# ABSTRACT

In today's world of malicious intent and growing technology, the ease of hiding one's identity increasing by the day, making it harder for cybersecurity professionals to track cyber criminals. The most common tools used for obfuscating network activity are Tor browser and VPNs. This project aims to identify the misuse of these anonymity-enabling technologies. For this project, the CICDarknet2020 dataset, with around 158659 data points, was used to train the CNN model for classification of the network packets. Preprocessing of data includes methodologies like Feature Selection using Random Forest, normalization and label encoding, etc. After training the model was successfully able to categories network packets into 4 major categories, namely: Tor, VPN, Non-Tor, Non-VPN with an accuracy of 95.09%.

# TABLE OF CONTENTS

| DESCRIPTION | PAGE NUMBER |
|---|---|
| CERTIFICATE | i |
| DECLARATION | ii |
| ACKNOWLEDGEMENTS | iii |
| ABSTRACT | iv |
| LIST OF FIGURES | vi |
| LIST OF TABLES | vii |
| ABBREVIATIONS | viii |
| 1. INTRODUCTION | 1 |
| 1.1. OBJECTIVES | 1 |
| 1.2. BACKGROUND & MOTIVATION | 1 |
| 1.3. DATASET DESCRIPTION | 2 |
| 2. METHODOLOGIES | 3 |
| 2.1. DATA PREPROCESSING | 3 |
| 2.2. MODEL DEVELOPMENT & TRAINING | 3 |
| 2.3. TOOLS & TECHNOLOGIES USED | 5 |
| 2.3.1. PROGRAMMING LANGUAGE | 5 |
| 2.3.2. LIBRARIES AND FRAMEWORKS | 5 |
| 2.3.3. ML TECHNIQUES | 5 |
| 3. TESTING & EVALUATION | 6 |
| 4. CONCLUSION | 7 |
| REFERENCES | 8 |

# LIST OF FIGURES

| S. No | Figure Title | Page No. |
|---|---|---|
| **1** | Dataset Details | 2 |
| **2** | Sequential Model Layers Representation | 3 |
| **3** | Loss Graph | 6 |
| **4** | Accuracy Graph | 6 |
| **5** | Confusion Matrix | 6 |

# LIST OF TABLES

| **S. No.** | **Table Name** | **Page No.** |
|---|---|---|
| **1** | CICDarknet 2020 Dataset composition | 2 |
| **2** | CICDarknet 2020 Dataset | 2 |
| **3** | Sequential Model Structure | 4 |

# ABBREVIATIONS

| Abbreviation | Full-Form |
|---|---|
| CNN | Convolutional Neural Network |
| VPN | Virtual Private Network |
| 2D-Array | 2-Dimensional Array |
| DPI | Deep Packet Inspection |
| VoIP | Voice over Internet Protocol |
| IQR | Interquartile Range |
| ML | Machine Learning |
| P2P | Peer-to-Peer |
| MSE | Mean Squared Error |
| TCN | Temporal Convolution Network |

# CHAPTER 1: INTRODUCTION

This project explores packet-level micro-behaviour classification using Convolutional Neural Networks (CNNs) to categorize network packets as Tor, VPN, or regular traffic. By representing each packet as a structured 2D array—where features such as byte values, headers, and payload sizes form an image-like input—the model leverages CNN architectures to capture local patterns in packet structures. This approach provides a robust framework for real-time traffic classification, aiding in network security monitoring, intrusion detection, and regulatory compliance. Through CNN-based feature extraction this project aims to improve the accuracy and efficiency of network traffic analysis.

## 1.1 Objectives

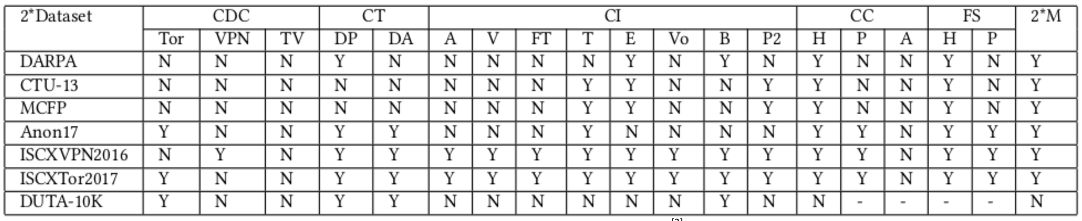
This project aims to classify network packets into 4 categories of Tor, Non-Tor, VPN or Non-VPN to aide in the detection of these technologies to prevent malicious use. While, the secondary objective of this project is to evaluate the effectiveness of a CNN model on a network packet's metadata.

## 1.2 Background & Motivation

With the rise of privacy-focused technologies such as Tor and VPNs, organizations face increasing challenges in monitoring and securing network traffic. While these tools provide essential anonymity for users, they can also be exploited for illicit activities such as cyberattacks, data exfiltration, and unauthorized access to restricted content. Traditional traffic classification methods, relying on port numbers or Deep Packet Inspection (DPI), struggle to differentiate between encrypted traffic types, making them less effective against modern evasion techniques. Deep learning, particularly Convolutional Neural Networks (CNNs), has demonstrated remarkable success in pattern recognition tasks, including image classification and anomaly detection. By treating packet-level network traffic as structured image-like inputs, CNNs can extract spatial dependencies and local patterns that conventional techniques fail to capture. This project is driven by the need for an intelligent, scalable, and automated approach to classify network packets as Tor, VPN, or regular traffic with high accuracy. The goal is to enhance cybersecurity frameworks by providing an advanced tool for detecting encrypted traffic behaviours while ensuring minimal reliance on manual feature engineering.

## 1.3 Dataset Description

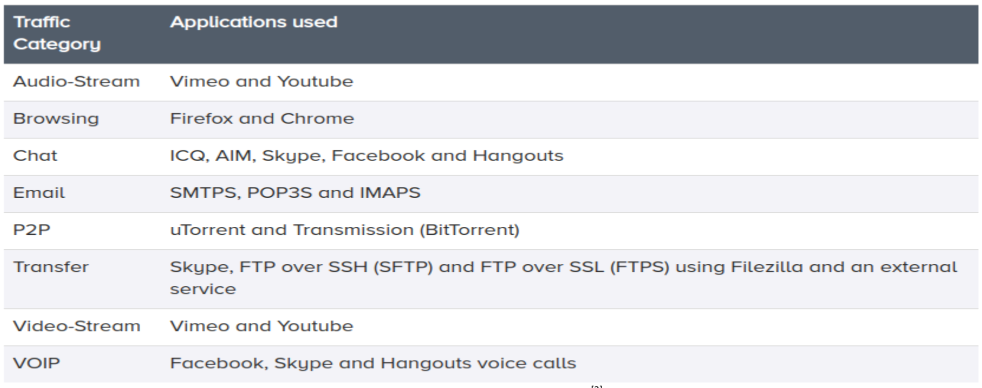
For this study, the CICDarknet2020 [1] data was used. In CICDarknet2020 dataset, a two-layered approach is used to generate benign and darknet traffic at the first layer.



| 2*Dataset | CDC | | | CT | | CI | | | | | | | | CC | | | FS | | 2*M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Tor | VPN | TV | DP | DA | A | V | FT | T | E | Vo | B | P2 | H | P | A | H | P | |
| DARPA | N | N | N | Y | N | N | N | N | N | Y | N | Y | N | Y | N | N | Y | N | Y |
| CTU-13 | N | N | N | N | N | N | N | N | Y | Y | N | N | Y | Y | N | N | Y | N | Y |
| MCFP | N | N | N | N | N | N | N | N | Y | Y | N | N | Y | Y | N | N | Y | N | Y |
| Anon17 | Y | N | N | Y | Y | N | N | N | Y | N | N | N | N | Y | Y | N | Y | Y | Y |
| ISCXVPN2016 | N | Y | N | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | N | Y | Y | Y |
| ISCXTor2017 | Y | N | N | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | N | Y | Y | Y |
| DUTA-10K | Y | N | N | Y | Y | N | N | N | N | N | N | Y | N | N | - | - | - | - | N |

Table 1. CICDarknet 2020 Dataset composition [2]

The dataset comprises of 24311 Darknet entries and 134348 Benign entries (total 158659 entries) split over different applications used as shown in Table 2.



| Traffic Category | Applications used |
|---|---|
| Audio-Stream | Vimeo and Youtube |
| Browsing | Firefox and Chrome |
| Chat | ICQ, AIM, Skype, Facebook and Hangouts |
| Email | SMTPS, POP3S and IMAPS |
| P2P | uTorrent and Transmission (BitTorrent) |
| Transfer | Skype, FTP over SSH (SFTP) and FTP over SSL (FTPS) using Filezilla and an external service |
| Video-Stream | Vimeo and Youtube |
| VOIP | Facebook, Skype and Hangouts voice calls |

Table 2. CICDarknet 2020 Dataset [2]

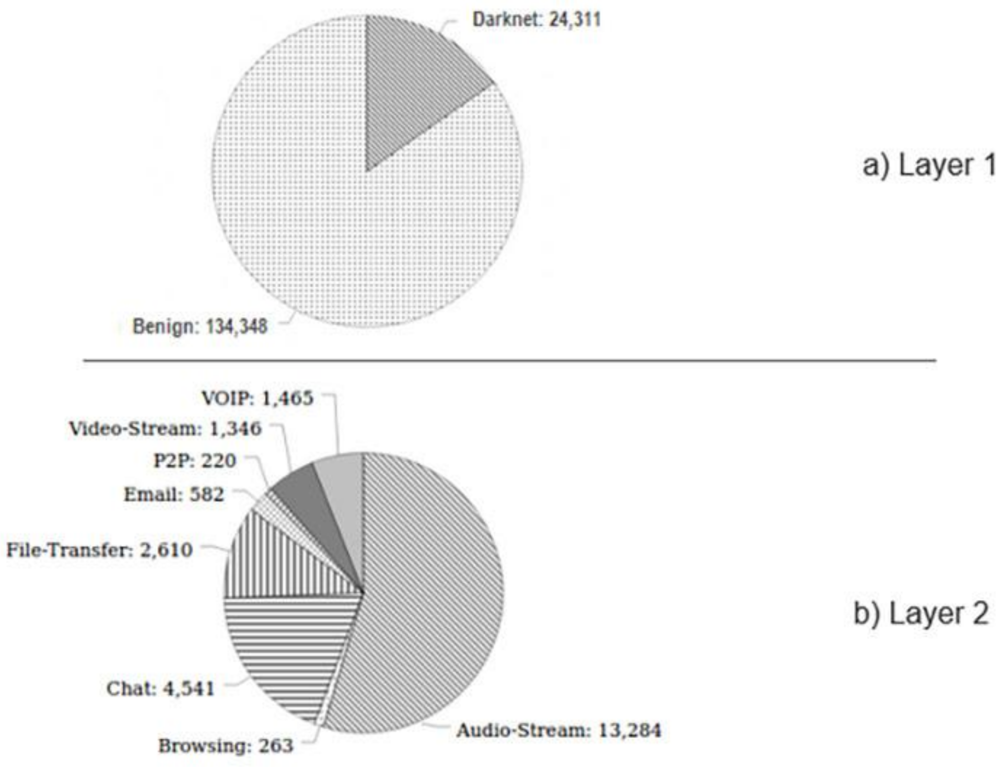


Fig 1. Dataset Details [4]

# CHAPTER 2: METHODOLOGIES

## 2.1. Data Preprocessing

**Data Cleaning.** Data cleaning is a crucial part of preparing data for a ML model training, it ensures that the data is consistent, not having missing values and contain only relevant information. Data cleaning consists of handling missing values, removing duplicate or irrelevant information and fix structural issues. After cleaning, we need to convert the data into a format suitable for the machine learning models. This comprises of activities like feature selection, scaling and normalization, feature encoding, etc. Raw network traffic contains many attributes, but a lot of them are considered moot in the process of anomaly detection, hence, we need to select a few characteristic features for training and testing. Similarly, Continuous numerical features (e.g., packet size, time intervals) are normalized to ensure all values are on a similar scale.

**Training and testing.** Out of the dataset, we require some data, to train the models, and some to test the accuracy of the models. A proper train-test split ensures the model generalizes well to unseen data.

## 2.2. Model Development & Training

**Architecture Design.** Generally, CNN are used to classify images, however, in our project, we use a CNN model because of its capabilities to learn hierarchical patterns from structured, sequential network data without manual feature engineering, so the data needs to be processed in a shape that can be accepted by the CNN model. The architecture consists of 8 layers, with 2 Convolution layers, 2 MaxPooling layer, 1 Flatten, 1 Dropout and 2 Dense layers.

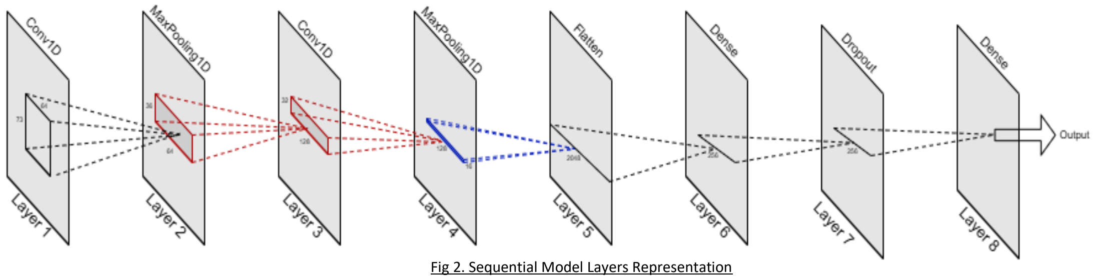
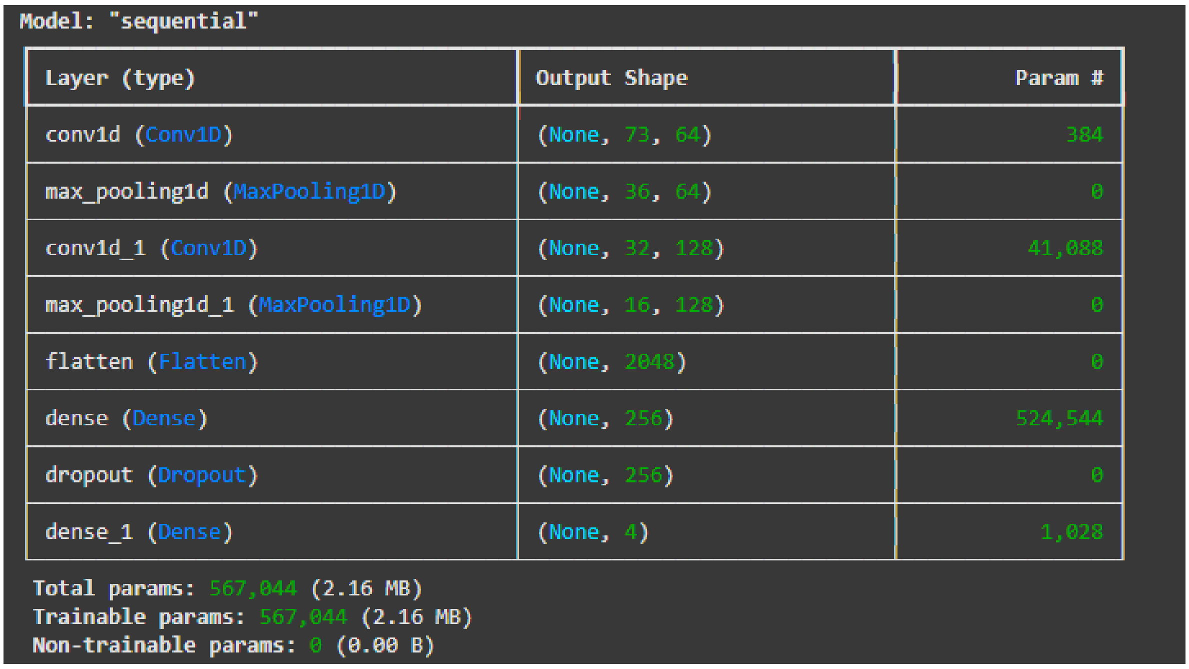


Fig 2. Sequential Model Layers Representation

Each layer carries a significance and role of its own, such as:

- **Conv1D:** Extracts spatial features from input metadata
- **MaxPooling1D:** Down samples data to reduce computation
- **Flatten:** Converts 3D output to 1D for dense layers
- **Dense:** Performs final classification
- **Dropout:** Prevents overfitting by randomly deactivating neurons during training



```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                        ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv1d (Conv1D)                     │ (None, 73, 64)              │             384 │
├─────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling1d (MaxPooling1D)        │ (None, 36, 64)              │               0 │
├─────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv1d_1 (Conv1D)                   │ (None, 32, 128)             │          41,088 │
├─────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling1d_1 (MaxPooling1D)      │ (None, 16, 128)             │               0 │
├─────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                   │ (None, 2048)                │               0 │
├─────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                       │ (None, 256)                 │         524,544 │
├─────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout (Dropout)                   │ (None, 256)                 │               0 │
├─────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                     │ (None, 4)                   │           1,028 │
└─────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 567,044 (2.16 MB)
 Trainable params: 567,044 (2.16 MB)
 Non-trainable params: 0 (0.00 B)
```

Table 3. Sequential Model Structure

**Mathematical Formulation.**

- **Input**: A sequence of traffic features $X = \{x_1, x_2, . . . , x_T\}$, where $x_t \in \mathbb{R}^d$ is a feature vector at time step *t*.
- **Model**: Convolutional Neural Network C with parameters $\theta$.
- **Output**: Predicted next step $x\hat{}_{C+1} = C(X; \theta)$.
- **Anomaly Detection**: Anomaly if $||x\hat{}_{C+1} - x_{C+1}||_2 > \delta$, where $\delta$ is a predefined threshold.
- **Loss Function**: Mean Squared Error (MSE)-

$$L(\hat{x}, x) = \frac{1}{C}\sum_{t=1}^{C} ||\hat{x}_t - x_t||_2{}^2$$

**Training and Performance Metrics.** The model was trained for 30 epochs using a batch size of 64 to ensure efficiency. With continuous evaluation of accuracies and loss, the final epoch iteration gave an accuracy of 95.09%.

## 2.3. Tools & Technology Used

### 2.3.1 Programming Language

**Python** language was chosen for the development of this CNN model due to its large selection of ML libraries and functionalities

### 2.3.2 Libraries & Frameworks

- **Pandas** [3]**, NumPy** [4] – Data handling and numerical operations
- **Scikit-learn** [5] – Preprocessing, feature selection, and splitting dataset
- **TensorFlow** [6] **/ Keras** [7] – Building, training, and evaluating the CNN model
- **Matplotlib** [8] – For visualizing model performance (accuracy, precision, recall, etc.)
- **LabelEncoder & StandardScaler** – For data encoding and normalization

### 2.3.3 ML Techniques

- **Random Forest:** A commonly-used machine learning algorithm, trademarked by Leo Breiman and Adele Cutler, that combines the output of multiple decision trees to reach a single result. It was used in this project for feature selection, to identify which features or parameter holds the most importance for classification.
- **Convolutional Neural Network:** A convolutional neural network (CNN) is a category of machine learning model. Specifically, it is a type of deep learning algorithm that is well suited to analyzing visual data. CNNs are commonly used to process image and video tasks.

# CHAPTER 3: TESTING & EVALUATION

The trained CNN model was tested and evaluated on 20% of the dataset (approx. 20,625 samples) giving an overall accuracy of 95.09%, along with a precision of 0.99 and a recall of 0.99 as well, indicating strong performance in classification of different network packets. The final classification report showed high precision and recall for most classes, with very strong results for Non-Tor class and respectable scores for VPN and Non-VPN classes. A confusion matrix was also generated to visualize the class-wise predictions and misclassifications of the model as follows:

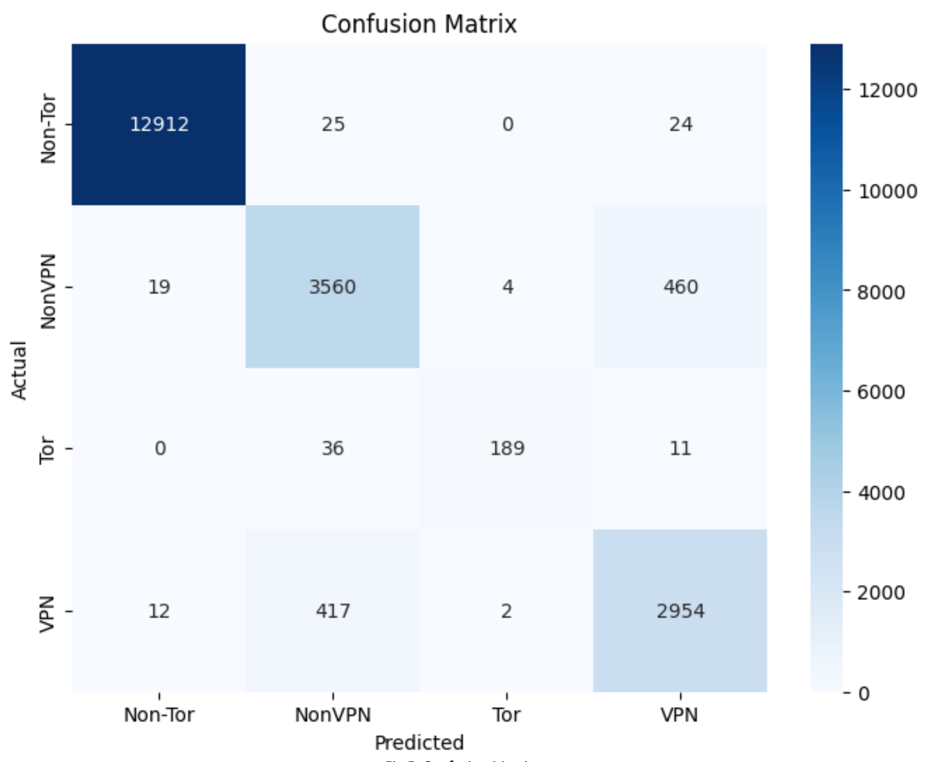


Fig 5. Confusion Matrix

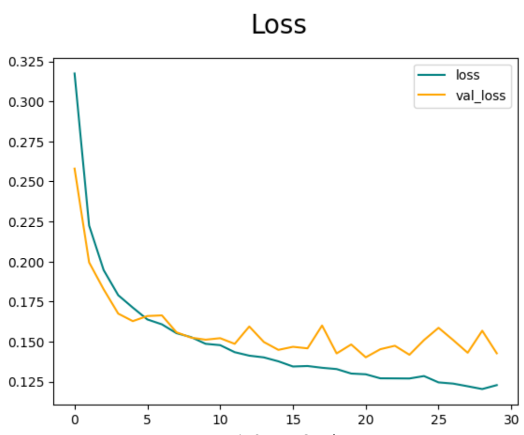


Fig 3. Loss Graph

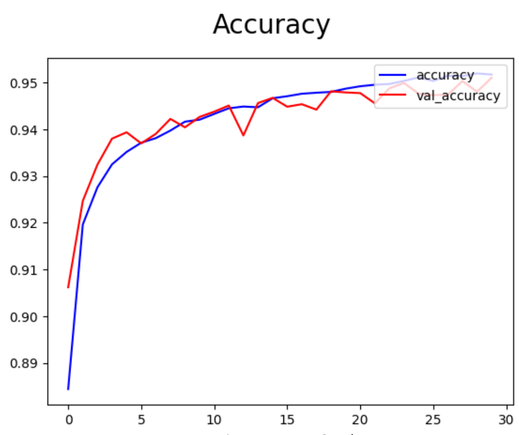


Fig 4. Accuracy Graph

# CHAPTER 4: CONCLUSION

In this project, we successfully implemented a CNN-based traffic classification model to distinguish between Tor, VPN, and regular network traffic using the CIC-Darknet2020 dataset. Through feature selection with RandomForestClassifier, we identified the most significant attributes, improving model efficiency and reducing computational overhead. Standardization and preprocessing ensured balanced and normalized data for effective model training. The 1D CNN model was trained and evaluated, achieving high accuracy in classification, with precision, recall, and overall accuracy recorded in percentage form. The confusion matrix and confidence scores provided further insight into classification performance, highlighting areas of strong differentiation and potential misclassification. This project demonstrates how deep learning techniques can enhance cybersecurity by accurately classifying encrypted and anonymized network traffic, aiding in network monitoring, threat detection, and anomaly identification. Future improvements could involve experimenting with TCN (Temporal Convolutional Networks), optimizing hyperparameters, and testing with real-time traffic data to further enhance the model's robustness

## REFERENCES

[1] https://www.kaggle.com/datasets/dhoogla/cicdarknet2020/code

[2] https://www.unb.ca/cic/datasets/darknet2020.html

[3] https://pandas.pydata.org/

[4] https://numpy.org/

[5] https://scikit-learn.org/stable/

[6] https://www.tensorflow.org/

[7] https://keras.io

[8] https://matplotlib.org/