TEPIAN Vol. 6 No. 2 (June 2025) 85-96 p-ISSN 2721-5350 e-ISSN 2721-5369
https://doi.org/10.51967/tepian.v6i2.3260 © 2025 Tepian Politeknik Pertanian Negeri Samarinda

Optimisation of Network Logs for Fake Bandwidth Classification using CNN

Azriel Christian Nurcahyo *
Design and Technology Centre, School of Computing and Creative Media, University of Technology Sarawak, 96000, Malaysia
pic24030001@student.uts.edu.my
**Corresponding author*

Huong Yong Ting
Design and Technology Centre, School of Computing and Creative Media, University of Technology Sarawak, 96000, Malaysia
alan.ting@uts.edu.my

Abdulwahab Funsho Atanda
Design and Technology Centre, School of Computing and Creative Media, University of Technology Sarawak, 96000, Malaysia
abdulwahab@uts.edu.my

Submitted: 2025-03-26; Accepted: 2025-04-21; Published: 2025-06-01

Abstract— The goal of this study is to enhance the classification accuracy of fake bandwidth using a CNN model, leveraging network logs collected in real-time. For this research, the network logs from the Cyber Security Laboratory of the University of Technology Sarawak are used as a dataset for training the CNN model. The dataset consists of 20 days of continuous network activity logging, which results in over 500,000 data entries. According to the model evaluation results, the trained CNN model demonstrated high accuracy in classifying genuine bandwidth (Precision: 0.92, Recall: 0.95). Moreover, it achieved considerable success in detecting fake bandwidth (Precision: 0.89, Recall: 0.90) and the no heavy activity category (Precision: 0.98, Recall: 0.84). Analysis of Loss Over Epochs showed a dramatic decrease in loss during the training phase, with optimal convergence reached by epoch 2000. Identifying these characteristics enables monitoring systems to classify network data with high certainty, detecting bandwidth manipulation in expansive networks. Thus, this research aids the design of dynamic network monitoring systems that require minimal response time while maintaining high accuracy.

Keywords— Convolutional Neural Network (CNN), Fake Bandwidth, Network Logs, Bandwidth Utilization

I. INTRODUCTION

The World is in an era where virtually all human interactions across different sectors require a stable Internet connection, which is used to facilitate countless activities (Dolgui, A et al., 2021). In an AI-powered world, without the use of the Internet, all work models involving software and applications would become nearly impossible, especially those requiring high connectivity (Giordano et al., 2022). The Internet is critical for software and applications, especially those providing essential services, to function effectively in today's digital environment (Sari, M et al., 2023). Many enterprise devices or network loads are always associated with high bandwidth usage (Shi, J. et al., 2024). On a broader spectrum, high as well as low levels of an enterprise may not be accurately depicted as the real user of the bandwidth based on certain activities (Oji et al., 2021). Exceptional situations can be found where network bandwidth is referred to as fake bandwidth or manipulated bandwidth. This is known as a means to artificially make network bandwidth appear high from a speed test, while no actual data traffic use exists that continuously works in conjunction with the results provided by the speed test. Additionally, in certain instances, the Service Licence Agreement outlines a sizable bandwidth figure, but at certain times, its ability to deliver service in line with the parameters of the agreement is hindered, hence the service is burdened and the level of access which ought to be available to major users is not probable (Razian et al., 2022). Such problems arise where users or businesses have contracts with ISPs for a certain Mbps figure, but instead of the contracted value, they are given lower values or, at times, only some fraction of the figure during some other periods (Hapsari, 2022). There are also other situations where bandwidth management is performed as per the agreement, but for some reason, only half of the anticipated bandwidth is received in the monitoring of upload and download activities (Jhaveri et al., 2021). Moreover, having Internet outages for more than the specified durations in the agreement will, without doubt, pose a problem for many, especially those whose business functionality is highly dependent on the Internet (Biswal et al., 2024).

Bandwidth inconsistency caused by the availability of bandwidth manipulation methods can be extremely harmful to consumers and service providers alike, as it results in Quality of Service (QoS) degradation. High latency or delay in data communication occurs when realized bandwidth falls below the commitments of the Service Level Agreements (SLAs), causing a notable effect on overall user experience for applications such as video conferencing. With bandwidth classification, it is expected that the optimisation of ISP services can be improved since ISPs can ensure that the provided bandwidth meets the requirements of the users and prevents wastage of network resources. Additionally, a better classification of bandwidth can enhance the Service Level Agreements (SLAs) between Internet Service Providers (ISPs) and their customers, while increasing the effectiveness of resource allocation at the same time. This improvement allows ISPs to provide bandwidth with more precision according to the particular application demands, reduces unnecessary bandwidth usage, and ensures sufficient bandwidth for critical user requests (Collier, 2024; Gomes et al., 2016).

An examination of how service processes are run demonstrates that these trends can prove to be economically expensive as they lead to the decline of service quality, waste of service resources, or even incorrect decisions regarding service router infrastructure management (Ye et al., 2023). More technically, loss of bandwidth, false bandwidth, or manipulation of queue models, can in large unattended settings such as data centres, educational institutions, or corporations, damage the quality of service (QoS) and pose a risk to modern network management. Bandwidth that is lost, for example, during prolonged inactivity, is often not tracked by monitoring systems, thus preventing them from detecting what truly should be flagged: when data traffic appears to be normal (Jhaveri et al., 2021). Network intrusion detection systems based on statistics or models include clustering methods, distribution-based methods, graphical methods, and density-based methods (Komadina et al., 2024). Regardless of the existence of alternative solutions, approaches to network analysis will always have gaps due to the rapid development of internet technology, thus, improvement of the actual network log data is always going to be required (Alhammadi, O., & Abul, O., 2024).

A broad range of research conducted by Zhao et al. (2024) and Cermák et al. (2023), corroborates that both manual and automated network surveillance techniques have a myriad of shortcomings when it comes to detecting anomalies such as loss of bandwidth, especially when there is no database to monitor it (Duan, et al., 2022). Zhao et al. (2024) demonstrate that conventional bandwidth traffic monitoring systems fail to recognise abnormal activities because the daily network traffic that is captured, and often conducted manually, ‘appears’ normal but is, in fact, twelve hours' worth of non-detection (Zhao et al., 2024). This was echoed by Cermák et al. (2023), stating that analysis of system network logs not only takes significant amounts of time, but also heavily depends on the judgment of the analyst (Cermák et al., 2023), hence, introducing the issue of bias and misinterpretation. AsSadhan et al. (2020) suggest, automatic systems employing rudimentary upper and lower threshold boundaries are exceedingly problematic as they tend to have high rates of false positives and negatives because, unfortunately, the simplistic approach does not allow accommodating the ever-changing dynamics of network traffic (AsSadhan et al., 2020). Onietan et al. (2023) claimed that rule-driven systems still have some deficiencies in recognising new or undetected patterns, having to do with the condition of network traffic, because such systems base their action log substantiated behaviours on already documented phenomena. It was also asserted that the commonly used SNMP and NetFlow commercial monitoring tools do not enable full visibility into traffic behaviour at the application level, where most bandwidths are contended (Zhang, et al., 2024). Moreover, it was also noted that marker-based intrusion detection systems are not fully reliable in anomaly detection; they only sense threats defined in advance (Ring, et al., 2021).

Saha et al. (2020) pointed out that traditional methods of dealing with network traffic flows are not very effective when it comes to dynamic changes in internet network traffic because they rely on fixed parameters. In the same vein, Zhang et al. (2024) argued that offline batch monitoring systems cannot retain important real-time information during periods of high activity and data volume. Dong et al. (2021) noted that a large number of automated network anomaly detection systems do not differentiate between genuine and non-genuine network traffic surges because there is no understanding of the information that is to be analysed before processing. A noteworthy insight here is that, although machine learning is starting to be used for network supervision, most of the systems tend to overfit and fail to generalise to new forms of anomalies, such as excessive use of bandwidth (Wang et al., 2021). In their research on the application of CNN for cyber-attack detection, Najar et al. (2024) mentioned that CNN’s prowess in this field is yet to be leveraged in the context of bandwidth DDoS cases. This was also noted in the work of Fotiadou et al. (2021), asserting that while CNN is capable of identifying abnormal network traffic behaviour, applying it to network logs is challenging unless powerful servers for massive network log data are utilised. In Cyber Security, CNN has proved to be able to identify DDoS attacks with up to 20% higher accuracy than traditional models. Yet, one of the biggest challenges is presented by non-IID (non-independent and identically distributed) data, i.e., the dataset is compromised by drastically different data distributions per device. In solving the non-IID problem, more intricate and elaborate algorithms must be employed to resolve the inconsistency of data distribution so that data retrieval and identification become more accurate and efficient (Lv et al., 2022). CNN in bandwidth utilization has been addressed in bandwidth forecasting with the aid of edge based on a CNN hybrid architecture. Yet, even though the results of testing reflect tremendous improvements in MAE and RMSE accuracy, the model could be less effective in very dynamic scenarios or where there are sudden spikes in bandwidth utilization. Consequently, the accuracy of the predictions made by the model improves only when incoming data exhibit stability, but in extreme situations such as DDoS attacks or abrupt network changes, the reliability of the prediction outcomes may be compromised, thereby rendering the detection of actual real time data challenging (Wen et al., 2022).

Moreover, Alrubayyi et al. (2023) further emphasised that most existing AI systems lack necessary accuracy in the real world due to the limitations of the training datasets used, thus complicating bandwidth analysis in IoT devices even further. In addition, Šabanovic et al. (2024) reported that real-time AI-based monitoring systems still cannot fully sense micro anomalies such as network bandwidth spikes in 5G at a false negative rate of 2.37% on a persistent basis because of a high amount of noise in log data, which requires structured data logs. The same goes for the research by Magnani et al. (2022), which also pointed out that there remains a gap in the integration of real-time network monitoring systems with self-adapting features to cope with abnormal network traffic. Collectively, these previous studies demonstrate that there is still a large gap in the market for effective network monitoring technologies set against the accurate detection and manipulation of bandwidth from real-time data streams. Therefore, there must be a different solution that is not just overriding and flexible, but also capable of transforming logs to accurate data in real-time every second, wherein the network log data is categorised as such, as proposed by the deep learning model of this study.

In a prior analysis, the researcher (Zhuang, 2024) stated that the integration of artificial intelligence and deep learning technologies has become a valuable tool for identifying intricate phenomena in massive datasets. In this study, CNN is more suitable for classification because it has greater advantages compared to other algorithms. For example, it has a high accuracy rate of 99%, whereas SVM has 74%. The CNN model employs a deep learning architecture, which enables the network to extract features automatically, thereby enhancing generalisation capabilities for complicated tasks like multi-class classification. Conversely, SVM, although efficient for less complex tasks, is inefficient when dealing with high-dimensional data and more than two classes, hence a greater classification error rate. Thus, CNN's capacity to learn from hierarchies in data and automatically extract features makes it outshine SVM in intricate classification tasks (Fadlil et al., 2022). Moreover, Lyu and Liu (2021) confirmed that the Convolutional Neural Network (CNN) architecture, which is often considered one of the most famous, was first developed for image analysis, but is now being extended to other fields like text processing. Similarly, Pham et al. (2023) demonstrated the use of CNN in detecting hidden patterns for cyber-attack classification from network logs. Similarly, work by Antonius et al. (2023) proved the successful application of CNN in modelling network traffic anomalies with high accuracy. Nevertheless, some of the previous studies that specifically applied CNN have bypassed the applicability of real network logs with many lines of data, say above half a million or even 700 thousand lines. This is mainly because there are very few means of obtaining network log data for analysis, most commonly having to depend on Wireshark, daily speed tests, or metered attacks from an observing IP address. This creates a gap in the literature that warrants further investigation, particularly in the area of identifying illegitimate bandwidth.

This study is developed and carried out to fill the gap in research concerning fake bandwidth as compared to genuine bandwidth, with the growing need for internet use today. The network model that is tested in this research is the backbone model of the internet at the University of Technology Sarawak from the public domain in the Cyber Security lab. Then, real-time network log data is captured every second and subsequently categorised to estimate the levels of fake bandwidth, genuine bandwidth, and no heavy activity using the CNN algorithm. The major novelty of this method lies in two features: how the network performance is improved for data traffic that is received every second, something that has not been done in earlier studies. Then, the log labels are assigned every second for the fake activity model, the genuine activity model, and the no heavy activity model. The log data is processed and stored in what is referred to as the first part of the training data. Twenty days later, real data, log data collected over twenty-four hours each day, is uploaded as real data for feature extraction, which is the process of transforming raw log data into numerical forms that can be utilised by a CNN model. As for the results of this particular study, the model aims to do more than just classify network attacks or types of traffic; it also aims to classify the amounts of detected fake and real (genuine) bandwidth, as well as periods of inactivity (no heavy activity), all within the constraints of 2000 epochs, or as many iterations needed to optimally showcase the accuracy of the created CNN. The accuracy of the CNN's classification is attributed to its architecture, compared to conventional approaches like Random Forest or Naive Bayes, which can better manage noise in the data and capture spatial dependencies within the data, hence yielding more precise classification results (Abdfilminaam et al., 2024).

This study aims to construct an automated log extraction system on the internet backbone network using network log optimisation, and subsequently conduct deep learning analysis that can store, retrieve, and process vast amounts of log data, identify phantom bandwidth, and classify it at the University of Technology Sarawak. The system should work in actual network environments for further studies in network logs and deep learning towards assisting system administrators to monitor the system performance and minimise the effect of bandwidth abuse or fake bandwidth. This study also contributes to the theoretical advancement of more flexible and applicable deep learning approaches outside their traditional boundaries by employing the capabilities of CNN in non-image domains like network logs, especially in the real-time management of data captured every second of the day, averaging 25,000 to 35,000 rows a day. This study provided a classification of three criteria from the summation of the abnormal bandwidth which is genuine bandwidth, fake bandwidth, and no heavy activity bandwidth, and each epoch will show accuracy loss validation accuracy validation loss and learning rate values while this study aims to set 2000 epochs to get the best results of CNN.

II. Research Methods

A. *The Design of The Network Model*

The methodology of this study is divided into three parts: the design of the network model, network data logs, and the CNN model. The first diagram comprises the design of a distributed computer network starting from the ISP side to the public IP address 60.53.x.x/26, which belongs to Telkom Malaysia Berhad and is one of the public routes utilised by this research as the Cyber Security Laboratory at the University of Technology Sarawak. The purpose of the public IP connection is to facilitate stable bandwidth conditions for the purpose of analysing and optimising bandwidth management. The first step is to set up the network infrastructure, which comprises the backbone router, core switch, and dynamic DHCP to the Access Point, including users in Lab 4. In the backbone and core switch configuration part, it consists of DNS setup, firewall configuration, and bandwidth management turned off by the queue and switch to bypass a 100 -1000 Mbps connection. It includes categorically active network configuration, automatic logging for 20 days. The network configuration is followed by deep learning with the use of real-time log data, as illustrated in Figure 1.

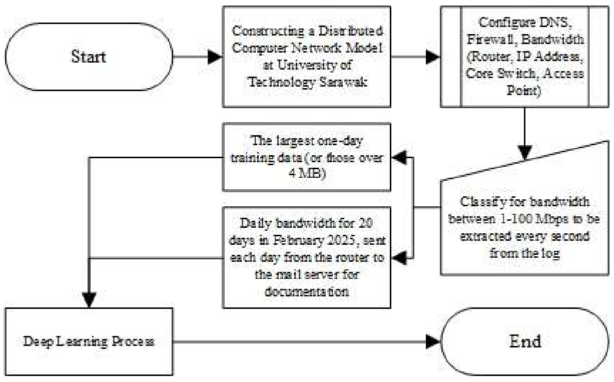
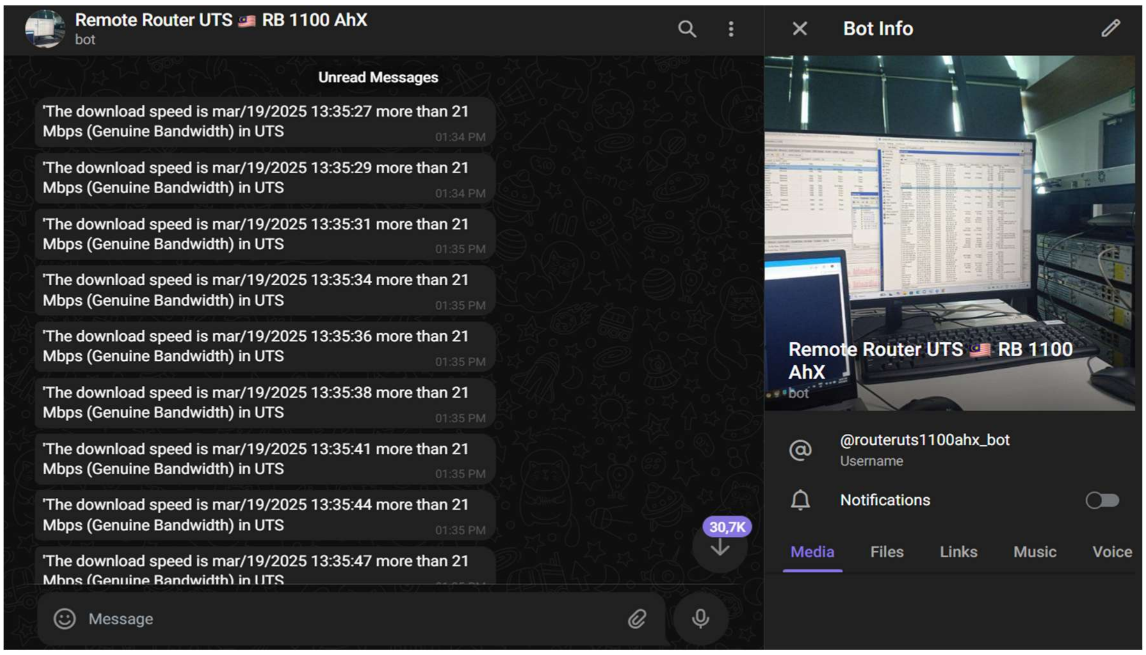

Start
Constructing a Distributed Computer Network Model at University of Technology Sarawak
Configure DNS, Firewall, Bandwidth (Router, IP Address, Core Switch, Access Point)
The largest one-day training data (or those over 4 MB)
Classify for bandwidth between 1-100 Mbps to be extracted every second from the log
Daily bandwidth for 20 days in February 2025, sent each day from the router to the mail server for documentation
Deep Learning Process
End

Figure 1. Network Model / Log Optimization

Moving on, the log data collection section seeks to locate and fetch the largest training data for a single day. The training data sample includes files larger than 4 MB in size, which were collected between 12:01 on 20th February and 12:00 on 21st February 2025. Moreover, monitoring of network traffic is performed for 20 days in February 2025. This information is captured at the router and transmitted to the mail server, while being monitored in real time through Telegram using the ID = @routeruts1100ahx_bot which is actively uploading files with the user ID and download speeds of 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85,90, 95, 100, 105, 110, 115, 120, 125 until 200 Mbps per second. “The download speed on 2nd March 2025 11:48:26 was more than 21 Mbps (Genuine Bandwidth) in UTS” is one of the examples given from the obtained model. Figure 2 is used to illustrate the Telegram model that was developed to showcase the automated active traffic data collection that comes from the API-key from Mikrotik 1100 AHx.

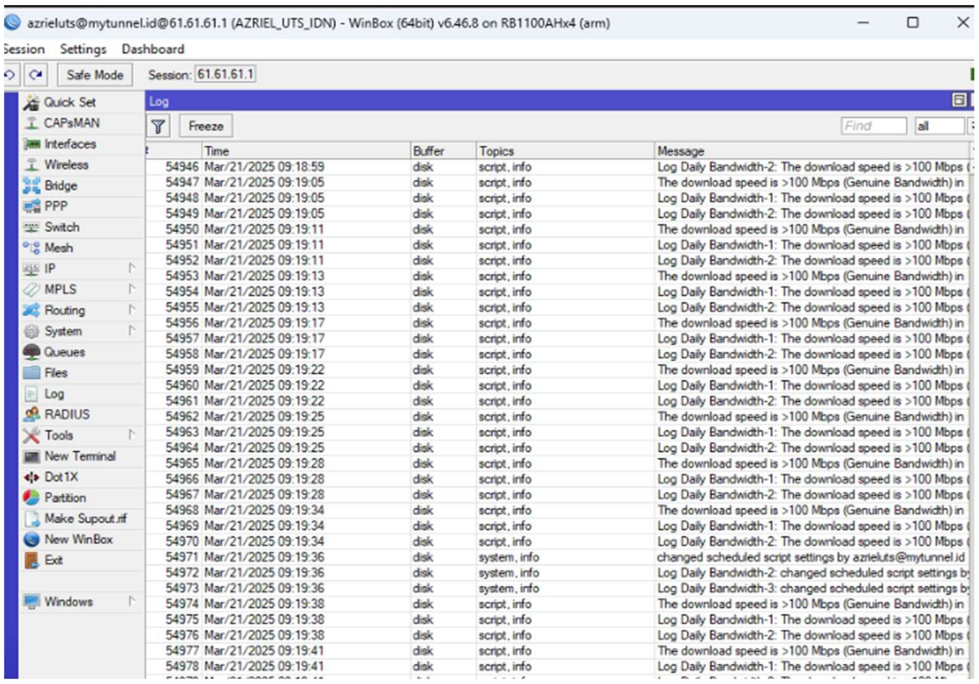


'The download speed is mar/19/2025 13:35:27 more than 21 Mbps (Genuine Bandwidth) in UTS 01:34 PM
'The download speed is mar/19/2025 13:35:29 more than 21 Mbps (Genuine Bandwidth) in UTS 01:34 PM
'The download speed is mar/19/2025 13:35:31 more than 21 Mbps (Genuine Bandwidth) in UTS 01:35 PM
'The download speed is mar/19/2025 13:35:34 more than 21 Mbps (Genuine Bandwidth) in UTS 01:35 PM
'The download speed is mar/19/2025 13:35:36 more than 21 Mbps (Genuine Bandwidth) in UTS 01:35 PM
'The download speed is mar/19/2025 13:35:38 more than 21 Mbps (Genuine Bandwidth) in UTS 01:35 PM
'The download speed is mar/19/2025 13:35:41 more than 21 Mbps (Genuine Bandwidth) in UTS 01:35 PM
'The download speed is mar/19/2025 13:35:44 more than 21 Mbps (Genuine Bandwidth) in UTS 01:35 PM
'The download speed is mar/19/2025 13:35:47 more than 21 Mbps (Genuine Bandwidth) in UTS

Figure 2. Real-Time Monitoring Model on @routeruts1100ahx_bot

The Usage Networking system logs are output into two partitions; the first being the auto-originating logs from the mail server through the RB1100 AHx connection, while the second is real-time monitoring output, which gets streamed straight to the Telegram bot. The next stage is classifying the bandwidth, where the network utilisation is further divided into 1 Mbps ranges up to 100 Mbps. This classification can be done by extracting the log data every second, which allows for a more precise and further optimised bandwidth meter pattern, as described in Figure 3. Having three data points every second translates into a maximum of 55,000 lines worth of data in an astonishing day. These logs are very important because they tell a story of the trends in bandwidth utilisation and the anomalies that could be present in the network traffic in Lab 4, University of Technology Sarawak, Figure 3.

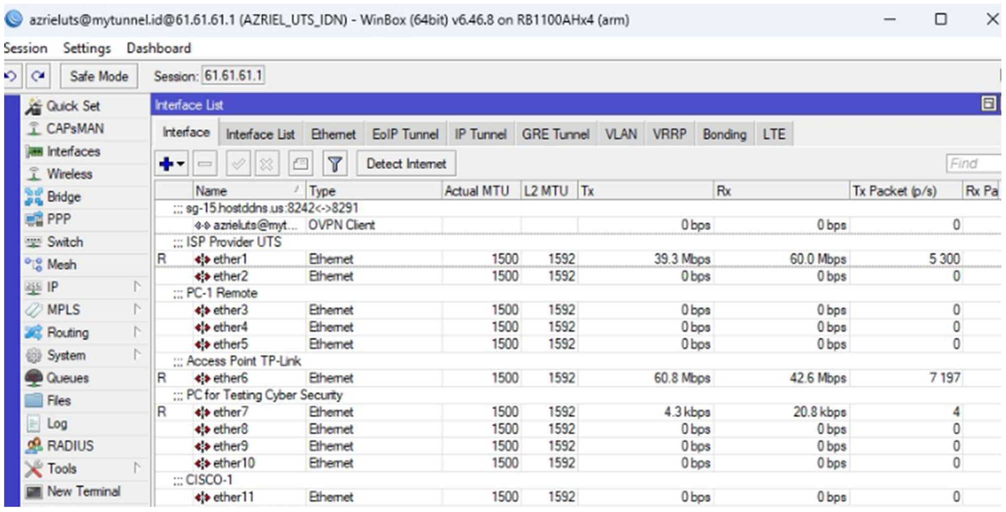
Nurcahyo, A. C., Yong , T. H., & Atanda, A. F. (2025). Optimisation of Network Logs for Fake Bandwidth Classification using CNN. TEPIAN, 6(2). https://doi.org/10.51967/tepian.v6i2.3260



```
#      Time                  Buffer  Topics        Message
54946  Mar/21/2025 09:18:59  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54947  Mar/21/2025 09:19:05  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54948  Mar/21/2025 09:19:05  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
54949  Mar/21/2025 09:19:05  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54950  Mar/21/2025 09:19:11  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54951  Mar/21/2025 09:19:11  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
54952  Mar/21/2025 09:19:11  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54953  Mar/21/2025 09:19:13  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54954  Mar/21/2025 09:19:13  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
54955  Mar/21/2025 09:19:13  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54956  Mar/21/2025 09:19:17  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54957  Mar/21/2025 09:19:17  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
54958  Mar/21/2025 09:19:17  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54959  Mar/21/2025 09:19:22  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54960  Mar/21/2025 09:19:22  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
54961  Mar/21/2025 09:19:22  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54962  Mar/21/2025 09:19:25  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54963  Mar/21/2025 09:19:25  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
54964  Mar/21/2025 09:19:25  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54965  Mar/21/2025 09:19:28  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54966  Mar/21/2025 09:19:28  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
54967  Mar/21/2025 09:19:28  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54968  Mar/21/2025 09:19:34  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54969  Mar/21/2025 09:19:34  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
54970  Mar/21/2025 09:19:34  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54971  Mar/21/2025 09:19:36  disk    system, info  changed scheduled script settings by azrieluts@mytunnel.id
54972  Mar/21/2025 09:19:36  disk    system, info  Log Daily Bandwidth-2: changed scheduled script settings by
54973  Mar/21/2025 09:19:36  disk    system, info  Log Daily Bandwidth-3: changed scheduled script settings by
54974  Mar/21/2025 09:19:38  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54975  Mar/21/2025 09:19:38  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
54976  Mar/21/2025 09:19:38  disk    script, info  Log Daily Bandwidth-2: The download speed is >100 Mbps (
54977  Mar/21/2025 09:19:41  disk    script, info  The download speed is >100 Mbps (Genuine Bandwidth) in
54978  Mar/21/2025 09:19:41  disk    script, info  Log Daily Bandwidth-1: The download speed is >100 Mbps (
```

Figure 3. Bandwidth log on RB 1100 AHx Lab 4, University of Technology Sarawak

This stage uses created logs that contain bandwidth activities, inclusive of the fake, genuine, and inactive values, alongside login information, script alterations, and captured activities saved on the disk. The next step is implementation using deep learning techniques, particularly neural network models. Applying these techniques allows for an analysis of the established patterns to classify the different percentages of genuine, fake, and inactive users based on their daily activity history. In this phase, the deep learning model is trained after the log collection to aid in understanding the patterns of bandwidth consumption and in finding anomalies in bandwidth usage using CNN. All network traffic in Lab 4 is captured in ether 1 connection for Tx and Rx, as illustrated in Figure 4.



```
     Name                      Type         Actual MTU  L2 MTU  Tx          Rx          Tx Packet (p/s)  Rx Pa
   ;;; sg-15.hostddns.us:8242<->8291
     azrieluts@myt...          OVPN Client                      0 bps       0 bps       0
   ;;; ISP Provider UTS
R    ether1                    Ethernet     1500        1592    39.3 Mbps   60.0 Mbps   5 300
     ether2                    Ethernet     1500        1592    0 bps       0 bps       0
   ;;; PC-1 Remote
     ether3                    Ethernet     1500        1592    0 bps       0 bps       0
     ether4                    Ethernet     1500        1592    0 bps       0 bps       0
     ether5                    Ethernet     1500        1592    0 bps       0 bps       0
   ;;; Access Point TP-Link
R    ether6                    Ethernet     1500        1592    60.8 Mbps   42.6 Mbps   7 197
   ;;; PC for Testing Cyber Security
R    ether7                    Ethernet     1500        1592    4.3 kbps    20.8 kbps   4
     ether8                    Ethernet     1500        1592    0 bps       0 bps       0
     ether9                    Ethernet     1500        1592    0 bps       0 bps       0
     ether10                   Ethernet     1500        1592    0 bps       0 bps       0
   ;;; CISCO-1
     ether11                   Ethernet     1500        1592    0 bps       0 bps       0
```

Figure 4. Interface ether 1 / main internet connection used for capturing the entire recorded log data.

Following the shift from data collection via Telegram and the mail server's .txt format for 20 consecutive days, we will now implement data classification using Python in conjunction with the CNN algorithm. The extracted and classified bandwidth data will further undergo evaluation to derive the metrics and performance indices for the network.

B. CNN Implementation

The CNN process is a continuation of the data that has been pre-processed in the backbone network in UTS Lab 4. It entails data preprocessing, cleansing, feature engineering, labelling, data normalisation, noise filtering, as well as the training of a deep learning model on the first day's dataset, followed by CNN classification and results presentation in the form of graphs as illustrated in the flow diagram in Figure 5.

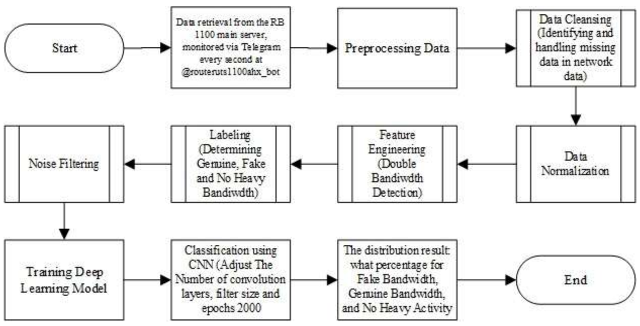

Start
Data retrieval from the RB 1100 main server, monitored via Telegram every second at @routeruts1100ahx_bot
Preprocessing Data
Data Cleansing (Identifying and handling missing data in network data)
Data Normalization
Feature Engineering (Double Bandiwdth Detection)
Labeling (Determining Genuine, Fake and No Heavy Bandiwdth)
Noise Filtering
Training Deep Learning Model
Classification using CNN (Adjust The Number of convolution layers, filter size and epochs 2000
The distribution result: what percentage for Fake Bandwidth, Genuine Bandwidth, and No Heavy Activity
End

Figure 5. CNN Bandwidth Classification Model

The data flow through the CNN pipeline is conducted in a structured manner, commencing with model training utilizing complex network logs gathered over five days, and ending with the classification of real-time data accumulated over twenty days. Firstly, network logs are collected second by second from Computer Lab 4 within the University of Technology Sarawak (UTS). During the model's training period, logs are gathered for five days, which represents a broader and more sophisticated range of data. The dataset includes all forms of network activities, including extraneous entries, thereby making it larger in size and challenging for the model to learn from. The data flow for CNN processing from log acquisition to training and real-time data classification can be seen in Figure 6 as follows.

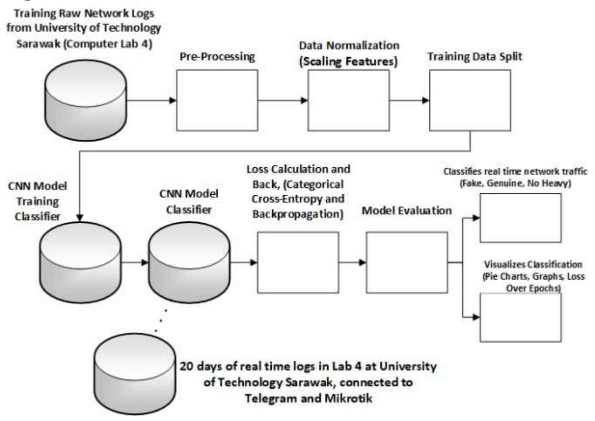

Training Raw Network Logs from University of Technology Sarawak (Computer Lab 4)
Pre-Processing
Data Normalization (Scaling Features)
Training Data Split
CNN Model Training Classifier
CNN Model Classifier
Loss Calculation and Back, (Categorical Cross-Entropy and Backpropagation)
Model Evaluation
Classifies real time network traffic (Fake, Genuine, No Heavy)
Visualizes Classification (Pie Charts, Graphs, Loss Over Epochs)
20 days of real time logs in Lab 4 at University of Technology Sarawak, connected to Telegram and Mikrotik

Figure 6. Data Processing Flow of the CNN Algorithm

The training logs are then subjected to a pre-processing step, in which the unwanted data is removed, missing values are addressed, and anomalies are corrected. Feature extraction is subsequently performed, in which data such as bandwidth values and timestamps are converted into numerical representations understandable by the CNN model. The log entries are then tagged as Fake Bandwidth, Genuine Bandwidth, or No Heavy Activity. This is followed by data normalization so that all of the features are scaled to a common range, typically between 0 and 1. The dataset is split into two datasets: training data and validation data. This splitting allows us to evaluate the performance of the model on unseen data. The CNN architecture is then trained using the given training dataset. It learns to identify patterns within the data through convolutional layers, pool layers, and fully connected or dense layers. The utilization of ReLU activation functions allows for the addition of non-linearity and thereby enhances the capabilities of the model to identify more intricate patterns.

During the training process, the model calculates the loss value, which is the difference between the predicted labels and the actual labels. The loss is calculated using the Categorical Cross-Entropy Loss function, and the model's internal weights are adjusted using the backpropagation algorithm to improve accuracy. At the end of the training process, the model is tested using the validation dataset. Several performance measures, such as Precision, Recall, and F1-Score, are used to assess the performance of the model in classifying entries into the three defined categories. It is utilized to categorize real-time data only after the model achieves good accuracy. Logs collected for twenty days from Computer Lab 4 at UTS are processed at this stage. This real-time dataset is cleaner and more precise as it contains only logs related to actual network activity, those corresponding to Telegram and Mikrotik. Then the model developed classifies every real-time log entry into one of the three different categories: Fake Bandwidth, Genuine Bandwidth, or No Heavy Activity. The results of the classification process are visualized using pie charts, trend graphs, and loss-over-epochs graphs generated during the training process, thus illustrating the learning progress of the model.

As the first step in the CNN section, information is collected from the RB 1100 server every second through the bot @routeruts1100data_bot on Telegram. The information collected constitutes network logs detailing users' bandwidth consumption. This data is processed as the main input in the analysis shown below.

```
def load_data(file):
    with open(file.name, "r", encoding="utf-8")
as f:
        lines = f.readlines()
    data, labels, bandwidth = [[] for _ in
range(3)]
    for line in lines:
       line = line.strip().lower()
        if "fake bandwidth" in line:
            labels.append("Fake")
bandwidth.append(int(re.search(r'(\d+)',
line).group()))
        elif "genuine bandwidth" in line:
            labels.append("Genuine")
bandwidth.append(int(re.search(r'(\d+)',
line).group()))
        elif "no heavy activity" in line:
            labels.append("No Heavy")
            bandwidth.append(0)
        data.append(line)
    return data, labels, bandwidth
```

Mathematically, the preprocessing operation can be written as (1)

$$X\_cleaned = clean\_text(X\_raw) \tag{1}$$

where X_raw is the provided data of the unprocessed log, and X_cleaned is the data after a text cleaning process that removes non-alphanumeric characters and applies text normalisation. Data obtained from network logs undergo a preprocessing procedure to ensure that the format is appropriate for the output of three criteria, which are fake, genuine, and no heavy; this process consists of text cleansing and normalisation, as well as formatting the data to make it ready to be processed as follows.

```
output = text.lower()
output = re.sub(r'[^a-zA-Z0-9\s]', '', output)
return output
```

Following the preprocessing, data cleansing was performed by detecting and resolving missing or invalid data. Next, normalisation was done to ensure that all data had a common uniform scale. Normalisation is the process of changing the scale of data so that features such as bandwidth speed can be equitably compared. The values for this normalisation were set by transforming all values to a range between 0 and 1, represented as follows (2)

$$X\_norm = (X - X\_min) / (X\_max - X\_min) \quad (2)$$

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
labels_train =
label_encoder.fit_transform(labels_train)
```

The next step is advanced Feature Engineering, in this case, Double Bandwidth Detection. At this stage, the aim is to identify the implementation of dual bandwidth due to networking anomalies or fraudulent activities, for example, at over 100 Mbps, as the model only considers up to 100 Mbps.

$$X\_double = \{ x \mid x > 100 \text{ Mbps} \} \quad (3)$$

Wherein X_double is filtered bandwidth data containing values greater than 100 Mbps, indicating the possibility of an anomaly, using the following script.

```
bandwidth_values = [int(re.search(r'(\d+)',
line).group()) for line in data if
re.search(r'(\d+)', line)]
double_bandwidth_detected = [bw for bw in
bandwidth_values if bw > 100]
```

Data that has been processed will be labelled to distinguish genuine bandwidth, fake bandwidth, and no heavy activity. When data demonstrates an upload speed from 21.22 to 100 Mbps, the data will be labelled as genuine bandwidth for both upload and download. Meanwhile, if the data shows an upload speed of less than 20 Mbps, the data will be labelled as fake bandwidth. Data without heavy activity will be labelled as no heavy activity.

```
if "fake bandwidth" in line:
    labels.append("Fake")
elif "genuine bandwidth" in line:
    labels.append("Genuine")
elif "no heavy activity" in line:
    labels.append("No Heavy")
```

Based on the above flowchart, the next stage involves noise filtering to select irrelevant data or errors present in the log file, such as uncontrolled logon changes or bandwidth information augmenting that are outside the scope of fake, genuine, or no heavy, or deleting the so-called short strings which are devoid of meaning.

```
filtered_data = [line for line in data if
len(line) > 10]
```

The main part, that is training data, is done with some convolution and pooling layers to capture patterns in the log data of the network. Parameter models such as the number of convolutional layers, filter size, and epoch were implemented on the output. The model was then trained with processed data, while the results were evaluated to measure the accuracy of classification for fake heavy bandwidth, genuine heavy, and no heavy bandwidth, which was over five hundred thousand rows of data. The batch size utilised in this experiment was 32, which indicates that, in every iteration of the training, 32 samples of data will be considered for gradient calculation and weight update. The core of the CNN model includes several convolutional layers and pooling layers that operate on raw input data to learn spatial features. Mathematically, the convolution operation can be expressed as follows:

$$C(i,j) = \text{sum\_m sum\_n } X(i+m, j+n) * W(m,n) \quad (4)$$

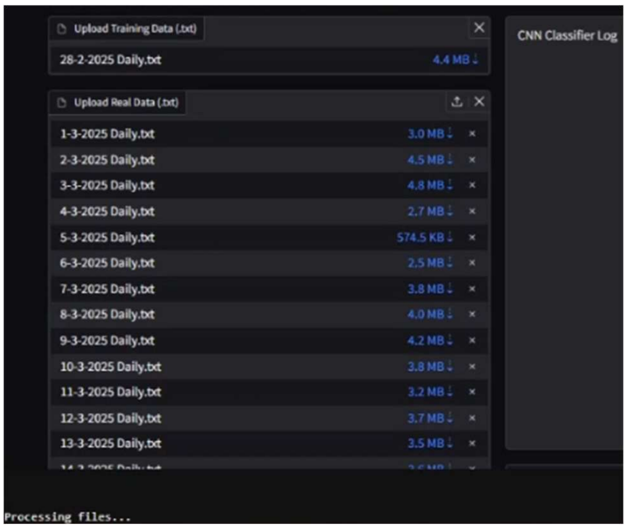
Where X indicates the input data (log data or bandwidth value), W is the convolutional filter, and C is the resultant feature map. Filter W learns the spatial patterns in the data using local analysis windows, which are then followed by pooling operations to decrease the dimensions of the feature map and capture the characteristics by the encoding model as follows:

```
model = Sequential([
     Embedding(num_words, embedding_dim,
input_length=input_length),
     Conv1D(256, 3, activation='relu'),
     MaxPooling1D(3),
     Conv1D(128, 3, activation='relu'),
     MaxPooling1D(3),
     Conv1D(64, 3, activation='relu'),
     MaxPooling1D(3),
     GlobalMaxPooling1D(),
     Dense(128, activation='relu'),
     Dropout(0.5),
     Dense(num_classes, activation='softmax')
 ])
 model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
 return model
```

After the CNN model has been trained with the initial data, model evaluation is carried out by testing the data to see the accuracy obtained from outside. The classification results are then visualised in a pie chart depicting the distribution of categories.

```
category_distribution =
"category_distribution.png"
plt.figure(figsize=(8, 6))
labels = ['Fake Bandwidth', 'Genuine Bandwidth',
'No Heavy Activity']
sizes = [fake_detected, genuine_detected,
no_heavy_detected]
def func(pct, allvalues):
    absolute = int(np.round(pct / 100.*
np.sum(allvalues)))
return f"{pct:.1f}%\n({absolute:d})"
plt.pie(sizes, labels=labels, autopct=lambda
pct: func(pct, sizes), startangle=140)
```

The output of the main step that occurs in this application is the uploading of data or files. The user can upload network log files in the form of Daily.txt, which contains records of bandwidth speed activity over 24 hours at three-second intervals. The data in the first capture was trained in the first 20 days; later, this training was used as real-time data, processed daily, and updated automatically from the mail server and monitoring Telegram, as shown in Figure 6 model of the upload dataset.



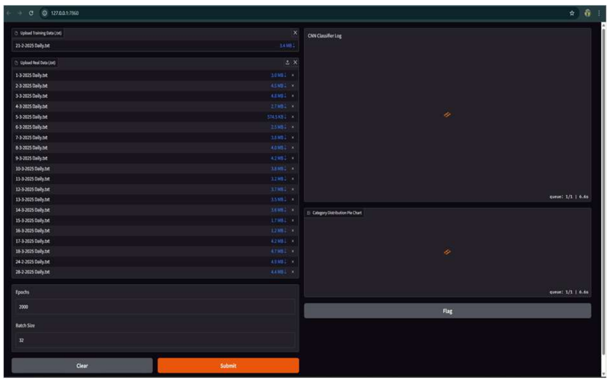
28-2-2025 Daily.txt 4.4 MB

1-3-2025 Daily.txt 3.0 MB
2-3-2025 Daily.txt 4.5 MB
3-3-2025 Daily.txt 4.8 MB
4-3-2025 Daily.txt 2.7 MB
5-3-2025 Daily.txt 574.5 KB
6-3-2025 Daily.txt 2.5 MB
7-3-2025 Daily.txt 3.8 MB
8-3-2025 Daily.txt 4.0 MB
9-3-2025 Daily.txt 4.2 MB
10-3-2025 Daily.txt 3.8 MB
11-3-2025 Daily.txt 3.2 MB
12-3-2025 Daily.txt 3.7 MB
13-3-2025 Daily.txt 3.5 MB

```
Processing files...
```

Figure 6. Dataset Upload Model

In the deep learning model training process, this application issues two parameters subject to manual configuration as outputs: epoch and batch size. An epoch, also referred to as an iteration, is a complete forward and backward pass of all the training data through the model. In this context, the maximum number of epochs is set to 2000, which is sufficient to enable the model to deeply learn the patterns within the bandwidth data. Batch size is referred to as the number of records that are processed in a single cycle of weight update. In this case, a batch size of 32 was used, meaning that 32 samples were processed every time the model performed an update, as illustrated in Figure 7.



21-2-2025 Daily.txt 3.4 MB

1-3-2025 Daily.txt 3.0 MB
2-3-2025 Daily.txt 4.5 MB
3-3-2025 Daily.txt 4.8 MB
4-3-2025 Daily.txt 2.7 MB
5-3-2025 Daily.txt 574.5 KB
6-3-2025 Daily.txt 2.5 MB
7-3-2025 Daily.txt 3.8 MB
8-3-2025 Daily.txt 4.0 MB
9-3-2025 Daily.txt 4.2 MB
10-3-2025 Daily.txt 3.8 MB
11-3-2025 Daily.txt 3.2 MB
12-3-2025 Daily.txt 3.7 MB
13-3-2025 Daily.txt 3.5 MB
14-3-2025 Daily.txt 3.6 MB
15-3-2025 Daily.txt 1.7 MB
16-3-2025 Daily.txt 1.2 MB
17-3-2025 Daily.txt 4.2 MB
18-3-2025 Daily.txt 4.7 MB
24-2-2025 Daily.txt 4.9 MB
28-2-2025 Daily.txt 4.4 MB

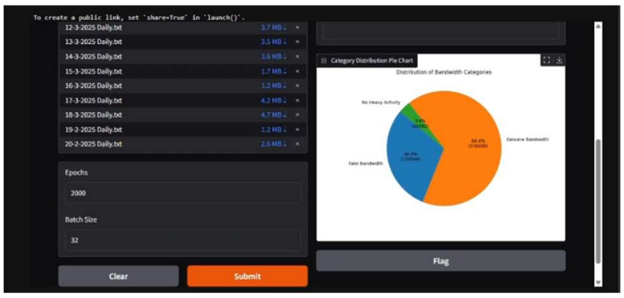
Epochs
2000

Batch Size
32

queue: 1/1 | 6.6s

Figure 7. Python based CNN Log Classifier Model

During training, the CNN model performed a deep analysis of the incoming log data. Each epoch provided new learning for the model to be able to identify features from the three categories of bandwidths. Genuine Bandwidth pertains to network activities that display legitimate and real bandwidth usage. Fake Bandwidth, on the other hand, refers to the activity in which there is less than 15% of the allowed 100 Mbps bandwidth utilisation (in which case 200 Mbps is used but there is a tolerance limit of 20 Mbps) to show a high level of utilisation without any actual data traffic for 24 hours a day for every second. Moreover, the No Heavy Activity evidence shows having little or no significant bandwidth activity. The model was first trained, and the next step was to evaluate the model using the test data that was set aside. The model was tested for the degree of accuracy on how it classifies new data from the set of three predetermined categories, as shown in Figure 8.


Category Distribution Pie Chart
Distribution of Bandwidth Categories
No Heavy Activity
Genuine Bandwidth
Fake Bandwidth
Flag
Clear
Submit

```
To create a public link, set `share=True` in `launch()`.
```

12-3-2025 Daily.txt 3.7 MB
13-3-2025 Daily.txt 3.5 MB
14-3-2025 Daily.txt 3.6 MB
15-3-2025 Daily.txt 1.7 MB
16-3-2025 Daily.txt 1.2 MB
17-3-2025 Daily.txt 4.2 MB
18-3-2025 Daily.txt 4.7 MB
19-2-2025 Daily.txt 1.2 MB
20-2-2025 Daily.txt 2.6 MB

Epochs
2000

Batch Size
32

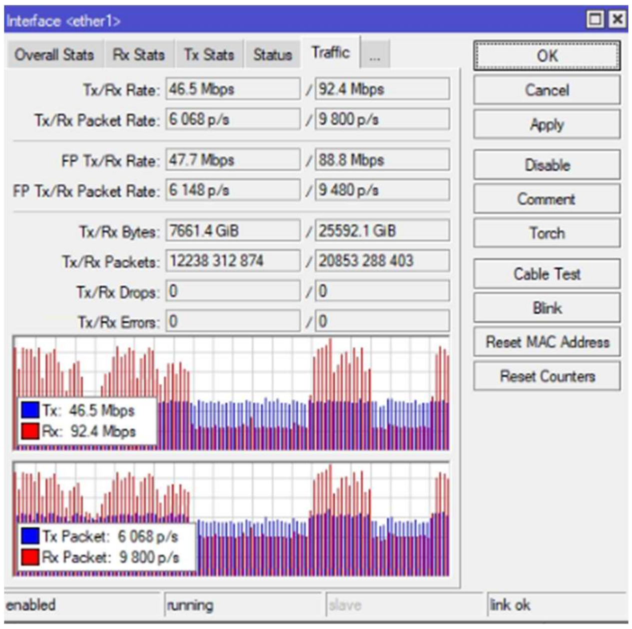
Figure 8. Python-based CNN Classification Results Model

III. RESULTS AND DISCUSSIONS

A. CNN Model Evaluation Results

In this study, the CNN architecture was used to classify network logs into Fake Bandwidth, Genuine Bandwidth, and No Heavy Activity. The results were obtained from the bandwidth logs that had been documented from the traffic graphs into data logs as in the Figures 9 and 10 which are the results of monitoring and have consumed the bandwidth resources until downloading 25 Terabytes and uploading 7 Terabytes for 20 days of bypass bandwidth monitoring which was later converted to monitoring data in seconds.

Nurcahyo, A. C., Yong , T. H., & Atanda, A. F. (2025). Optimisation of Network Logs for Fake Bandwidth Classification using CNN. TEPIAN, 6(2). https://doi.org/10.51967/tepian.v6i2.3260



Tx/Rx Rate:	46.5 Mbps	92.4 Mbps
Tx/Rx Packet Rate:	6 068 p/s	9 800 p/s
FP Tx/Rx Rate:	47.7 Mbps	88.8 Mbps
FP Tx/Rx Packet Rate:	6 148 p/s	9 480 p/s
Tx/Rx Bytes:	7661.4 GiB	25592.1 GiB
Tx/Rx Packets:	12238 312 874	20853 288 403
Tx/Rx Drops:	0	0
Tx/Rx Errors:	0	0

Tx: 46.5 Mbps
Rx: 92.4 Mbps
Tx Packet: 6 068 p/s
Rx Packet: 9 800 p/s

enabled | running | slave | link ok

Figure 9. Network Traffic Model

The data depicted in Figures 8 and 9 was converted into a .txt format and sent from the mail server phdscm01@uts.edu.my over port 587, which is utilized for the transmission of emails encrypted using Secure SMTP (SMTPS). The information is, on average, 3-6 MB per file per day. It is to be uploaded as real-world data that has been classified using CNN and needs to be processed.

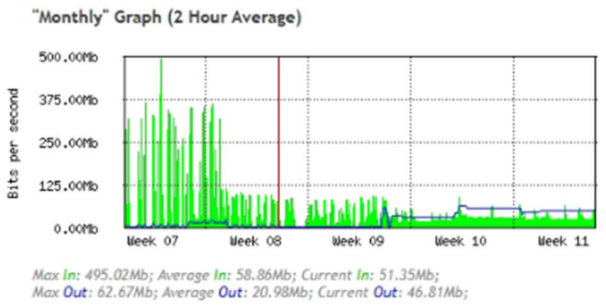


Figure 10. Monitoring Results for 20 Days from RB 1100 AHx in Lab 4 UTS

CNN was used to classify internet logs of Fake Bandwidth, Genuine Bandwidth, and No Heavy Activity. For the evaluation, Precision, Recall, F1-Score, and Support were calculated for each category. The use of these metrics is crucial to assess the effectiveness of the model in log data pattern classification. After training the model with 2000 epochs, the evaluation results are presented in Table 1.

Table 1. Precision, Recall, and F1-Score Test Results

Type	Precision	Recall	F1-Score	Support
Fake Bandwidth	0.89	0.90	0.89	170544
Genuine Bandwidth	0.92	0.95	0.93	378289
No Heavy Activity	0.98	0.84	0.91	19382

Based on the table above, we observe that Genuine Bandwidth possesses very high Precision and Recall (0.92 and 0.95), which suggests that this model is highly accurate in distinguishing valid network activities. On the other hand, while Fake Bandwidth had a lower Precision (0.89), the higher Recall (0.90) suggests that the model is very aggressive in detecting fake bandwidths. No Heavy Activity has astonishingly high Precision (0.98) with lower Recall (0.84), which implies that though this category is detected accurately, the model does not capture all pertinent entries. This is corroborated by the pie chart results in Figure 11, as obtained from the application.

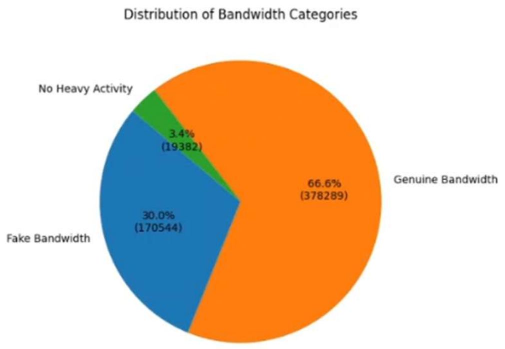


Figure 11. Bandwidth Distribution Model from CNN Classification at Lab 4 UTS

The evaluation was carried out using the metrics of Precision, Recall, F1-Score, and Support for each category. The importance of these metrics is vital when it comes to assessing the effectiveness of a model in recognizing patterns in the log data.

B. Loss Over Epochs

On the external analysis of this application, calculations were made for the Loss Over Epochs graph that illustrates the loss value during training over 2,000 epochs, which took 7 hours of processing. This evidence illustrates the level of prediction error made by the model in relation to the actual prediction results versus the real labels. Categorical cross-entropy was used to calculate loss, and its mathematical expression is (4)

$$\text{Loss} = -\Sigma\, y\_i \log(\hat{y}\_i) \qquad (4)$$

Where y_i is the target value for class i, and ŷ_i is the predicted probability of the model for that class. From the training results displayed in Figure 12, it is evident that the loss value decreases sharply, which indicates that the model learns these things very quickly because the log data was already distributed properly according to the required criteria, although it necessitated the router and server to perform at a high resource cost.

Figure 12 illustrates that CNN classified bandwidths for the first time during training. The loss was approximately equal to 1.0 at epoch zero, then sharply decreased to 0.5 by epoch 100, indicating that the model was beginning to assimilate data swiftly. Subsequently, from epochs 100 to 500, the loss reduction rate slowed down progressively, hitting approximately 0.2, which indicates the model was nearing the convergence point. Within the region of epochs 500 to 1000, the loss continued to decrease and reached around 0.1, which indicated that the model was becoming more effective in reducing the prediction errors.

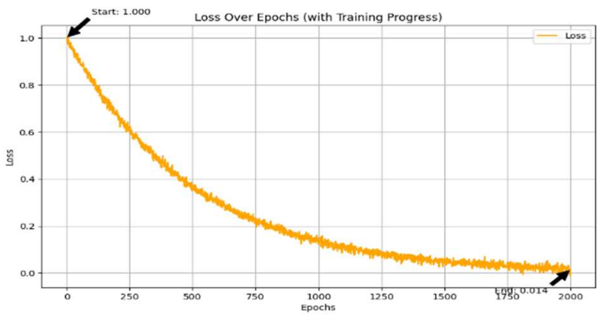
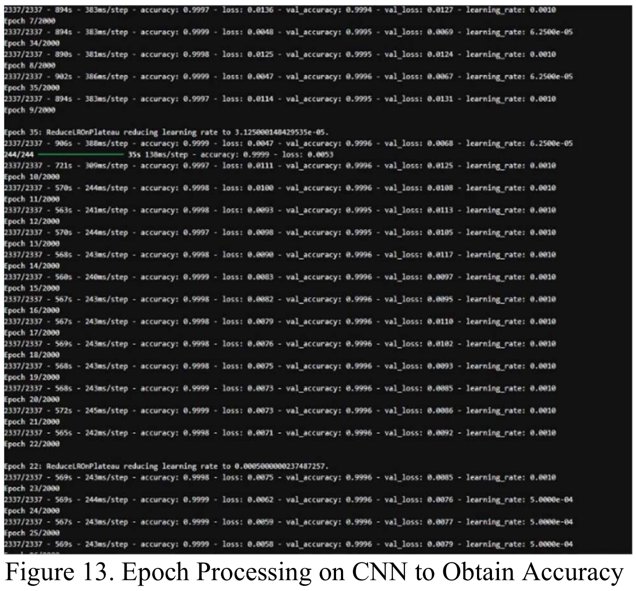


Figure 12. Loss Over Epochs of the CNN Classification Results

At the 2000th epoch, which is the end of the training period, the loss value was around 0.05, which is incredibly low and suggests that the model achieved optimal convergence, where the prediction error is nearly zero. The use of epochs shows that the robustness of the model can be increased by using techniques where the learning rate can be dropped at certain points during training. Figure 13 shows the CNNs processing in this study.



```
2337/2337 - 894s - 383ms/step - accuracy: 0.9997 - loss: 0.0136 - val_accuracy: 0.9994 - val_loss: 0.0127 - learning_rate: 0.0010
Epoch 7/2000
2337/2337 - 894s - 383ms/step - accuracy: 0.9999 - loss: 0.0048 - val_accuracy: 0.9995 - val_loss: 0.0069 - learning_rate: 6.2500e-05
Epoch 34/2000
2337/2337 - 890s - 381ms/step - accuracy: 0.9998 - loss: 0.0125 - val_accuracy: 0.9995 - val_loss: 0.0124 - learning_rate: 0.0010
Epoch 8/2000
2337/2337 - 902s - 386ms/step - accuracy: 0.9999 - loss: 0.0047 - val_accuracy: 0.9996 - val_loss: 0.0067 - learning_rate: 6.2500e-05
Epoch 35/2000
2337/2337 - 894s - 383ms/step - accuracy: 0.9997 - loss: 0.0114 - val_accuracy: 0.9995 - val_loss: 0.0131 - learning_rate: 0.0010
Epoch 9/2000

Epoch 35: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
2337/2337 - 906s - 388ms/step - accuracy: 0.9999 - loss: 0.0047 - val_accuracy: 0.9996 - val_loss: 0.0068 - learning_rate: 6.2500e-05
244/244 ━━━━━━━━━━━━━━━━━━━━ 35s 138ms/step - accuracy: 0.9999 - loss: 0.0053
2337/2337 - 721s - 309ms/step - accuracy: 0.9997 - loss: 0.0111 - val_accuracy: 0.9996 - val_loss: 0.0125 - learning_rate: 0.0010
Epoch 10/2000
2337/2337 - 570s - 244ms/step - accuracy: 0.9998 - loss: 0.0100 - val_accuracy: 0.9996 - val_loss: 0.0108 - learning_rate: 0.0010
Epoch 11/2000
2337/2337 - 563s - 241ms/step - accuracy: 0.9998 - loss: 0.0093 - val_accuracy: 0.9995 - val_loss: 0.0113 - learning_rate: 0.0010
Epoch 12/2000
2337/2337 - 570s - 244ms/step - accuracy: 0.9997 - loss: 0.0098 - val_accuracy: 0.9995 - val_loss: 0.0105 - learning_rate: 0.0010
Epoch 13/2000
2337/2337 - 568s - 243ms/step - accuracy: 0.9998 - loss: 0.0090 - val_accuracy: 0.9996 - val_loss: 0.0117 - learning_rate: 0.0010
Epoch 14/2000
2337/2337 - 560s - 240ms/step - accuracy: 0.9999 - loss: 0.0083 - val_accuracy: 0.9996 - val_loss: 0.0097 - learning_rate: 0.0010
Epoch 15/2000
2337/2337 - 567s - 243ms/step - accuracy: 0.9998 - loss: 0.0082 - val_accuracy: 0.9996 - val_loss: 0.0095 - learning_rate: 0.0010
Epoch 16/2000
2337/2337 - 567s - 243ms/step - accuracy: 0.9998 - loss: 0.0079 - val_accuracy: 0.9996 - val_loss: 0.0110 - learning_rate: 0.0010
Epoch 17/2000
2337/2337 - 569s - 243ms/step - accuracy: 0.9998 - loss: 0.0076 - val_accuracy: 0.9996 - val_loss: 0.0102 - learning_rate: 0.0010
Epoch 18/2000
2337/2337 - 568s - 243ms/step - accuracy: 0.9998 - loss: 0.0075 - val_accuracy: 0.9996 - val_loss: 0.0093 - learning_rate: 0.0010
Epoch 19/2000
2337/2337 - 568s - 243ms/step - accuracy: 0.9999 - loss: 0.0073 - val_accuracy: 0.9996 - val_loss: 0.0085 - learning_rate: 0.0010
Epoch 20/2000
2337/2337 - 572s - 245ms/step - accuracy: 0.9999 - loss: 0.0073 - val_accuracy: 0.9996 - val_loss: 0.0086 - learning_rate: 0.0010
Epoch 21/2000
2337/2337 - 565s - 242ms/step - accuracy: 0.9998 - loss: 0.0071 - val_accuracy: 0.9996 - val_loss: 0.0092 - learning_rate: 0.0010
Epoch 22/2000

Epoch 22: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
2337/2337 - 569s - 243ms/step - accuracy: 0.9998 - loss: 0.0075 - val_accuracy: 0.9996 - val_loss: 0.0085 - learning_rate: 0.0010
Epoch 23/2000
2337/2337 - 569s - 244ms/step - accuracy: 0.9999 - loss: 0.0062 - val_accuracy: 0.9996 - val_loss: 0.0076 - learning_rate: 5.0000e-04
Epoch 24/2000
2337/2337 - 567s - 243ms/step - accuracy: 0.9999 - loss: 0.0059 - val_accuracy: 0.9996 - val_loss: 0.0077 - learning_rate: 5.0000e-04
Epoch 25/2000
2337/2337 - 569s - 243ms/step - accuracy: 0.9999 - loss: 0.0058 - val_accuracy: 0.9996 - val_loss: 0.0079 - learning_rate: 5.0000e-04
```

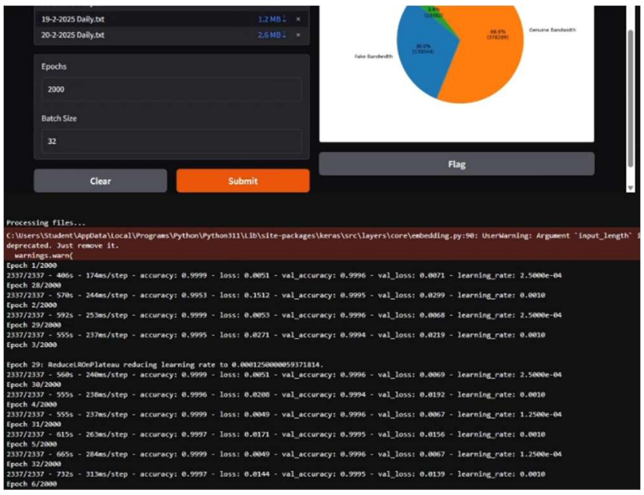
Figure 13. Epoch Processing on CNN to Obtain Accuracy and Loss Over Epochs

C. *Results of the Novelty of The Research (Network Log Optimisation)*

The novelty of the results in this research refers to the consideration of using network logs captured in real-time over 20 continuous days, with data collection frequency every second, to uncover fake bandwidth or more than 500,000 lines of data used to train the CNN model. With the presence of real-time network logs, there is room for a more dynamic and accurate approach in monitoring the utilization of large-scale computer networks using log monitoring and CNN-based classification. The success in classifying hundreds of thousands of lines of logs demonstrates CNN's extraordinary advantages in operating on big data and ability to capture very subtle phenomena in complex data that as shown:

Detected Fake Bandwidth	: 170544 (30.01%)
Detected Genuine Bandwidth	: 378289 (66.57%)
Detected No Heavy Activity	: 19382 (3.41%)

The log analysis approach can overcome the difficulties posed by traditional techniques that rely on larger time frames or sparse data. The results from this CNN model show a high classification accuracy, even when working with a large amount of data in real time. This model validated the notion that CNNs can be modified and enhanced for the application of everyday real-time data, which requires instant classification along with a very high accuracy level, as evidenced in Figure 14.



19-2-2025 Daily.txt 1.2 MB
20-2-2025 Daily.txt 2.6 MB

Epochs
2000

Batch Size
32

```
Processing files...
C:\Users\Student\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument `input_length` i
deprecated. Just remove it.
  warnings.warn(
Epoch 1/2000
2337/2337 - 406s - 174ms/step - accuracy: 0.9999 - loss: 0.0051 - val_accuracy: 0.9996 - val_loss: 0.0071 - learning_rate: 2.5000e-04
Epoch 28/2000
2337/2337 - 570s - 244ms/step - accuracy: 0.9953 - loss: 0.1512 - val_accuracy: 0.9995 - val_loss: 0.0299 - learning_rate: 0.0010
Epoch 2/2000
2337/2337 - 592s - 253ms/step - accuracy: 0.9999 - loss: 0.0053 - val_accuracy: 0.9996 - val_loss: 0.0068 - learning_rate: 2.5000e-04
Epoch 29/2000
2337/2337 - 555s - 237ms/step - accuracy: 0.9995 - loss: 0.0271 - val_accuracy: 0.9994 - val_loss: 0.0219 - learning_rate: 0.0010
Epoch 3/2000

Epoch 29: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
2337/2337 - 560s - 240ms/step - accuracy: 0.9999 - loss: 0.0051 - val_accuracy: 0.9996 - val_loss: 0.0069 - learning_rate: 2.5000e-04
Epoch 30/2000
2337/2337 - 555s - 238ms/step - accuracy: 0.9996 - loss: 0.0208 - val_accuracy: 0.9994 - val_loss: 0.0192 - learning_rate: 0.0010
Epoch 4/2000
2337/2337 - 555s - 237ms/step - accuracy: 0.9999 - loss: 0.0049 - val_accuracy: 0.9996 - val_loss: 0.0067 - learning_rate: 1.2500e-04
Epoch 31/2000
2337/2337 - 615s - 263ms/step - accuracy: 0.9997 - loss: 0.0171 - val_accuracy: 0.9995 - val_loss: 0.0156 - learning_rate: 0.0010
Epoch 5/2000
2337/2337 - 665s - 284ms/step - accuracy: 0.9999 - loss: 0.0049 - val_accuracy: 0.9996 - val_loss: 0.0067 - learning_rate: 1.2500e-04
Epoch 32/2000
2337/2337 - 732s - 313ms/step - accuracy: 0.9997 - loss: 0.0144 - val_accuracy: 0.9995 - val_loss: 0.0139 - learning_rate: 0.0010
Epoch 6/2000
```

Figure 14. CNN Application Model for Lab 4 UTS

IV. CONCLUSION

This study demonstrated that the CNN model can be used to classify network logs to detect fake bandwidth, genuine bandwidth, and no heavy activity with high accuracy using log data collected in real-time every second for 20 days. This CNN model has proven capable of processing over 500,000 lines of log data at once, demonstrating the model's ability to recognise very complex and subtle patterns in large volumes of data that traditional methods struggle with. CNN successfully classified genuine bandwidth with high accuracy, evidenced by model performance evaluation based on metrics of: Precision 0.92, Recall 0.95, fake bandwidth detected with Precision 0.89 and Recall 0.90, and no heavy activity detected with Precision 0.98 and Recall 0.84. The Loss Over Epochs results show that as the number of epochs increased, the CNN model experienced a decrease in loss from 1.0 at epoch 0 to 0.05 at epoch 2000. This estimate revealed that the CNN model successfully minimised prediction errors with a stable training process that approached an optimal convergence point indicated by the decrease in loss. As part of the key innovation in this research, the optimisation of real-time network log collection and processing using CNN has proven effective on dynamically distributed data. Because the data is gathered systematically every second, the CNN model can classify it with great accuracy, even at a very large scale. This approach not only improves the accuracy of detecting imprecise manipulative fake bandwidth, but also greatly aids fake bandwidth detection and network monitoring optimisation in real-time responsive environments.

REFERENCES

Abdfilminaam, D., Alfarouk, S., Fouad, K., Slait, R., & Wasfy, R. (2024). Optimizing Brain Tumor Detection: Enhancing Diagnostic Accuracy in Brain Tumor Detection Using A Hybrid Approach of Machine Learning and Deep Learning Models. *2024 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, 1-8. https://doi.org/10.1109/MIUCC62295.2024.10783526.

Alhammadi, O., & Abul, O. (2024). Real-time Web Server Log Processing with Big Data Technologies. *2024 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 1-8. https://doi.org/10.1109/ASYU62119.2024.10757033.

Alrubayyi, H., Goteng, G., & Jaber, M. (2023). AIS for Malware Detection in a Realistic IoT System: Challenges and Opportunities. *Network*, 3, 522-537. https://doi.org/10.3390/network3040023.

Antonius, F., Sekhar, J., Rao, V., Pradhan, R., Narendran, S., Borda, R., & Silvera-Arcos, S. (2023). Unleashing the power of Bat optimized CNN-BiLSTM model for advanced network anomaly detection: Enhancing security and performance in IoT environments. *Alexandria Engineering Journal*. https://doi.org/10.1016/j.aej.2023.11.015.

AsSadhan, B., AlShaalan, R., Diab, D., Alzoghaiby, A., Alshebeili, S., Al-Muhtadi, J., Bin-Abbas, H., & El-Samie, F. (2020). A robust anomaly detection method using a constant false alarm rate approach. *Multimedia Tools and Applications*, 79, 12727 - 12750. https://doi.org/10.1007/s11042-020-08653-8.

Biswal, P., & Karekar, P. (2024). An Analytic Study of The Relationship Between Internet Connectivity and Productivity in The Workplace. *Journal of Informatics Education and Research*. https://doi.org/10.52783/jier.v4i1.635.

Cermák, M., Fritzová, T., Rusňák, V., & Sramkova, D. (2023). Using relational graphs for exploratory analysis of network traffic data. *Forensic Science International: Digital Investigation*. https://doi.org/10.1016/j.fsidi.2023.301563.

Collier-Brown, D. (2024). You Don't Know Jack about Bandwidth: If you're an ISP and all your customers hate you, take heart. This is now a solvable problem. *Commun. ACM*, 67, 38-41. https://doi.org/10.1145/3674953.

Dolgui, A., & Ivanov, D. (2021). 5G in digital supply chain and operations management: fostering flexibility, end-to-end connectivity and real-time visibility through internet-of-everything. *International Journal of Production Research*, 60, 442 - 451. https://doi.org/10.1080/00207543.2021.2002969.

Dong, S., Xia, Y., & Peng, T. (2021). Network Abnormal Traffic Detection Model Based on Semi-Supervised Deep Reinforcement Learning. *IEEE Transactions on Network and Service Management*, 18, 4197-4212. https://doi.org/10.1109/tnsm.2021.3120804.

Duan, X., Fu, Y., & Wang, K. (2022). Network traffic anomaly detection method based on multi-scale residual classifier. *Comput. Commun.*, 198, 206-216. https://doi.org/10.1016/j.comcom.2022.10.024.

Fadlil, A., Umar, R., Sunardi, .., & Nugroho, A. (2022). Comparison of Machine Learning Approach for Waste Bottle Classification. *Emerging Science Journal*. https://doi.org/10.28991/esj-2022-06-05-011.

Fotiadou, K., Velivasaki, T., Voulkidis, A., Skias, D., Tsekeridou, S., & Zahariadis, T. (2021). Network Traffic Anomaly Detection via Deep Learning. *Inf.*, 12, 215. https://doi.org/10.3390/info12050215.

Giordano, G., Palomba, F., & Ferrucci, F. (2022). On the use of artificial intelligence to deal with privacy in IoT systems: A systematic literature review. *J. Syst. Softw.*, 193, 111475. https://doi.org/10.1016/j.jss.2022.111475.

Gomes, R., Bittencourt, L., Madeira, E., Cerqueira, E., & Gerla, M. (2016). A combined energy-bandwidth approach to allocate resilient virtual software defined networks. *J. Netw. Comput. Appl.*, 69, 98-106. https://doi.org/10.1016/j.jnca.2016.02.024.

Hapsari, L. (2022, July 12). *Kecepatan internet Indonesia paling lambat di Asia Tenggara, apa penyebabnya?* Kumparan. https://kumparan.com/listyanihapsari171/kecepatan-internet-indonesia-paling-lambat-di-asia-tenggara-apa-penyebabnya-1yRrujdWhgj

Jhaveri, R., Ramani, S., Srivastava, G., Gadekallu, T., & Aggarwal, V. (2021). Fault-Resilience for Bandwidth Management in Industrial Software-Defined Networks. *IEEE Transactions on Network Science and Engineering*, 8, 3129-3139. https://doi.org/10.1109/tnse.2021.3104499.

Komadina, A., Martinić, M., Groš, S., & Mihajlović, Ž. (2024). Comparing Threshold Selection Methods for Network Anomaly Detection. *IEEE Access*, 12, 124943-124973. https://doi.org/10.1109/ACCESS.2024.3452168.

Lyu, S., & Liu, J. (2021). Convolutional Recurrent Neural Networks for Text Classification. *J. Database Manag.*, 32, 65-82. https://doi.org/10.4018/jdm.2021100105.

Lv, D., Cheng, X., Zhang, J., Zhang, W., Zhao, W., & Xu, H. (2022). DDoS attack detection based on CNN and federated learning. *Proceedings of the 2021 Ninth International Conference on Advanced Cloud and Big Data (CBD)*, 236-241. https://doi.org/10.1109/CBD54617.2021.00048

Magnani, S., Risso, F., & Siracusa, D. (2022). A Control Plane Enabling Automated and Fully Adaptive Network Traffic Monitoring With eBPF. *IEEE Access*, 10, 90778-90791. https://doi.org/10.1109/ACCESS.2022.3202644.

Najar, A., & S, M. (2024). Cyber-Secure SDN: A CNN-Based Approach for Efficient Detection and Mitigation of DDoS attacks. *Comput. Secur.*, 139, 103716. https://doi.org/10.1016/j.cose.2024.103716.

Oji, C., Nwankokwo, O., & Adu, C. (2021). Development Of An Enhanced Bandwidth Control Platform For Effective Monitoring And Utilization In Corporate Networks. *International Journal of Scientific and Research Publications (IJSRP)*. https://doi.org/10.29322/ijsrp.11.08.2021.p11636.

Onietan, C., Martins, I., Owoseni, T., Omonedo, E., & Eze, C. (2023). A Preliminary Study on the Application of Hybrid Machine Learning Techniques in Network Intrusion Detection Systems. *2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG)*, 1, 1-7. https://doi.org/10.1109/SEB-SDG57117.2023.10124596.

Pham, H., Nguyen, V., Tran, N., & Nguyen, M. (2023). Log Analysis For Network Attack Detection Using Deep Learning Models. *Proceedings of the 12th International Symposium on Information and Communication Technology*. https://doi.org/10.1145/3628797.3628943.

Razian, M., Fathian, M., Bahsoon, R., Toosi, A., & Buyya, R. (2022). Service composition in dynamic environments: A systematic review and future directions. *J. Syst. Softw.*, 188, 111290. https://doi.org/10.1016/j.jss.2022.111290.

Ring, J., Van Oort, C., Durst, S., White, V., Near, J., & Skalka, C. (2021). Methods for Host-Based Intrusion Detection with Deep Learning. *Digital Threats: Research and Practice*. https://doi.org/10.1145/3461462.

Saha, S., Haque, A., & Sidebottom, G. (2022). An Empirical Study on Internet Traffic Prediction Using Statistical Rolling Model. *2022 International Wireless Communications and Mobile Computing (IWCMC)*, 1058-1063. https://doi.org/10.1109/IWCMC55113.2022.9825059

Sari, M., Ningki, C., Rosa, F., Novando, K., & Mukin, Y. (2023). Analysis of Bandwidth Management Quality of Internet Network Services at the Shanti Bhuana Institute. *Journal of Information Technology*. https://doi.org/10.46229/jifotech.v3i1.666.

Shi, J., Fu, K., Wang, J., Chen, Q., Zeng, D., & Guo, M. (2024). Adaptive QoS-Aware Microservice Deployment With Excessive Loads via Intra- and Inter-Datacenter Scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 35, 1565-1582. https://doi.org/10.1109/TPDS.2024.3425931.

Šabanović, K., Arendt, C., Fricke, S., Geis, M., Böcker, S., & Wietfeld, C. (2024). AI-Based Anomaly Detection for Industrial 5G Networks by Distributed SDR Measurements. *2024 IEEE International Symposium on Measurements & Networking (M&N)*, 1-5. https://doi.org/10.1109/MN60932.2024.10615402.

Wang, S., Balarezo, J., Kandeepan, S., Al-Hourani, A., Chavez, K., & Rubinstein, B. (2021). Machine Learning in Network Anomaly Detection: A Survey. *IEEE Access*, PP, 1-1. https://doi.org/10.1109/ACCESS.2021.3126834.

Wen, H., Yu, J., Pan, G., Chen, X., Zhang, S., & Xu, S. (2022). A Hybrid CNN-LSTM Architecture for High Accurate Edge-Assisted Bandwidth Prediction. *IEEE Wireless Communications Letters*, 11, 2640-2644. https://doi.org/10.1109/LWC.2022.3213017.

Ye, M., Member, I., Member, I., & Fellow, I. (2023). FlexDATE: Flexible and Disturbance-Aware Traffic Engineering With Reinforcement Learning in Software-Defined Networks. *IEEE/ACM Transactions on Networking*, 31, 1433-1448. https://doi.org/10.1109/TNET.2022.3217083.

Zhang, H., Zhang, L., Yang, Z., Lyu, Z., Yang, H., Zhang, C., Bobrovs, V., Ozoliņš, O., Pang, X., & Yu, X. (2024). Equivalent Photoconductive Time-Domain Sampling for Monitoring High-Speed Terahertz Communication Signals. *Journal of Lightwave Technology*, 42, 4476-4484. https://doi.org/10.1109/JLT.2024.3372382.

Zhang, Y., Liu, W., Kuok, K., & Cheong, N. (2024). Anteater: Advanced Persistent Threat Detection With Program Network Traffic Behavior. *IEEE Access*, 12, 8536-8551. https://doi.org/10.1109/ACCESS.2024.3349943.

Zhao, M., Gahrooei, M., & Ilbeigi, M. (2024). Change Detection in Partially Observed Large-Scale Traffic Network Data. *IEEE Transactions on Intelligent Transportation Systems*, 25, 18913-18924. https://doi.org/10.1109/TITS.2024.3440836.

Zhuang, G. (2024). Research on Large-scale Data Anomaly Detection based on Deep Learning. *2024 5th International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, 249-252. https://doi.org/10.1109/ISPDS62779.2024.10667570.