National College of Ireland

# Bridging the Gap Between Detection and Enforcement: A Critical Review of Deep Learning-Based Zero-Trust Intrusion Detection Models

Practicum

MSc Cybersecurity


Muhammad Saqib

Student ID: X23374501

School of Computing

National College of Ireland


Supervisor: Prof. Vikas Sahni

**National College of Ireland**

**MSc Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Muhammad Saqib |
| **Student ID:** | X23374501 |
| **Programme:** | MSc Cyber Security **Year:** 2025 |
| **Module:** | Practicum Part 2 |
| **Supervisor:** | Prof. Vikas Sahni |
| **Submission Due Date:** | 11/12/2025 |
| **Project Title:** | Bridging the Gap Between Detection and Enforcement: A Critical Review of Deep Learning-Based Zero-Trust Intrusion Detection Models |
| **Word Count:** | **970** **Page Count 5** |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** ……M Saqib ……

**Date:** ……11/12/2025……

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual & Replication Guide

**Project Title:** Bridging the Gap Between Detection and Enforcement:

A Critical Review of Deep Learning-Based Zero-Trust Intrusion Detection Models

**Author:** Muhammad Saqib (X23374501)

**Platform:** Google Colab Pro (Cloud-Based)

# 1. Introduction

The below manual is a sequential instruction on how to install the experiment environment, the software dependencies, and the Deep Learning pipeline that was created in this study. The system will be modular and the data preprocessing, model training, and the interactive Zero-Trust simulation will be separated into separate executables.

In order to make the project reproducible, Google Colab is used, with support to faster training on the NVIDIA GPUs.

# 2. System Requirements

Ahead of the code implementation, make sure that your environment has the following specifications. As this project is based on the cloud infrastructure, the hardware specifications are the virtual instance deployed by Google Colab.

## 2.1 Hardware Configuration

- **Platform:** Google Colab Pro (Recommended).
- **GPU:** NVIDIA T4 (16GB VRAM) or NVIDIA A100 (40GB VRAM).
    - *Note: Deep Learning training on CPU is not recommended due to high latency.*
- **RAM:** High-RAM Runtime (approx. 32GB) is required to handle the SMOTE balancing process for large datasets.
- **Storage:** Google Drive with at least 5GB of free space.

## 2.2 Software Stack

The construction of the project takes place in Python 3.10+. The core libraries which are required are as follows:

# 3. Installation & Setup

### Step 1: Directory Structure

Your Google drive will require you to develop a particular folder structure in order to enable the scripts to find datasets and save models automatically.

1. Open **Google Drive**.
2. Create a main folder named: `MSc_ZeroTrust_Project`.
3. Inside this folder, create two sub-folders:
    - `Datasets/` (Upload your CSV files here, e.g., `CICIDS2017.csv`, `Edge-IIoTset.csv`).
    - `Artifacts/` (Leave this empty; the script will save trained models here).

### Step 2: Environment Initialization

Open the first notebook (`Zero-Trust Intrusion Detection Models.ipynb`) in Google Colab. Run the following command in the first cell to mount your drive and install dependencies:

# 4. Execution Workflow

The project is divided into two phases. You must run **Phase 1** completely before attempting **Phase 2**.

### Phase 1: Model Training & Feature Engineering

*File:* `Zero-Trust Intrusion Detection Models.ipynb`

This script handles the heavy lifting: loading data, training the AI, and saving the "brain" of the system.

1. **Load Data:** Run the cells under "Data Loading." The script will automatically merge the CSV files and remove `NaN` or `Infinity` values.
2. **Feature Selection:** The script uses Random Forest to identify the Top 20 features.

3. **Balancing (SMOTE):** The script will apply SMOTE to the training data.
    - *Warning:* This step may take 5–10 minutes depending on RAM availability.
4. **Model Training:** The 1D-CNN model will train for 10-20 epochs.
    - *Early Stopping:* Training may stop early if validation accuracy stabilizes.
5. **Export Artifacts:** Upon completion, the script automatically saves three critical files to your `Artifacts/` folder:
    - `cnn_model.h5` (The trained AI).
    - `scaler.pkl` (The mathematical scaler).
    - `features.pkl` (The list of 20 features).

### Phase 2: Interactive SOC Simulation

*File:* `Zero-Trust DL-IDS SOC_Simulation.ipynb`

This script launches the dashboard. It does **not** require training the model again; it simply loads the saved files from Phase 1.

1. **Load Artifacts:** Run the setup cells. The script will look for `cnn_model.h5` in your Drive.
2. **Launch Dashboard:** Execute the final cell containing the `gradio` launch command.
3. **Access Interface:** Click the public link generated (e.g., `https://...gradio.live`).

# 5. Using the Dashboard

Once the Gradio interface is running, you can simulate network attacks:

1. **Input Parameters:** Use the sliders to adjust network traffic values (e.g., *Flow Duration*, *Packet Size*).
    - *Tip:* Click the **"Simulate Attack"** button to auto-fill the fields with data typical of a DDoS attack.
2. **Run Inference:** Click **"Analyze Traffic."**
3. **View Results:**
    - **Risk Gauge:** Visualizes the threat level (Green/Red).
    - **Policy Decision:** Displays the automated action (e.g., **BLOCK**).
    - **Audit Log:** Scroll down to see the timestamped record of the decision.

# 6. Troubleshooting

*This manual acknowledges the fact that the software artifacts that accompany this research are operational, reproducible as well as documented, in accordance to the standards of academic documentation*.

Screenshot of code necessary libraries:

```
!pip install xgboost imbalanced-learn --quiet

import os
import glob
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount("/content/drive")

# Paths (adjust if needed)
CICIDS_FOLDER    = "/content/drive/MyDrive/Data sets/MachineLearningCVE"
EDGE_IOT_ML_FILE = EDGE_IOT_FOLDER + "/ML-EdgeIIoT-dataset.csv"
print("CICIDS folder:", CICIDS_FOLDER)
print("Edge-IIoT ML file:", EDGE_IOT_ML_FILE)

print("\nCICIDS files:")
!ls "/content/drive/MyDrive/Data sets/MachineLearningCVE"

print("\nEdge-IIoTset files:")
!ls "/content/drive/MyDrive/Data sets/Edge-IIoTset Dataset"
```

```python
def load_cicids2017(folder_path):
    csv_files = glob.glob(os.path.join(folder_path, "*.csv"))
    print(f"Found {len(csv_files)} CSV files:")
    for f in csv_files:
        print(" -", os.path.basename(f))

    dfs = []
    for f in csv_files:
        print(f"   Reading: {os.path.basename(f)}")
        df = pd.read_csv(
            f,
            index_col=None,
            header=0,
            skipinitialspace=True
        )
        dfs.append(df)

    df_all = pd.concat(dfs, ignore_index=True)
    return df_all

cic_df = load_cicids2017(CICIDS_FOLDER)
print("\nMerged CICIDS shape:", cic_df.shape)
cic_df.head()
```

```python
    dataset["Binary_Label"] = dataset["Label"].apply(lambda x: 0 if x == "BENIGN" else 1)

    print("\nBinary Label distribution (0=Benign, 1=Malicious):")
    print(dataset["Binary_Label"].value_counts())

    print("\n Data ready for EDA and splitting.")
```

```
Binary Label distribution (0=Benign, 1=Malicious):
Binary_Label
0    2271320
1     556556
Name: count, dtype: int64

 Data ready for EDA and splitting.
```

```python
from sklearn.model_selection import train_test_split

print(" Splitting into Train / Val / Test...")

non_feature_cols = [
    "Label", "Binary_Label",
    "Flow ID", "Source IP", "Destination IP",
    "Src IP", "Dst IP", "Timestamp"
]
non_feature_cols = [c for c in non_feature_cols if c in dataset.columns]

X = dataset.drop(columns=non_feature_cols)
y = dataset["Binary_Label"]

print("Feature shape:", X.shape)

X_train, X_temp, y_train, y_temp = train_test_split(
    X, y,
    test_size=0.30,
    stratify=y,
    random_state=42
)

X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp,
```

```python
from sklearn.ensemble import RandomForestClassifier

print(" Running Random Forest for feature selection...")

rf_fs = RandomForestClassifier(
    n_estimators=200,
    n_jobs=-1,
    random_state=42
)
rf_fs.fit(X_train, y_train)

feat_imp = pd.DataFrame({
    "feature": X_train.columns,
    "importance": rf_fs.feature_importances_
}).sort_values(by="importance", ascending=False)

TOP_N = 20
top_features = feat_imp.head(TOP_N)["feature"].tolist()

print(f"\nTop {TOP_N} features:")
print(top_features)

X_train_top = X_train[top_features].copy()
X_val_top   = X_val[top_features].copy()
X_test_top  = X_test[top_features].copy()
```

```python
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import MinMaxScaler

print(" Applying SMOTE on training set only...")

smote = SMOTE(random_state=42)
X_train_sm, y_train_sm = smote.fit_resample(X_train_top, y_train)

print("After SMOTE:")
print("Train:", X_train_sm.shape)
print("Label distribution:")
print(pd.Series(y_train_sm).value_counts())

print("\n Scaling features with MinMaxScaler...")

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train_sm)
X_val_scaled   = scaler.transform(X_val_top)
X_test_scaled  = scaler.transform(X_test_top)

print("Scaled shapes:")
print("Train:", X_train_scaled.shape)
print("Val:",   X_val_scaled.shape)
print("Test:",  X_test_scaled.shape)
```

```python
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import xgboost as xgb

def print_metrics(name, y_true, y_pred, y_proba):
    print(f"\n===== {name} =====")
    print(classification_report(y_true, y_pred))
    auc = roc_auc_score(y_true, y_proba)
    cm  = confusion_matrix(y_true, y_pred)
    print("ROC-AUC:", auc)
    print("Confusion matrix:\n", cm)
    return auc

print(" Training RandomForest baseline...")
rf_clf = RandomForestClassifier(
    n_estimators=300,
    n_jobs=-1,
    random_state=42
)
rf_clf.fit(X_train_scaled, y_train_sm)

y_pred_rf  = rf_clf.predict(X_test_scaled)
y_proba_rf = rf_clf.predict_proba(X_test_scaled)[:, 1]
rf_auc = print_metrics("RandomForest – CICIDS2017", y_test, y_pred_rf, y_proba_rf)

print("\n Training XGBoost baseline...")
```

```python
import tensorflow as tf
from tensorflow.keras import layers, models

print(" Reshaping data for 1D-CNN...")

n_features = X_train_scaled.shape[1]

X_train_cnn = X_train_scaled.reshape(-1, n_features, 1)
X_val_cnn   = X_val_scaled.reshape(-1, n_features, 1)
X_test_cnn  = X_test_scaled.reshape(-1, n_features, 1)

print("Train:", X_train_cnn.shape)
print("Val:",   X_val_cnn.shape)
print("Test:",  X_test_cnn.shape)
```

```
Reshaping data for 1D-CNN...
rain: (3179848, 20, 1)
al: (424181, 20, 1)
est: (424182, 20, 1)
```

```python
def build_cnn(input_shape):
    model = models.Sequential()
    model.add(layers.Conv1D(64, 3, activation='relu', input_shape=input_shape))
    model.add(layers.Conv1D(64, 3, activation='relu'))
    model.add(layers.MaxPooling1D(2))
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv1D(128, 3, activation='relu'))
    model.add(layers.MaxPooling1D(2))
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dropout(0.3))
    model.add(layers.Dense(1, activation='sigmoid'))

    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
    return model

cnn_model = build_cnn((n_features, 1))
cnn_model.summary()
```

```python
y_proba_cnn = cnn_model.predict(X_test_cnn).flatten()
y_pred_cnn  = (y_proba_cnn >= 0.5).astype(int)

cnn_auc = print_metrics("1D-CNN – CICIDS2017", y_test, y_pred_cnn, y_proba_cnn)
```

```
13256/13256 ━━━━━━━━━━━━━━━━━━━━ 12s 889us/step

===== 1D-CNN – CICIDS2017 =====
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    340698
           1       0.94      1.00      0.97     83484

    accuracy                           0.99    424182
   macro avg       0.97      0.99      0.98    424182
weighted avg       0.99      0.99      0.99    424182

ROC-AUC: 0.9992279868569451
Confusion matrix:
 [[334967   5731]
 [   174  83310]]
```

```python
)

history_edge = cnn_edge.fit(
    X_edge_train_cnn, y_edge_train_sm,
    validation_split=0.2,
    epochs=40,
    batch_size=256,
    callbacks=[early_stop_edge],
    verbose=1
)

y_edge_proba_cnn = cnn_edge.predict(X_edge_test_cnn).flatten()
y_edge_pred_cnn  = (y_edge_proba_cnn >= 0.5).astype(int)

print_metrics("1D-CNN – Edge-IIoTset (trained on Edge)", y_edge_test, y_edge_pred_cnn, y_edge_proba_cnn)
```

```
Edge CNN shapes: (186898, 42, 1) (47340, 42, 1)
Epoch 1/40
585/585 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.7742 - loss: 0.4350 - val_accuracy: 0.6719 - val_loss: 0.4354
Epoch 2/40
585/585 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8567 - loss: 0.2900 - val_accuracy: 0.8149 - val_loss: 0.3196
Epoch 3/40
585/585 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.8880 - loss: 0.2406 - val_accuracy: 0.8833 - val_loss: 0.2506
Epoch 4/40
585/585 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9025 - loss: 0.2079 - val_accuracy: 0.8942 - val_loss: 0.2156
```