies such as Long Short Term Memory (LSTM) networks and Recurrent Neural Networks (RNNs) have limited use in real-time applications most notably because of high computation costs and the difficulty of extracting long-term dependencies in sequential data [6].

Maximally exploiting Causal and dilated convolution as well as the convolutional architecture designed for Sequential data, Temporal Convolutional Networks (TCNs) have proven their worth as well. For parallel processing, which is a game changer for recurrent models, TCNs are able to model long-term dependencies. TCNs have been able to address model performance and imbalanced datasets that are common in rare attack detection and have proven their worth in various cyber security domains, including network intrusion detection, in which TCNs have benchmarked other models as most accurate [8].

This work attempts to use TCNs for predictive analytics in advanced cyber threat detection. The proposed model, Dual Stacked TCN(DS-TCN) will evaluate network traffic and system event log records to predict critical values, including threat level and anomaly score, and anomaly detection.

2. LITERATURE SURVEY

In a 2025 study published in Computer Fraud & Security, researchers proposed an updated TCN-SE intrusion detection model specifically optimized for resource-constrained IoT environments.[9] By integrating a Temporal Convolutional Network (TCN) with a Squeeze-and-Excitation (SE) attention module, the model successfully recalibrates feature responses to focus on critical traffic channels. This architecture achieved 98.4% accuracy on the CIC-IDS2018 benchmark, providing a computationally efficient solution that maintains high precision while minimizing resource usage on economic IoT devices.

TCN-Based Anomaly Detection in SDN Addressing security in Software-Defined Networking (SDN), a 2025 study introduced a multi-class S-TCN framework designed to capture long-range dependencies in dynamic network flows. By leveraging causal and dilated convolutions, this approach effectively models packet length sequences and multi-level timing features.[10] Tested on the InSDN dataset, the model improved detection accuracy by approximately 5% over previous state-of-the-art methods, demonstrating superior robustness against zero-day attacks and complex traffic patterns compared to traditional CNN and RNN architectures.

For the near real-time detection of insider threats, Ye Xiaoyun and colleagues (2025) developed a hybrid TCN-Transformer architecture[11]. This framework utilizes a sliding window approach where the TCN component extracts short-term local features and preserves temporal order, while the Transformer module captures global, long-range dependencies in daily user logs. The model achieved a recall rate of 95% with a 30-day sequence length, effectively identifying malicious insider behaviors that evolve slowly over time—patterns often missed by standard anomaly detection systems.

TCN with Self-Attention for Critical Infrastructure. Pushing the boundaries of network intrusion detection, a newly proposed MRG-ID-SA-TCN framework (2025) integrates multi-receptive fields, gating mechanisms, and self-attention blocks. Designed for high-stakes environments like Electric Vehicle Charging Stations (EVCS), this model uses an attention mechanism to explicitly model interactions between different data channels. It demonstrated outstanding performance on the CICEVSE2024 dataset, outperforming traditional TCN and LSTM models by effectively recognizing complex, spatio-temporal attack patterns across varying time scales [12].

Automated Detection of FPE Ransomware. In response to advanced ransomware employing Format-Preserving Encryption (FPE)—a technique that bypasses entropy-based detection by retaining file structure—researchers introduced a new machine learning-based detection framework in 2025. This system analyzes specific file entropy characteristics and neutralizing techniques used by attackers. Evaluation results showed the model successfully identified FPE-encrypted ransomware variants with 94.6% precision, marking a significant milestone in combating modern, evasive Ransomware-as-a-Service (RaaS) attacks [13].

3. PROPOSED IDEA

We propose a Cyber Threat Prediction Framework that identifies cyberattacks in real-time using Machine Learning (ML) and Deep Learning (DL) techniques. At the heart of the system is the Threat Detection and Response Temporal Convolutional Network (TCN). Evaluating the capabilities of TCN to track and analyze sequential patterns within the network to pinpoint and rectify flooding attacks or malware intrusions renders it the most suitable framework for the proposed system [14].

The DS-TCN(Dual Stacked TCN) delivers three different predictive outputs:

- A multi-dimensional severity score which determines the alert prioritization,

- An ever-evolving anomaly score which detects zero-day or unseen threats, and
- An attack type categorization on the classification of an attack.

This predictive architecture empowers the system to go beyond a rudimentary binary classification model. These outputs facilitate sophisticated decision-making for proactive threat response. The system backend was developed in Python/Django, using TensorFlow, Keras, PyTorch, NumPy, Pandas, and Scikit-learn, which assist in preprocessing, feature engineering, and model training. Raw network packets, which are unstructured, are converted into structured time-series d.

An desktop application is developed with help of Electron.js, that displays live network traffic with real-time alerts. Captured data and predictions are stored in MongoDB, for reliable and scalable data access. The cloud supports both forensic analysis and automated retraining, forming a continuous feedback loop for learning and improvement.

3.1 Framework Overview

The proposed system[Fig 1] integrates a deep learning engine, a visualization dashboard, cloud data architecture, and other components as unified architecture. The DS-TCN analyzes network sequences and makes pre-active security predictions for emerging security incidents.

Data is captured and pre-processed for the workflow, which entails conversion of raw network flows to sequences of time series data for the model [15][16]. The DS-TCN, built using TensorFlow/Keras, makes three predictions - severity of the threat, level of the anomaly, and the category of the attack. These predictions are visualized in real time on a dashboard built using Electron.js, while the network logs and model predictions are saved to MongoDB Cloud for archival and retraining purposes.

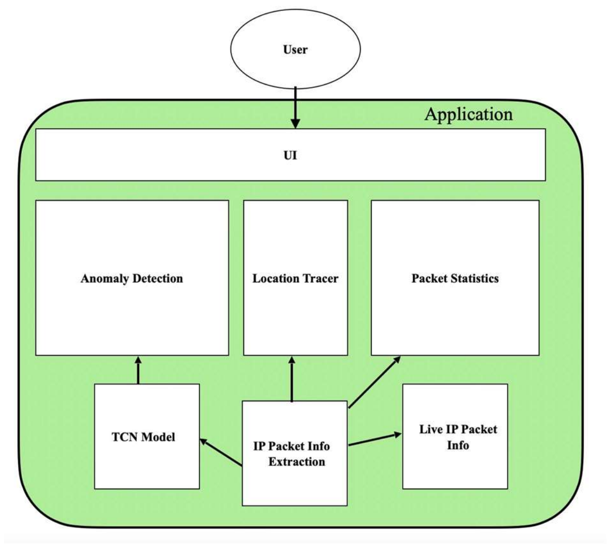


Figure 1: Visual Representation of the Proposed System's Workflow

3.2 Technology Stack and Components

3.2.1 Temporal Convolutional Network (TCN)

As the primary predictive component, TCN benefits from the long-range temporal dependencies made possible by cause and effect dilated convolutions as a sequence-modeling architecture. Training is faster and provides more stable gradients because they can run in parallel. You do not have the sequential computation limits of RNNs. That is why it is effective in anomaly detection of sophisticated threats like DDoS floods and stealth malware.

3.2.2 TensorFlow and Keras

Keras and Tensorflow are used for the neural net, with Keras as the high-level API. It helps in the design and experimentation of various models. Tensorflow and Keras work well together and provide extensive and versatile options for model fine-tuning, custom model deployments at the layer level, and multi dataset management for model deployment.

3.2.3 Scikit-learn, NumPy, and Pandas

The following libraries are used for data preprocessing.
· Pandas transforms raw log data into a dataframe for easier manipulation.
· NumPy carries out vectorized mathematical operations on multi-dimensional arrays.

· Scikit-learn handles feature scaling, and encoding, and evaluation using accuracy and F1 score. This set of tools.

3.2.4 Electron.js

Using Electron.js, we design cross-platform desktop applications with web technologies (HTML, CSS, and JavaScript). For this project, we created a dynamic interface that visualizes traffic statistics, shows model alert messages, and enables analysts to observe live network behaviors. The multi-OS capability ensures that the dashboard is operational on Windows, macOS, and Linux.

3.2.5 MongoDB Cloud

Using scalable, fault-tolerant, and secure MongoDB Cloud as a central data repository, we store large volumes of network data in packet logs and prediction results. It is also a data source for the periodic retraining of the DS-TCN model, which allows it to learn and improve on unstructured network data continuously. MongoDB is a NoSQL, document-oriented database also suited for semi-structured data.

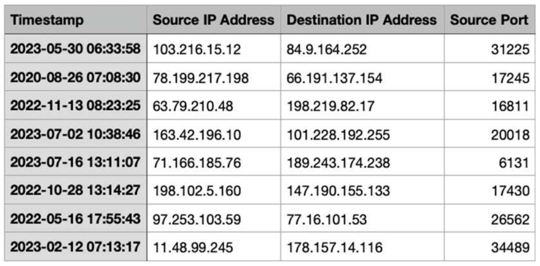
3.2.6 Python Development Environment

All components are in Python to take advantage of its expansive machine learning and data analysis libraries. For development and testing, We VISUAL Studio Code and PyCharm IDEs, which seamlessly integrate with DS-TCN and visualization modules, and offer debugging tools and environment management.

3.3 Data Description

Real and simulated network traffic logs collected over this period amounted to about 40,000 records and any given entry includes fundamental network features like source and destination IPs, source and destination port numbers, protocol types, packet sizes, and flow durations [Fig 2]. Data spans across multiple classes, including DDoS, port scanning, botnets, and normal traffic.

For balanced training, The data was organized in a 70:15:15 structure for training, validation, and testing. Consolidation involved normalization, categorical encoding, and time-windowing to sequence logs for DS-TCN input.



Timestamp	Source IP Address	Destination IP Address	Source Port
2023-05-30 06:33:58	103.216.15.12	84.9.164.252	31225
2020-08-26 07:08:30	78.199.217.198	66.191.137.154	17245
2022-11-13 08:23:25	63.79.210.48	198.219.82.17	16811
2023-07-02 10:38:46	163.42.196.10	101.228.192.255	20018
2023-07-16 13:11:07	71.166.185.76	189.243.174.238	6131
2022-10-28 13:14:27	198.102.5.160	147.190.155.133	17430
2022-05-16 17:55:43	97.253.103.59	77.16.101.53	26562
2023-02-12 07:13:17	11.48.99.245	178.157.14.116	34489

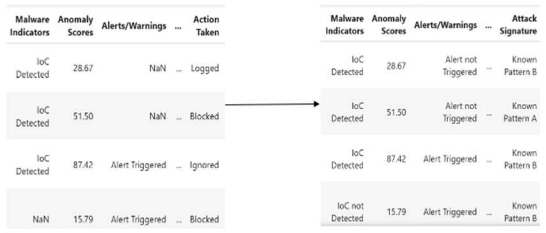
Figure 2: Key Attributes from Dataset

3.4 Data Preprocessing

Before the Model was subject to training, it underwent several data preprocessing techniques to help increase the quality and ability of consistency amongst the data points, such as:

3.4.1 Data Cleaning

Data cleaning was performed to address missing, inconsistent, and redundant information across the dataset. Missing or null values in attributes, such as Malware Indicators and Alerts/Warnings were filled in with an appropriate placeholder of "Malware Not Detected" or "IoC Not Detected" for consistency [Fig. 3]. During data cleaning, attributes that did not provide adequate information towards the cyber threat prediction, were removed to reduce noise during training. Log Source, Proxy Information, and Username fields were deemed as irrelevant attributes as they did not provide meaningful information about the eventual usefulness of the attribute towards a cyber threat prediction.



Malware Indicators	Anomaly Scores	Alerts/Warnings	...	Action Taken
IoC Detected	28.67	NaN	...	Logged
IoC Detected	51.50	NaN	...	Blocked
IoC Detected	87.42	Alert Triggered	...	Ignored
NaN	15.79	Alert Triggered	...	Blocked

Malware Indicators	Anomaly Scores	Alerts/Warnings	...	Attack Signature
IoC Detected	28.67	Alert not Triggered	...	Known Pattern B
IoC Detected	51.50	Alert not Triggered	...	Known Pattern A
IoC Detected	87.42	Alert Triggered	...	Known Pattern B
IoC not Detected	15.79	Alert Triggered	...	Known Pattern B

Figure 3: Null values of Columns such as Malware Indicators, Alerts/Warnings have been filled

3.4.2 Time Stamp Normalization

Temporal Convolutional Networks (TCNs) generally produce better predictions when input sequences are uniformly scaled with one another, especially in time series data. To this end, all timestamps in the data were converted to Coordinated Universal Time (UTC) and expressed in epoch seconds, such that all time series data had a temporal scale that was consistent and because of this, it was discounted the potential impact of differing time zones during the training of the model.

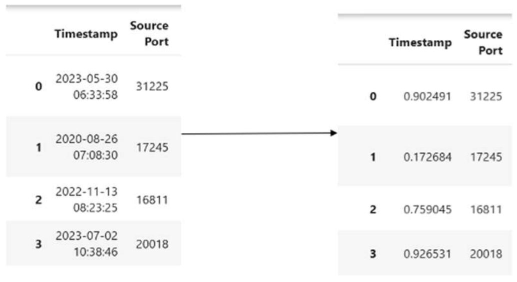
The conversion and normalization were performed as follows:

$$epoch\_seconds_i = (timestamp_i - \text{Jan } 1{,}1970) \times 86400 \quad (1)$$

After time conversion using (1), Min-Max normalization (2) was also used to scale the time values to a range [0,1][Fig 4]:

$$norm\_time_i = \frac{epoch\_seconds_i - min\_epoch\_seconds}{max\ \_epoch\_seconds - min\ \_epoch\_seconds} \quad (2)$$

This aided in training stability and convergence. Healthily scaled time values ensured the TCN learnt the time intervals proportional assets from either the epoch seconds or minutes.



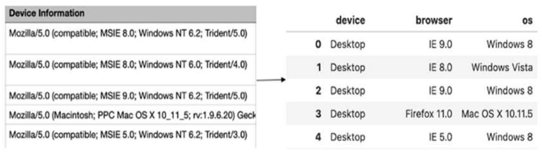
	Timestamp	Source Port
0	2023-05-30 06:33:58	31225
1	2020-08-26 07:08:30	17245
2	2022-11-13 08:23:25	16811
3	2023-07-02 10:38:46	20018

	Timestamp	Source Port
0	0.902491	31225
1	0.172684	17245
2	0.759045	16811
3	0.926531	20018

Figure 4: Timestamp has been normalized to [0,1]

3.4.3 User Agent String Parsing

The User Agent string in the dataset contains valuable context specific information related to the type of device, the operating system (OS) and the browser from which a web request was generated. A User Agent String Parser was used to separate the complex text field into structured sub-attributes - Device Type, Operating System and Browser Version.



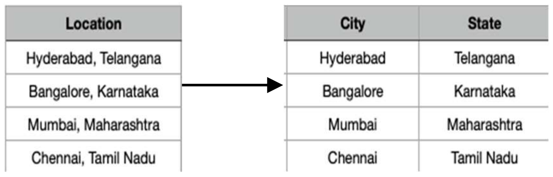
Device Information
Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.2; Trident/5.0)
Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0)
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.2; Trident/5.0)
Mozilla/5.0 (Macintosh; PPC Mac OS X 10_11_5; rv:1.9.6.20) Geck
Mozilla/5.0 (compatible; MSIE 5.0; Windows NT 6.2; Trident/3.0)

	device	browser	os
0	Desktop	IE 9.0	Windows 8
1	Desktop	IE 8.0	Windows Vista
2	Desktop	IE 9.0	Windows 8
3	Desktop	Firefox 11.0	Mac OS X 10.11.5
4	Desktop	IE 5.0	Windows 8

Figure. 5: User Agent String Has been Parsed into device, browser, os

3.4.4 Attribute Splitting

Composite attributes were separated into individual attributes for increased feature granularity. For example, the ocation attribute was separated into City and State attributes.



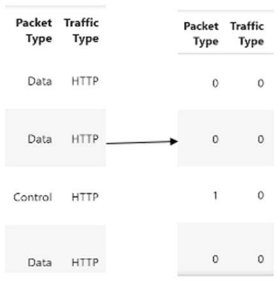
Location
Hyderabad, Telangana
Bangalore, Karnataka
Mumbai, Maharashtra
Chennai, Tamil Nadu

City	State
Hyderabad	Telangana
Bangalore	Karnataka
Mumbai	Maharashtra
Chennai	Tamil Nadu

Figure 6: Location Split into City & State

3.4.5 Label Encoding

Categorical attributes for Attack Signature, Attack Type, City, State, Protocol Type and Network Segment were numerically encoded with Label Encoding so the model could process non-numeric attributes.



Packet Type	Traffic Type
Data	HTTP
Data	HTTP
Control	HTTP
Data	HTTP

Packet Type	Traffic Type
0	0
0	0
1	0
0	0

Figure 7: Changed Categorical Values to Numerical

3.5 TCN Model Architecture

3.5.1. Model Design

The model[Fig 8] follows a sequential pipeline tailored to time-dependent network telemetry:

1. *Input Layer*: a sliding window of length $\boldsymbol{w}$ and feature width $\boldsymbol{d}$, i.e., $\mathbf{X} \in \mathbb{R}^{\boldsymbol{w}\times\boldsymbol{d}}$.
2. *TCN Layer*: two stacked temporal built with casual dilated 1-D convolutions
3. *Post-TCN Processing Block*: a compact head that stabilizes and compresses temporal features.

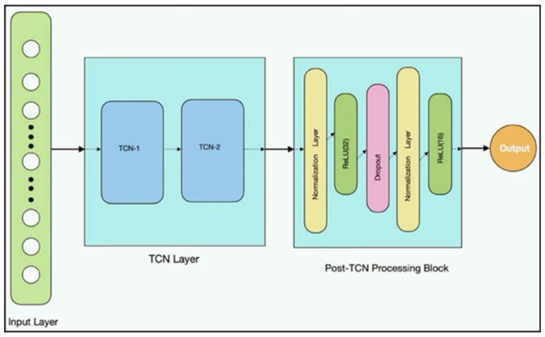


Figure 8: TCN Model Architecture

3.5.2: TCN layer

3.5.2.1: TCN-Block-1

This block expands the temporal receptive field while preserving strict causality. Four temporal blocks with increasing dilations capture short, mid, and long-range patterns without recurrence. Residual connections stabilize training and allow channel changes. The result is a feature map where each time step encodes ~16 prior steps of context (for $k = 2$)

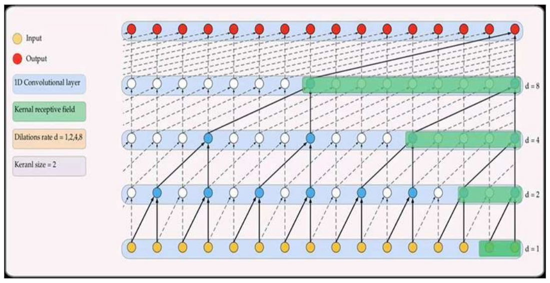


Figure 9: TCN-Block-1 Architecture

Input sequence: $X \in \mathbb{R}^{T\times C_0}$, Kernel size $k = 2$.
Dilations: $D_1 = \{1,2,4,8\}$.
Causal dilated 1-D convolution for dilation d and weights $W \in \mathbb{R}^{k\times C_{in}\times C_{out}}$:

$$(\mathrm{Conv}_d(x;W))_t = \sum_{i=0}^{k-1} W_i^T x_{t-di} \quad (3)$$

Normalization: $\mathrm{Norm}(z) = \gamma\frac{z-\mu}{\sigma} + \beta$. (4.1)
Activation: $\mathrm{ReLU}(z) = \max(0, z)$. (4.2)
Dropout: $\mathrm{Drop}(z) = M \odot z,\ M\sim\mathrm{Bernoulli}(q)$. (4.3)
Residual projection: $P(x) = \mathrm{Conv}_{1\times 1}(x)$ (4.4)

Temporal block with dilation d:

$$z_1 = \mathrm{Drop}(\mathrm{ReLU}(\mathrm{Norm}(\mathrm{Conv}_d(x;W_1))))$$
$$z_2 = \mathrm{Drop}(\mathrm{ReLU}(\mathrm{Norm}(\mathrm{Conv}_d(z_1;W_2)))) \quad (5)$$

$$\mathrm{TB}_d(x) = z_2 + \begin{cases} x, & C_{in} = C_{out} \\ P(x), & \text{otherwise.} \end{cases} \quad (6)$$

TB_i: the i-th transform block
$h^{(i)}$: the output feature of the i-th block, used as the input to the next block (i+1).

$$h^1 = TB_1(X) \quad (7.1)$$
$$h^2 = TB_2(h^1) \quad (7.2)$$
$$h^3 = TB_4(h^2) \quad (7.3)$$
$$Y_1 = TB_8(h^3) \quad (7.4)$$

Y_1: the final output after the last block.

3.5.2.2 TCN-Block-2

This block refines features produced by Block-1 and modestly extends temporal context. Using smaller dilation set, it denoises, sharpens phase-shifted patterns, and improves invariance[Fig.10]. Residual learning keeps gradients stable. The output retains causality and adds ~8 steps of extra context (for $k = 2$).

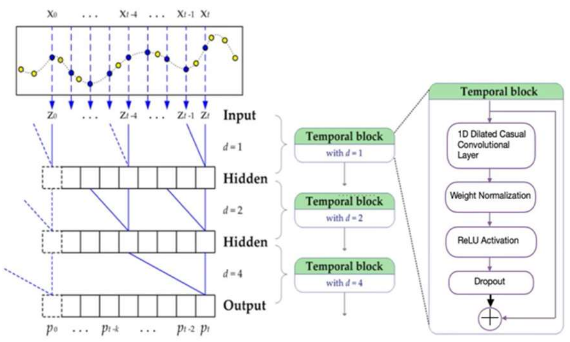

Input
d = 1
Hidden
d = 2
Hidden
d = 4
Output
Temporal block with d = 1
Temporal block with d = 2
Temporal block with d = 4
Temporal block
1D Dilated Casual Convolutional Layer
Weight Normalization
ReLU Activation
Dropout

Figure 10: TCN-Block-2 Architecture

Input $Y_1 \in \mathbb{R}^{T\times C_4}$, Kernel size $k = 2$,Dilations $D_2 = \{1,2,4\}$.
$\mathrm{TB}_d(\cdot)$is the temporal block defined above with its own weights (W_1, W_2).

$$h^{(4)} = TB_1(Y_1) \quad (8.1)$$
$$h^{(5)} = TB_2(h^{(4)}) \quad (8.2)$$
$$Y_2 = TB_4(h^{(5)}) \quad (8.3)$$

Y_2: Final output from the TCN layer

3.5.3 Post-TCN Layer

This block turns temporal features into task-ready representations. It first normalizes and linearly projects channels, then applies nonlinearity and dropout for regularization. A second normalization stabilizes the head. Finally, temporal aggregation feeds a linear classifier/regressor.

Input $Y_2 = \{y_t\}_{t=1}^{T}, y_t \in \mathbb{R}^{C_7}$.
Projection weights $W_p \in \mathbb{R}^{C_7 \times C_p}$, bias $b_p \in \mathbb{R}^{C_p}$.
Head weights $W_o \in \mathbb{R}^{C_p \times K}$, bias $b_o \in \mathbb{R}^{K}$
Temporal aggregation $Agg(\{a_t\}) = a_T$

$$\dot{y}_t = \mathrm{Norm}(y_t) \quad (9.1)$$
$$u_t = \dot{y}_t W_p + b_p \quad (9.2)$$
$$v_t = \mathrm{Drop}(\mathrm{ReLU}(u_t)) \quad (9.3)$$
$$\hat{y}_t = \mathrm{Norm}(v_t) \quad (9.4)$$
$$g = \mathrm{Agg}(\{\hat{y}_k\}_{t=1}^{(T-1)}) \quad (9.5)$$
$$o = g\, W_o + b_o \quad (9.6)$$

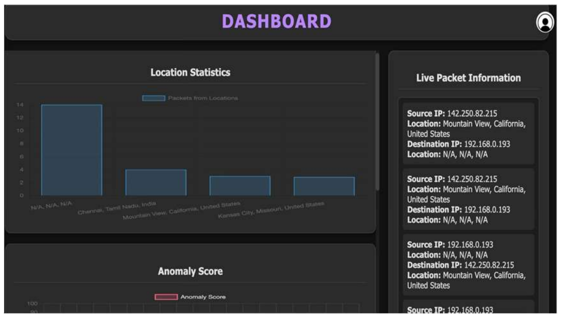
o is the output of the model which consists of values of anomaly score, threat type and threat severity.

3.6 ThreatScan Application Interface

The Dashboard is the principal visualization module of the application. It provides real-time visibility into network operations, anomaly detection, and threat intensity. The Dashboard consolidates data from many monitoring components to allow for an interactive visualization and analytical context for security administrators/analysts.

3.6.1 Location Statistics

This portal shows a bar-chart distribution of packets organized by their respective geographic origin[Fig. 11]. It aids in recognizing the geographic areas that account for the most network traffic rates, which assist in the identification of suspicious activity



Source IP: 142.250.82.215
Location: Mountain View, California, United States
Destination IP: 192.168.0.193
Location: N/A, N/A, N/A

Source IP: 142.250.82.215
Location: Mountain View, California, United States
Destination IP: 192.168.0.193
Location: N/A, N/A, N/A

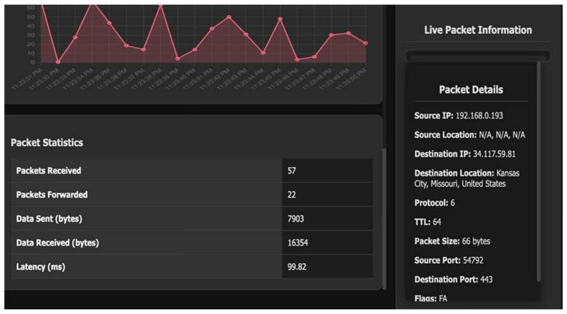
Source IP: 192.168.0.193
Location: N/A, N/A, N/A
Destination IP: 142.250.82.215
Location: Mountain View, California, United States

Source IP: 192.168.0.193

Figure 11: Dashboard Showing Location Statistics

3.6.2 Packet Statistics and Details

The packet statistics lists high-level metrics of the network: packets received, packets forwarded, latency (ms), and the total amount of bytes sent and received[Fig. 12]. Next to the packet statistics is the Live Packet Information panel that lists the information of packets live at the packet level including source and destination IPs, port numbers, protocols, TTL, and flags.



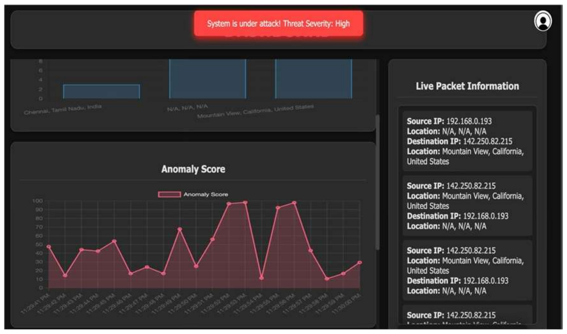
Packets Received	57
Packets Forwarded	22
Data Sent (bytes)	7903
Data Received (bytes)	16354
Latency (ms)	99.82

Source IP: 192.168.0.193
Source Location: N/A, N/A, N/A
Destination IP: 34.117.59.81
Destination Location: Kansas City, Missouri, United States
Protocol: 6
TTL: 64
Packet Size: 66 bytes
Source Port: 54792
Destination Port: 443
Flags: FA

Figure 12: Packet Statistics, Live Packet Information are shown in the dashboard

3.6.3 Anomaly Score and Threat Severity

The Anomaly Score Chart plots out the output generated by the DS-TCN, which continuously monitors traffic streams to detect variation from normal. The higher the peaks, the more severe the anomaly detected[Fig. 13]. The operationalization of data analytics and alerting provides an informative and accessible indication of system health, and provides the basis for rapid response and mitigation of a potential cyber-attack.


Chennai, Tamil Nadu, India
N/A, N/A, N/A
Mountain View, California, United States
Anomaly Score
Anomaly Score
Live Packet Information

System is under attack! Threat Severity: High

Source IP: 192.168.0.193
Location: N/A, N/A, N/A
Destination IP: 142.250.82.215
Location: Mountain View, California, United States

Source IP: 142.250.82.215
Location: Mountain View, California, United States
Destination IP: 192.168.0.193
Location: N/A, N/A, N/A

Source IP: 142.250.82.215
Location: Mountain View, California, United States
Destination IP: 192.168.0.193
Location: N/A, N/A, N/A

Source IP: 142.250.82.215

Fig. 13 Live Anomaly Detection shown in the Application

4. COMPARATIVE ANALYSIS

To evaluate the performance of the proposed Dual Stacked -Temporal Convolutional Network (DS-TCN) model, several metrics were used. These include Loss, Accuracy, Precision, Recall, F1-Score, and the Confusion Matrix. The results obtained were compared against other machine learning models such as Recurrent Neural Networks(RNN), Random Forest (RF), Support Vector Machine (SVM)..etc.

4.1 Loss Function

The loss function tells the difference between predicted and true outputs. The DS-TCN model minimizes the Binary Cross Entropy (BCE) loss, defined in eqn 10:

$Loss = \frac{-1}{N}\sum_{1}^{N}[\, y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\,]$ (10)

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

N : number of samples,

y_i : true label (0 or 1), and

$\hat{y}_i$: predicted probability.

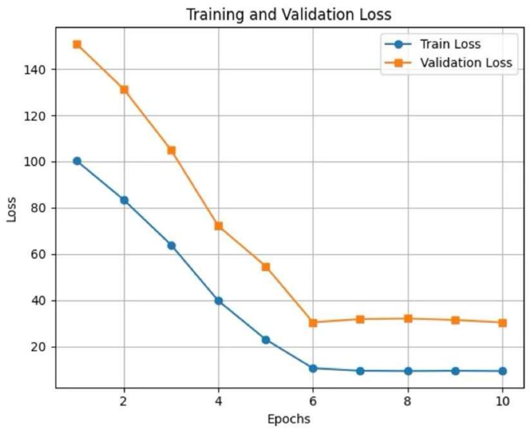


Figure 14: DS-TCN loss over time

4.2 Accuracy

Accuracy measures the ratio of correctly classified samples to the total number of samples (11):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (11)$$

TP: True Positives, TN: True Negatives,

FP: False Positives, FN: False Negatives.

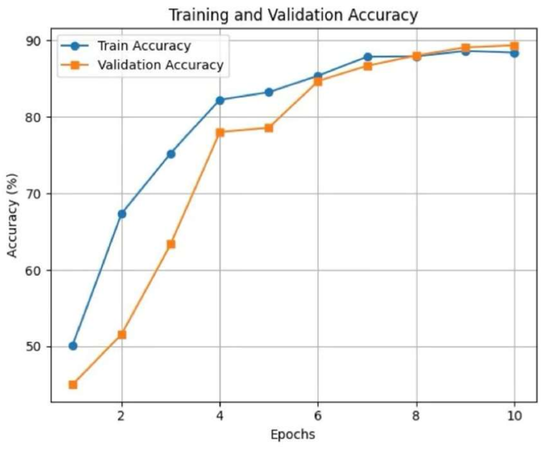


Figure 15: DS-TCN Accuracy over time

4.3 Precision, Recall & F1-Score

Precision tells how many of the samples predicted as positive are positive. (12)

Figure 16: Precision of DS-TCN Compared with other models over time

Recall measures the number of actual positives correctly identified by the model. (13)

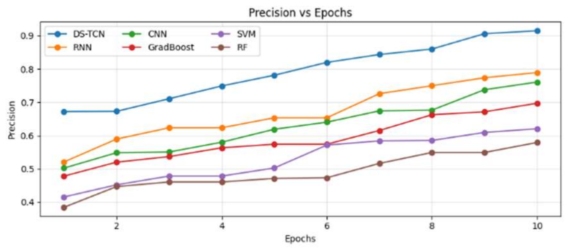


Figure. 17: Recall of DS-TCN Compared with other models over time

$$\text{Precision} = \frac{TP}{TP + FP}$$

The F1-Score is harmonic mean of precision and recall, providing a balance of both: (14)

$$\text{Recall} = \frac{TP}{TP + FN}$$

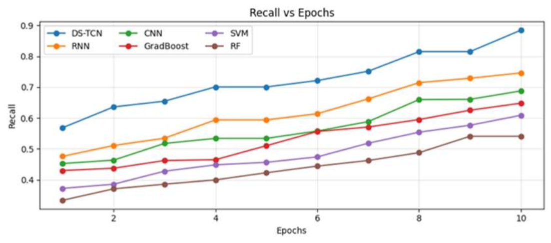


Figure 18: F1-Score of DS-TCN Compared with other models over time

4.4 Analysis of Classification Results

Confusion matrices provide a detailed understanding of classification performance, accompanying the attack type analysis with classification performance by attack severity. For the DDoS and Malware classes, the TCN achieved the highest number of true positives (8392 and 8822) across the three attack types, while the Intrusion class achieved 7693 true positives[Fig. 19]. The DDoS and Malware classes had exceedingly low misclassification of adjacent classes, suggesting strong interclass discrimination with regards to the model's predictive performance.

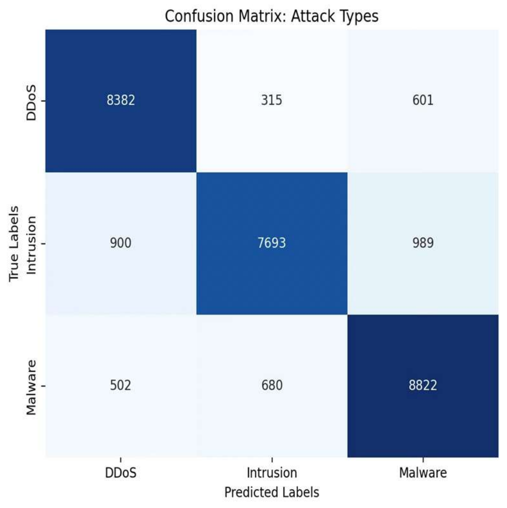


Figure. 19 Confusion Matrix for Attack Types Classified by DS-TCN

The confusion matrix from the severity level classification also indicated generally strong classification performance of the TCN across High, Medium, and Low classification labels. For High severity, the model correctly classified 9037 true positives, while Medium was associated with 8018 and Low with 7258 true positives[Fig. 20].

The relatively low number of misclassifications between adjacent classes indicates that there was general similarity for seizure patterns associated with High, Medium, and Low classes, even though they were misclassified.

As with other real-world examples, threats of differing intensity can be similar, thus, the TCN consistently demonstrates promising categorical and ordinal prediction performance while continuing to maintain classification decision boundaries despite overlap of input patterns**.**

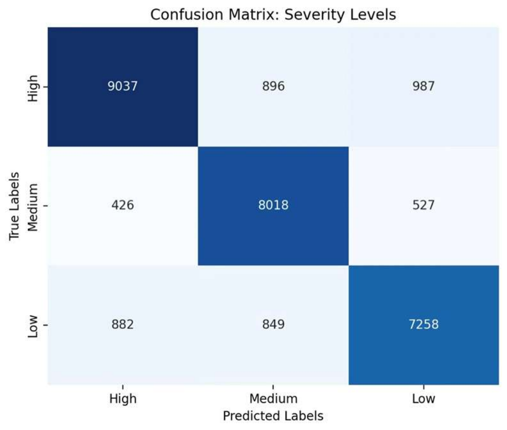


Figure. 20 Confusion Matrix for Severity Level of Each Attack Classified by DS-TCN

4.5 Performance Comparison

To validate the effectiveness of TCN model, it compared with other models trained on same dataset:

Table-1: Comparison of Accuracy & F1-Score of different models with DS-TCN

Model	Accuracy	F1-Score
DS-TCN	88.9 %	87.5%
RNN	78.4 %	76.9%
CNN	73.2%	71.0%
Gradient Boosting	68.5%	66.7%
SVM	63.8%	61.5%
Random Forest	58.1%	55.3%

The comparative review reveals that while the Double Stacked -Temporal Convolutional Network (DS-TCN) provides the optimal performance 88.9 % accuracy [Fig. 15] and 87.5 % F1-score by being capable of learning long-range temporal dependencies and interactions at scale within the context of cybersecurity traffic, it does so with dilated-causal convolutions that allow it to learn in parallel respecting temporal order and minimizing gradient decay to guarantee stable convergence, the RNN ranks second with an accuracy of 78.4 % and F1 of 76.9 % with acceptable performance in respect to short-term dependencies but limited long-term temporal dependencies owing to the vanishing-gradient problem over longer sequences and our CNN model achieved an accuracy of 73.2 % and F1 of 71.0 %[Fig. 18] owing to feature extraction of local temporal and spatial dimensions, but with less receptive-field flexibility relative to DS-TCN.

Of the classical machine learning baselines, the Gradient Boosting (68.5 % / 66.7 %) [Fig. 18] and the Support Vector Machine (63.8 % / 61.5 %) [Fig. 18] proved to be decent classifiers with respect to the tabular features while capturing nonlinear processes in the data but did not perform as well with relationships in the raw sequence data. Lastly, the Random Forest performed the worst in regard to modeling the sequence data (58.1 % / 55.3 %); as it cannot validate the time dependent patterns formed in the data.

Overall, the DS-TCN stands out as robust and accurate framework for cybersecurity threat prediction, effectively modeling temporal correlations across sequential data streams.

5. OBSERVATIONS AND RESULTS

5.1 Key Observations

The DS-TCN exhibit remarkable learning stability and overall classification performance across metrics evaluated. The training and validation loss curves steadily decline together ending near the 8th epoch while the accuracy was increasing during the same time frame ending near 89% [Fig. 15]. The very small gap between the training and validation accuracy demonstrates low overfitting and predictability on new (unseen) data. The precision, recall and F1 score metrics also generally improved together and observed near 0.85-0.87[Fig. 18] indicating a reasonable sensitivity and specificity trade-off. The confusion matrices also demonstrate most of the classifications were aligned with their correct class label[Fig. 19, 20].

Performance comparisons across various baseline models further supported the DS-TCN's performance superiority. In particular the DS-TCN achieved the highest accuracy (88.9%) and F1 score (87.5) compared to all architecture's evaluated as well as, the RNN, hybrid CNNs, and ensemble approaches such as RF and GB[Fig.18][Table-1]. The DS-TCNs ability to capture long-term temporal dependency effects through dilated convolutions gives the model a significant advantage when modeling sequences - with respect to stability, accuracies and performance. Overall, these findings demonstrate a promising architecture type with results is impressive and there is great precision, stability, increased computational efficiencies when learning complex time-series threat patterns than other recommended deep learning and traditional classifiers.

5.2 Experimental Results

The studies results indicate that the DS-TCN model not only quickly reduced the level of loss[Fig.14], but remained stable in accuracy for both training and validation datasets during the training phase[Fig 15]. The training graphs reflected rapid oscillation and sufficiently stable learning when only a handful of epochs had been completed in the training without solely having high-diverging capabilities. Therefore, the evolution of precision, recall, and F1 score reflects good indications for prediction quality[Fig.16,17,18]. More specifically, F1 sentimental is a strong indication for showing balance andF severity classification.

This indicates that the TCN model can provide some level of separation with the features to limit false detections and can give accurate multi-layer predictions. In specific terms, TCN saw 88.9% accuracy, and F1 score of 87.5% for the performance indicators, which were more favorable than all baselines, and indicate that architecture using TCN for temporal data is learning better sequences. The experimental evidence indicates the model generalizes well enough to reasonably distinct categories but still provides accuracy for categorical (attack types) and ordinal (severity levels) outputs respectively[Fig. 19,20]. Therefore, this supports that Custom TCN is a real-time cyber-threat detection model that demonstrates effectiveness and significant accuracy, while maintaining some degree of stability and more interpretation of detection.

5.3 Future Prospect

Custom-built intelligent cyber defense systems are very much possible, as highlighted by the multipath TCN model. As the architecture is still in the early stages of development, there is potential to expand the architecture for future applications of real-time monitoring of traffic flows and adaptive measures for security event actions to enhance the speed of anomalous behavior detection and the dynamic classification of attacks as they evolve. Additionally, TCNs can be integrated with edge computing, which enhances the model post-deployment in larger Security Operation Centers (SOC).

Models could enhance the future value of multi-modal data fusion/sampling and combining text logs, images, and environmental sensor data, and provide greater intelligence value in understanding high threat intelligence operating frameworks. The model could also develop toward future incremental improvements toward continuously learning pipelines by exposing new mitigation vectors to develop resilience against zero-day attacks. Explainability modules, possibly with some visualization dashboards, could help human users understand system processes and behaviors as a cyber security analyst. Along with this approach, it will stretch and not only TCN based disasters with their efforts to cut-edge cyber defense capabilities which include a value proposition for clarity such as: tight prediction leading to alert generation; mitigation containment; and cyber security policy recommendation.

REFERENCES:

[1] A. D. Kent and M. Thompson, "The evolution of DDoS attacks and mitigation strategies," IEEE Security & Privacy, vol. 21, no. 3, pp. 45–56, 2023.
[2] S. Srinivas and J. Patel, "Limitations of rule-based intrusion detection in modern threat landscapes," Journal of Information Security Research, vol. 18, no. 2, pp. 120–132, 2022.
[3] IBM Security, Cost of a Data Breach Report, 2023.
[4] CrowdStrike, 2024 Global Threat Report: Breakout Times and Adversary Insights, 2024.
[5] Y. Zhang and T. Hu, "Applications of machine learning in cyber threat detection," IEEE Access, vol. 10, pp. 45123–45137, 2022.
[6] S. Bengio, P. Frasconi, and Y. LeCun, "Learning long-term dependencies with gradient descent is difficult," IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157–166, 1994.
[7] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," arXiv preprint arXiv:1803.01271, 2018.
[8]A. M. Rawat and K. Bose, "Temporal convolutional networks for intrusion detection in cybersecurity," Computers & Security, vol. 132, 2024.
[9] "A Lightweight Network Intrusion Detection System Based on Temporal Convolutional Networks and Attention Mechanisms", Computer Fraud & Security,April 2025.
[10] "SDN Anomalous Traffic Detection Based on Temporal Convolutional Network," Applied Sciences, 2025.
[11] Ye, X., et al., "Daily insider threat detection with hybrid TCN transformer architecture," Scientific Reports, August 2025.
[12] "Advanced Temporal Convolutional Network Framework for Intrusion Detection in Electric Vehicle Charging Stations," IEEE International Conference on Integrated Intelligence, September 2025.
[13] "A Machine Learning-Based Ransomware Detection Method for Attackers' Neutralization Techniques Using Format Preserving Encryption," MDPI, 2025.
[14] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," arXiv preprint arXiv:1803.01271, 2018.
[15] J. Brown and A. Smith, "Design of integrated cybersecurity platforms for real-time intrusion prevention," IEEE Transactions on Network and Service Management, vol. 20, no. 2, pp. 512–525, 2023.
[16] T. Zhang and K. Li, "Real-time traffic preprocessing and feature generation for anomaly detection," Computers & Security, vol. 134, 2024.