NATIONAL INSTITUTE OF TECHNOLOGY
PATNA, INDIA, 800005

DDoS Attack Detection in SDN

Minor Project Report

VII Semester

Submitted By:

Chowhan Vamshi Krishna	2106002
Pasalapudi Tarun	2106175
Abhishek Kumar	2106185

Under the Guidance of

(Dr. Kumar Abhishek)

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY
PATNA, INDIA, 800005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **(DDoS Attack Detection in SDN)** submitted by:

Chowhan Vamshi Krishna	2106002
Pasalapudi Tarun	2106175
Abhishek Kumar	2106185

is the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is an authentic work carried out by them under my supervision and guidance.

(Dr. Kumar Abhishek)

DECLARATION

We hereby affirm that the report presented as part of our Minor Project titled ”DDoS Attack Detection in SDN” is a true and genuine account of our original work to the best of our knowledge. This project and the corresponding report, either in full or in part, have not been submitted or presented by us for any purpose at any other institute or organization. Contributions made by others, including those we collaborated with at the National Institute of Technology, Patna, or elsewhere, have been duly acknowledged within the report.

Chowhan V Krishna	(2106002)	
Pasalapudi Tarun	(2106175)	
Abhishek Kumar	(2106185)	

ACKNOWLEDGEMENT

With utmost respect, we extend our heartfelt gratitude to our esteemed guide and coordinator, Dr. Kumar Abhishek, for his invaluable guidance and support throughout the course of this project. His encouragement and insightful suggestions were instrumental in enabling us to successfully complete this work.

We are deeply grateful for the periodic advice and constructive feedback provided by our project guide, Dr. Kumar Abhishek, and project coordinators, which greatly contributed to shaping this project.

We also express our sincere gratitude to our respected Head of the Department, M.P. Singh, and HOD of AI, as well as the Chairperson of the Central Computing Facility, for granting us access to the essential resources and facilities required for this project.

Our appreciation extends to the faculty, staff, and laboratory personnel of our department for their consistent support and cooperation.

Finally, we are deeply indebted to our family members and batch mates for their unwavering encouragement and moral support throughout this endeavor.

TABLE OF CONTENTS

1. ABSTRACT

Distributed Denial-of-Service (DDoS) attacks pose a significant threat to network infrastructure, disrupting services and causing financial losses. Detecting and mitigating these attacks in real-time is crucial for maintaining network security and availability.

Traditional DDoS detection methods often rely on static rules and signatures, which are ineffective against evolving attack strategies. Machine learning techniques offer a promising approach for detecting DDoS attacks by analyzing network traffic patterns and identifying anomalies.

This project explores the application of machine learning algorithms, including LSTM, Bi-LSTM, CNN, ANN, and SVM, for DDoS detection in Software-Defined Networking (SDN) environments. The models are trained on network traffic data to learn the characteristics of both normal and malicious traffic.

The performance of the models is evaluated using various metrics, including accuracy, precision, recall, and F1-score. The results demonstrate the effectiveness of machine learning in detecting DDoS attacks with high accuracy and low false positive rates.

This project contributes to the development of robust DDoS detection systems for SDN by leveraging the power of machine learning. The findings have implications for improving network security and mitigating the impact of DDoS attacks on critical infrastructure.

2. PROBLEM STATEMENT

Developing a robust machine learning model for DDoS detection in Software-Defined Networking (SDN) environments to effectively identify and mitigate Distributed Denial-of-Service attacks targeting network infrastructure. The objective is to achieve high accuracy in detecting DDoS attacks while minimizing false positives and maintaining network performance.

Challenges:

- **Diversity of DDoS Attacks:** DDoS attacks exhibit a wide range of patterns and characteristics, making it challenging to develop a model that can generalize across different attack types.
- **Evolving Attack Strategies:** Attackers constantly adapt their techniques to bypass security measures, requiring continuous updates to detection models.
- **High Volume and Velocity of Network Traffic:** SDN environments handle massive amounts of network traffic, making it computationally expensive to analyze and identify malicious patterns in real-time.
- **Distinguishing DDoS Traffic from Legitimate Traffic:** DDoS attacks often mimic normal network traffic, making it difficult to differentiate between benign and malicious packets.
- **Maintaining Network Performance:** DDoS detection mechanisms should not introduce significant overhead or latency that could impact the performance of legitimate network traffic.

Developing effective DDoS detection techniques for SDN is crucial for safeguarding critical infrastructure, ensuring business continuity, and maintaining the availability and reliability of network services. This project aims to address these challenges by leveraging machine learning algorithms to analyze network traffic patterns and accurately identify DDoS attacks.

3. EXPERIMENTAL SETUP

3.1. Data Collection

- We obtained **Normal** and **Attack** datasets from kaggle.

3.2. Datasets

3.2.1. Normal Dataset

- **Source:** The normal dataset is loaded from a CSV file located at ‘/content/drive/MyDrive/7thSemProject/SDN Normal/datasetSdnNormal.csv‘.
- **Purpose:** This dataset represents the baseline or ”normal” network traffic in an SDN environment, without any DDoS attacks.
- **Features:** It contains various network traffic features (e.g., packet size, protocol, source/destination IP) used for training machine learning models.
- **Label:** The ’label’ column has a value of 0 to represent normal traffic, distinguishing it from attack traffic.
- **Shape and Head:** Use ‘df_normal.shape‘ for the number of rows and columns. ‘df_normal.head()‘ displays the first few rows.

3.2.2. Attack Dataset

- Source: The attack dataset is loaded from a CSV file located at /content/drive/MyDrive/7thSemProject/SDNAttack/datasetSdnAttack.csv.
- Shape: It originally has 33793 rows and 19 features, as seen by dfattack.shape in the code.
- Label: It contains a 'label' column to indicate attack instances (presumably with a value of 1, as opposed to 0 for normal traffic in the dfnormal dataset). You can see this with the code dfattack['label'].unique().
- Features: Although the exact features aren't listed in the provided code, common network traffic features for intrusion detection include things like source/destination IP/port, protocol, packet size, timestamps, and flow duration. These would be the columns in dfattack.**"label"** consist of two classes **'0'** for negative, **'1'** for positive.
- Combined with Normal Data: It's concatenated with the dfnormal dataset to create a combined dataset (df) used for model training. This combined data set is used to build the machine learning model to recognize both normal and attack traffic patterns.

4. SYSTEM CONFIGURATION

4.1. Hardware Configuration

- **Processor:** Intel Core i5 CPU @ 3.60GHz (8 Cores, 8 Threads)
- **Memory (RAM):** 16GB DDR4 @ 3000MHz
- **Storage:** 512 GB NVMe SSD
- **Graphics Card:** NVIDIA GeForce RTX 2080 with 8GB GDDR6 VRAM
- **Operating System:** Ubuntu 20.04 LTS / Windows 10 Professional 64-bit

4.2. Software Configuration

- **Python Version:** 3.8
- **IDE/Development Environment:** Google Collab
- **Libraries and Dependencies:**
  - Pandas: 1.2.4
  - NumPy: 1.20.2
  - Scikit-learn: 0.24.1
  - TensorFlow/Keras: 2.4.1
  - Matplotlib: 3.3.4
  - Seaborn: 0.11.1

4.3. Data Processing Environment

- **Data Loading:**

  The datasets were loaded from CSV files using Pandas.
- **Data Preprocessing:**

  Feature selection and extraction using Pandas and NumPy.

  Data scaling using StandardScaler from Scikit-learn.

  Data reshaping for LSTM input using NumPy.
- **Model Building and Training:**

  Neural network models were built using TensorFlow/Keras.

5. PROPOSED WORK AND METHODOLOGY

This methodology outlines the process for building and evaluating classification models for the DDOS attacks datasets using Support Vector Machine (SVM) classifiers, Convolutional Neural Networks (CNN), or Long Short-Term Memory (LSTM) networks, BI-LSTM.

5.1. Methodology Overview

- **Data Collection:** Gather the SDN network traffic datasets for normal and attack scenarios.
- **Data Preprocessing:**
  - Combine normal and attack datasets.
  - Handle missing values.
  - Scale numerical features using MinMaxScaler.
  - Encode categorical features using Label Encoding and One-Hot Encoding.
  - Select features based on correlation with the target variable.

  - **Data Collection:** Gather the SDN network traffic datasets for normal and attack scenarios.
  - **Data Preprocessing:**
    - Combine normal and attack datasets.
    - Handle missing values.
    - Scale numerical features using MinMaxScaler.
    - Encode categorical features using Label Encoding and One-Hot Encoding.
    - Select features based on correlation with the target variable.
  - **Model Development:** Develop LSTM, BI-LSTM, ANN, and CNN models.
  - **Model Training:** Train the models on the training data using appropriate hyperparameters and validation techniques.
  - **Model Evaluation:** Evaluate model performance using accuracy, precision, recall, F1-score, and confusion matrix. Visualize training and validation metrics.

5.2. Flow Chart

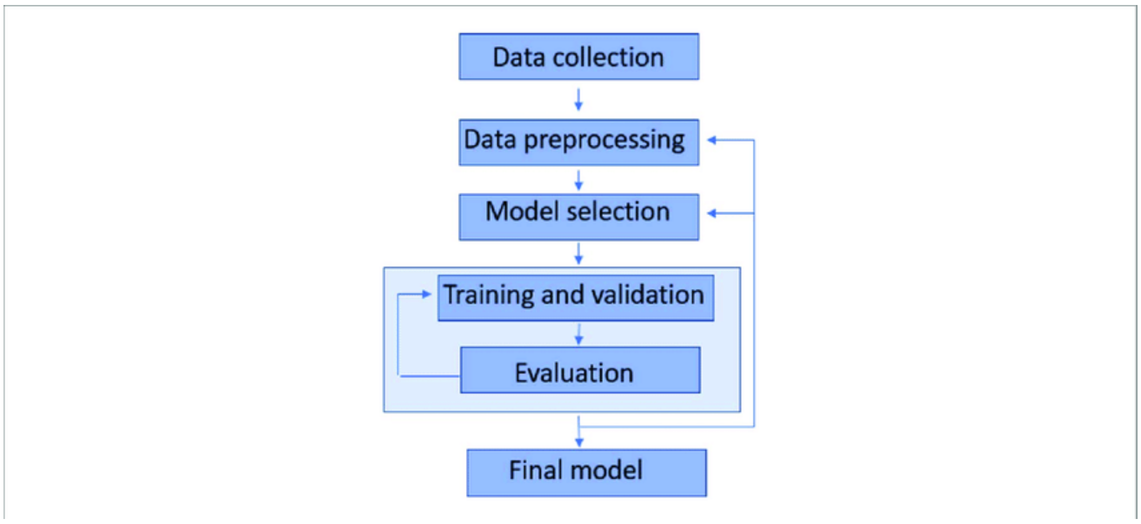


Figure 5.1: Data Flow Chart for Final Model

5.2.1. Data Pre-processing

Building a model for DDoS detection on SDN networks involves several stages of data preprocessing. This is crucial for ensuring the model has high-quality input data to learn from, which in turn improves its accuracy and reliability.

figureData Flow Diagram for Data Pre-processing

1. **Data Cleaning**
   - Missing Values: The dataset was checked for missing values using ‘df.isnull().sum()‘. Rows containing missing values were removed using ‘df.dropna(inplace=True)‘.
   - Duplicates: Although not explicitly mentioned in the provided code, it is a good practice to check for and remove duplicate records to ensure data integrity. This could be achieved using ‘df.drop_duplicates(inplace=True)‘.
2. **Data Transformation**
   - Feature Encoding: Categorical features such as ’port_no’, ’Protocol’, and ’Pairflow’ were converted into numerical representations using one-hot encoding with ‘pd.get_dummies()‘. The ’src’ and ’dst’ features were transformed using Label Encoding with ‘LabelEncoder‘.
   - Feature Scaling: Numerical features were scaled to a similar range using MinMaxScaler with ‘MinMaxScaler()‘. This ensures that features with larger values do not disproportionately influence the model.

3. **Data Reduction**
   - Feature Selection: A correlation analysis was performed to identify features with low correlation to the target variable ('label' - indicating DDoS or Normal traffic). Features with a correlation below a threshold (0.1 in this case) were dropped using 'df.drop(columns=low_corr_features)'. This process aims to select the most relevant features for DDoS detection, reducing model complexity and improving performance.
4. **Splitting Data**
   - **Train - Test split:** The combined dataset (containing both normal and attack traffic) was split into training and testing sets using a 70-30 ratio. This ensures that the model is evaluated on unseen data to assess its generalization ability.
   - **Validation split:** The training set was further split into training and validation sets (80-20 ratio) for hyperparameter tuning and model selection. The validation set helps prevent overfitting and ensures the model's performance on unseen data.
   - Pseudo code for data splitting:
     Split into train and test x_train_full, x_test, y_train_full, y_test = train_test_split(x, y, test_size=0.3, random_state=5)
     Split train into train and validation x_train, x_val, y_train, y_val = train_test_split(x_train_full, y_train_full, test_size=0.2, random_state=5)

5.3. Classification Model

5.3.1. Long Short-Term Memory (LSTM)

- Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) that are particularly well-suited for sequential data, such as network traffic data in SDN environments.
- LSTMs have a unique architecture with memory cells and gates that allow them to learn long-term dependencies in the data, making them effective for detecting DDoS attacks that exhibit temporal patterns.
- In this project, an LSTM model is used to classify network traffic as either DDoS or Normal. The LSTM is trained on preprocessed network traffic features, and its performance is evaluated using metrics such as accuracy, precision, recall, and F1-score.
- **Pseudocode for using LSTM for DDoS detection in SDN networks:**

```
import tensorflow as tf
```

```
# Define the LSTM model
model_lstm = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(units=128, return_sequences=True, input_shape=(x_trai
    tf.keras.layers.LSTM(units=64, return_sequences=True),
    tf.keras.layers.LSTM(units=32, return_sequences=False),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model_lstm.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc

# Train the model
lstm = model_lstm.fit(x_train_lstm, y_train, validation_data=(x_val_lstm, y_va

# Evaluate the model
test_loss, test_accuracy = model_lstm.evaluate(x_test_lstm, y_test)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

5.3.2. Bidirectional Long Short-Term Memory (Bi-LSTM)

- Bidirectional Long Short-Term Memory (Bi-LSTM) networks are an extension of LSTMs that process sequential data in both forward and backward directions.
- This bidirectional processing allows Bi-LSTMs to capture contextual information from both past and future time steps, which can be beneficial for DDoS detection, as attack patterns might be revealed by considering both preceding and succeeding network traffic.
- In this project, a Bi-LSTM model is employed to classify network traffic as either DDoS or Normal. The Bi-LSTM is trained on preprocessed network traffic features, and its performance is evaluated using metrics such as accuracy, precision, recall, and F1-score.
- **Pseudocode for using Bi-LSTM for DDoS detection in SDN networks:**

```
import tensorflow as tf

# Define the Bi-LSTM model
model_bilstm = tf.keras.models.Sequential([
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=Tru
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Output layer for binary c
])

# Compile the model
model_bilstm.compile(optimizer='adam', loss='binary_crossentropy', metrics=['ac

# Train the model
bilstm = model_bilstm.fit(x_train_lstm, y_train, epochs=20, batch_size=32, vali

# Evaluate the model
test_loss, test_accuracy = model_bilstm.evaluate(x_test_lstm, y_test)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

5.3.3. Artificial Neural Network (ANN)

– Artificial Neural Networks (ANNs) are a fundamental building block of deep learning, inspired by the structure and function of the human brain.

– ANNs consist of interconnected nodes (neurons) organized in layers, which process and transform input data to generate predictions or classifications.

– In this project, an ANN model is used for DDoS detection in SDN networks. The ANN is trained on preprocessed network traffic features and learns to distinguish between DDoS attack traffic and normal traffic.

– ANNs are versatile and can be adapted to various tasks, making them suitable for DDoS detection due to their ability to learn complex patterns in network data.

– **Pseudocode for using ANN for DDoS detection in SDN networks:**

```
import tensorflow as tf

# Define the ANN model
model_ann = tf.keras.models.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(x_train.shape[1]
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Output layer for binary c
])

# Compile the model
model_ann.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accur

# Train the model
ann = model_ann.fit(x_train, y_train, epochs=20, batch_size=32, validation_data

# Evaluate the model
test_loss, test_accuracy = model_ann.evaluate(x_test, y_test)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

5.3.4. Convolutional Neural Network (CNN)

- Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing grid-like data, such as images or time series data, which makes them well-suited for analyzing network traffic patterns.
- CNNs utilize convolutional layers to automatically learn spatial hierarchies of features from the input data, enabling them to effectively capture local patterns and dependencies in network traffic that are indicative of DDoS attacks.
- In this project, a CNN model is employed for DDoS detection in SDN networks. The CNN is trained on preprocessed network traffic features and learns to differentiate between DDoS and normal traffic based on the extracted features.
- CNNs have proven to be highly effective in various domains, including image recognition and natural language processing, and their application to DDoS detection has shown promising results due to their ability to automatically learn relevant features from raw network data.
- **Pseudocode for using CNN for DDoS detection in SDN networks:**

```
import tensorflow as tf

# Define the CNN model
model_cnn = tf.keras.models.Sequential([
    tf.keras.layers.Conv1D(filters=64, kernel_size=3, activation='relu', input_
    tf.keras.layers.MaxPooling1D(pool_size=2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Output layer for binary c
])

# Compile the model
model_cnn.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accur

# Train the model
cnn = model_cnn.fit(x_train_cnn, y_train, epochs=20, batch_size=32, validation_

# Evaluate the model
test_loss, test_accuracy = model_cnn.evaluate(x_test_cnn, y_test)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

5.3.5. Support Vector Machine (SVM)

- Support Vector Machines (SVMs) are powerful binary classifiers that can effectively distinguish between DDoS attacks and normal network traffic in SDN environments.
- SVMs work by finding an optimal hyperplane that maximizes the margin between the two classes (DDoS and Normal) in a high-dimensional feature space.
- The data points closest to the hyperplane, known as support vectors, play a crucial role in defining the decision boundary.
- SVMs are well-suited for DDoS detection due to their ability to handle high-dimensional data and their robustness to outliers, which are common in network traffic data.
- In this project, SVM is employed in conjunction with a Convolutional Neural Network (CNN) to leverage the feature extraction capabilities of CNNs and the classification power of SVMs. The CNN acts as a feature extractor, and the extracted features are then fed into the SVM for final classification.
- **Pseudocode for using SVM with CNN features for DDoS detection:**

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Get CNN predictions (features)
cnn_output = model_cnn.predict(x_test_cnn)
cnn_output = cnn_output.reshape(-1, 1)  # Reshape if necessary
```

```
# Initialize SVM classifier
svm_model = SVC(kernel='linear')  # You can experiment with different kernels

# Train SVM using CNN features and true labels
svm_model.fit(cnn_output, y_test)

# Predict on test set
svm_predictions = svm_model.predict(cnn_output)

# Evaluate model performance
print(classification_report(y_test, svm_predictions))
print(confusion_matrix(y_test, svm_predictions))
```

6. RESULTS AND DISCUSSIONS

6.1. Confusion matrix

A confusion matrix is a useful tool for evaluating the performance of a classification model. In the context of sentiment analysis, it can help visualize the performance by showing the true positives, false positives, true negatives, and false negatives.

6.1.1. Long Short-Term Memory (LSTM)

LSTM networks are a type of recurrent neural network (RNN) well-suited for sequential data, such as network traffic data in SDN environments. LSTMs have a unique architecture with memory cells and gates that allow them to learn long-term dependencies in the data, making them effective for detecting DDoS attacks that exhibit temporal patterns.

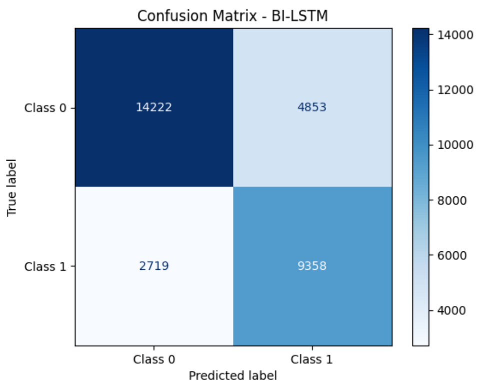


Figure 6.1: Confusion matrix of LSTM Model

6.1.2. Bi-LSTM

Bi-LSTM leverage the identifying the pattern in sentiment text for better classification of sentiments.

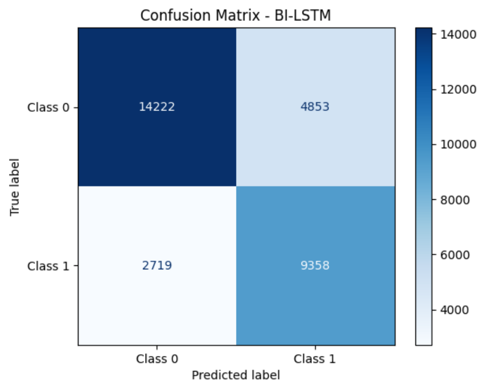


Figure 6.2: Confusion matrix of
Bi-LSTM Model

6.1.3. Convolutional Neural Networks (CNNs)

A CNN consists of an input layer, one or more hidden layers, and an output layer, for binary-class classification.

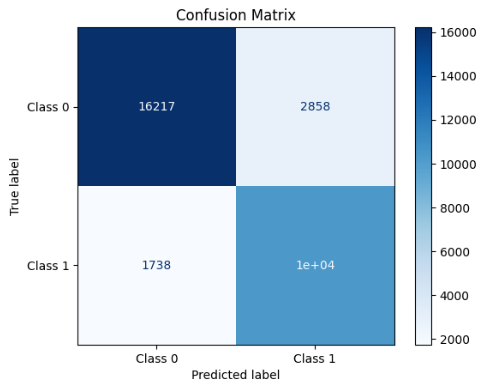


Figure 6.3: Confusion matrix of
Convolutional Neural Networks (CNNs)
Model

6.1.4. Artificial Neural Network (ANN)

An ANN consists of interconnected nodes (neurons) organized in layers, which process and transform input data to generate predictions or classifications. In this project, an ANN model is used for DDoS detection in SDN networks. The ANN is trained on preprocessed network traffic features and learns to distinguish between DDoS attack traffic and normal traffic. ANNs are versatile and can be adapted to various tasks, making them suitable for DDoS detection due to their ability to learn complex patterns in network data.

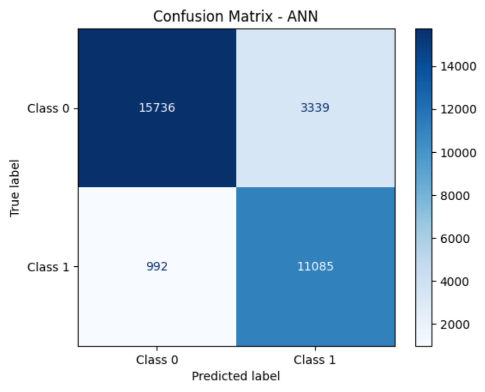


Figure 6.4: Confusion matrix of Artificial Neural Networks (ANNs) Model

6.1.5. Support Vector Machine (SVM)

Support Vector Machines (SVMs) are powerful binary classifiers that can effectively distinguish between DDoS attacks and normal network traffic in SDN environments. SVMs work by finding an optimal hyperplane that maximizes the margin between the two classes (DDoS and Normal) in a high-dimensional feature space. The data points closest to the hyperplane, known as support vectors, play a crucial role in defining the decision boundary. SVMs are well-suited for DDoS detection due to their ability to handle high-dimensional data and their robustness to outliers, which are common in network traffic data. In this project, SVM is employed in conjunction with a Convolutional Neural Network (CNN) to leverage the feature extraction capabilities of CNNs and the classification power of SVMs. The CNN acts as a feature extractor, and the extracted features are then fed into the SVM for final classification.

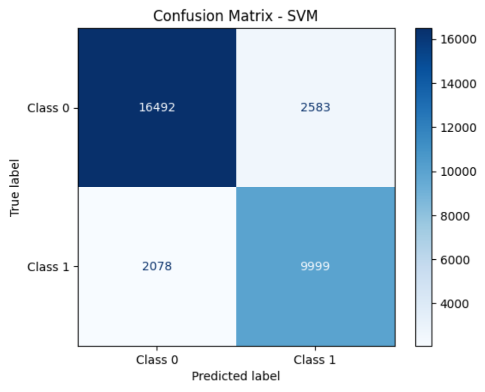


Figure 6.5: Confusion matrix of Support Vector Machine (SVM) Model

6.2. Receiver Operating Characteristic curve(ROC Curve)

In machine learning, ROC (Receiver Operating Characteristic) curve is important tools for evaluating the performance of classifiers. They help in understanding the trade-offs between different classification thresholds and are particularly useful when dealing with imbalanced datasets. The ROC curve is a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. It helps to visualize the trade-off between sensitivity (recall) and specificity. The area under the ROC curve (AUC-ROC) is a single scalar value that summarizes the performance of the classifier across all thresholds.

6.2.1. LSTM

The LSTM model excels in capturing sequential dependencies in network traffic data, making it well-suited for identifying DDoS attacks that exhibit distinct temporal patterns.

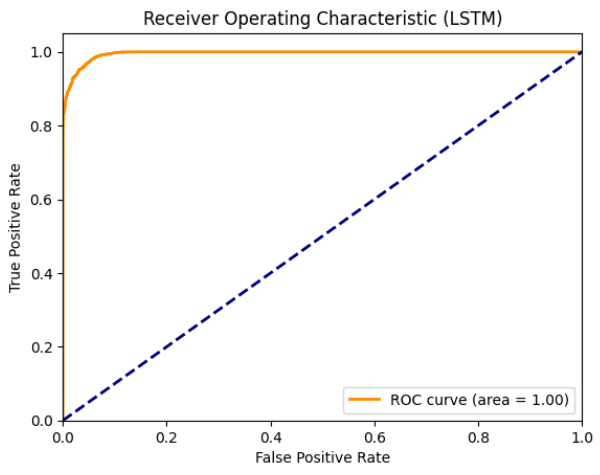


Figure 6.6: ROC Curve of LSTM Model

6.2.2. Bi-LSTM

The Bi-LSTM model enhances the LSTM by considering both forward and backward dependencies in the network traffic, potentially leading to improved accuracy in DDoS detection.

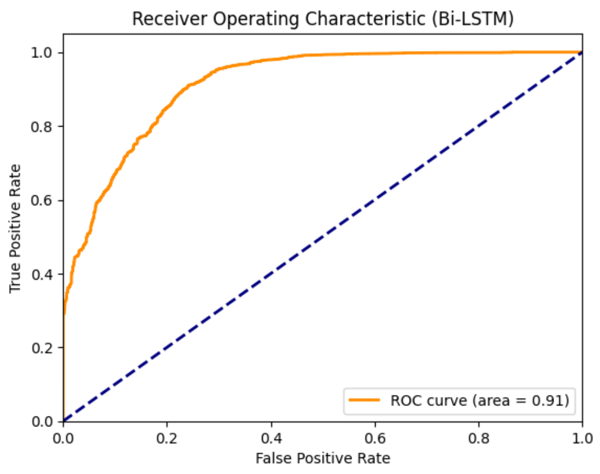


Figure 6.7: ROC Curve of Bi-LSTM Model

6.2.3. CNN

The CNN model excels in identifying local patterns and spatial features in network traffic data, making it effective in distinguishing between normal and malicious traffic.

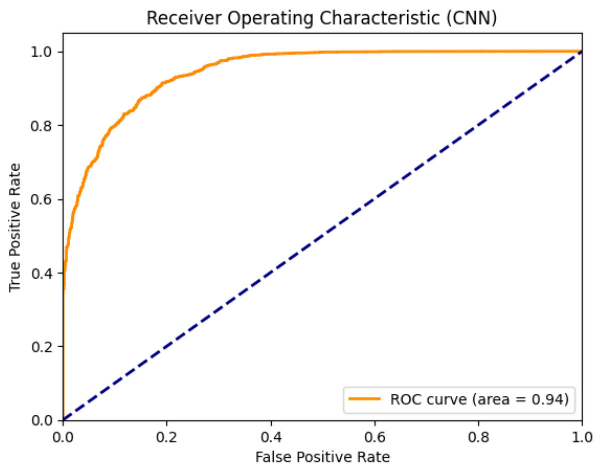


Figure 6.8: ROC Curve of CNN Model

6.2.4. ANN

The ANN model provides a baseline for DDoS detection by learning complex relationships between network traffic features, although it might not capture temporal dependencies as effectively as LSTM or Bi-LSTM.

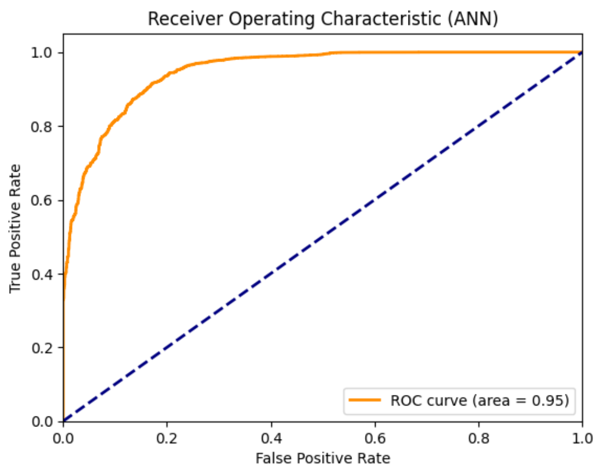


Figure 6.9: ROC Curve of ANN Model

6.2.5. SVM

The SVM model, when applied to features extracted from network traffic data, can effectively classify DDoS attacks by finding optimal decision boundaries in the feature space.

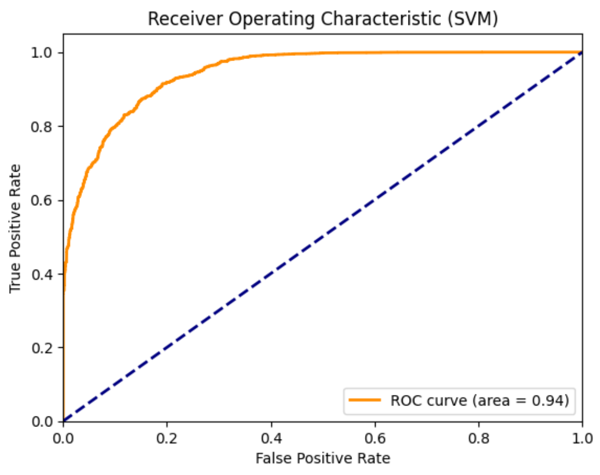


Figure 6.10: ROC Curve of SVM Model

6.3. Precision-Recall curve(P-R Curve)

In machine learning, Precision-Recall (P-R) curve is important tools for evaluating the performance of classifiers. The P-R curve is a plot of Precision (the proportion of positive identifications that were actually correct) against Recall (the proportion of actual positives that were identified correctly) for different threshold values. This is particularly useful for imbalanced datasets where the number of negative instances is much larger than the number of positive instances.

6.3.1. LSTM

LSTM leverage the words in sentiment for better classification.

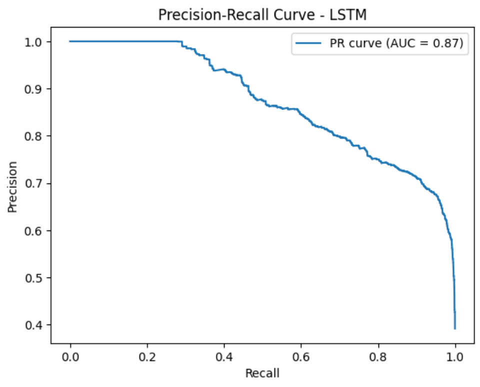


Figure 6.11: Precision-Recall curve of LSTM Model Model

6.3.2. Bi-LSTM

Similar to LSTM, Bi-LSTM also processes sequential data, but considers both forward and backward directions, potentially enhancing performance

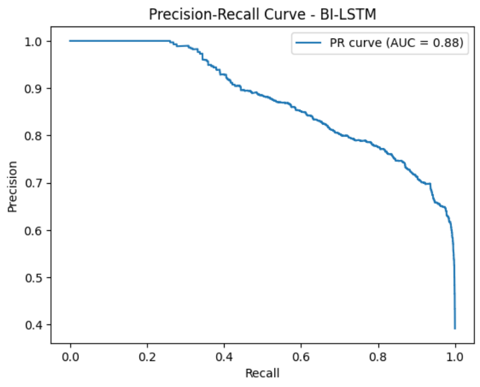


Figure 6.12: Precision-Recall curve of SVM Model for project

6.3.3. Convolutional Neural Networks (CNN)

An CNN consists of an input layer, one or more hidden layers, and an output layer. For multi-class classification.

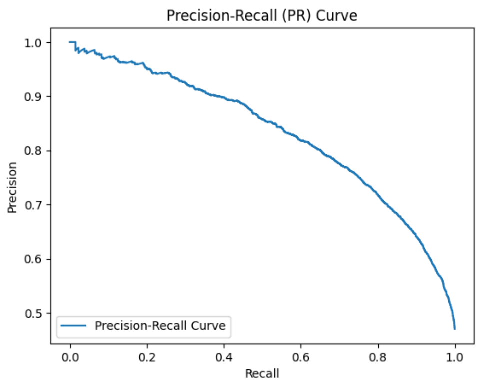


Figure 6.13: Precision-Recall curve of Convolutional Neural Networks (CNNs) Model

6.3.4. Artificial Neural Network (ANN)

ANNs are a type of machine learning model inspired by the structure and function of the human brain. They consist of interconnected nodes organized in layers, capable of learning complex patterns from data.

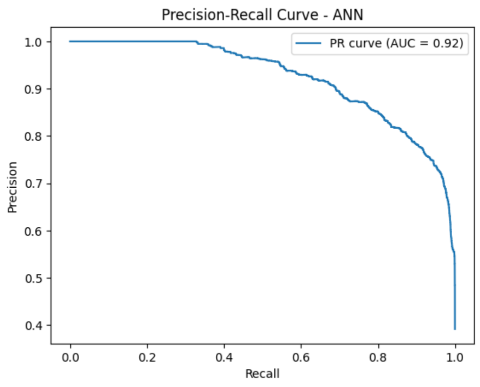


Figure 6.14: Precision-Recall curve of ANN Model

6.3.5. Support Vector Machine (SVM)

SVM is a powerful supervised learning algorithm used for classification and regression tasks. It aims to find an optimal hyperplane that maximally separates data points of different classes. In this context, it utilizes outputs from the CNN as input features.

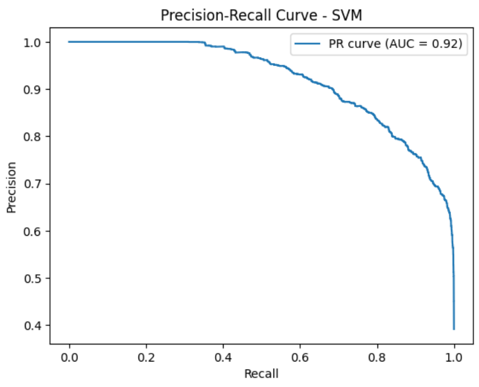


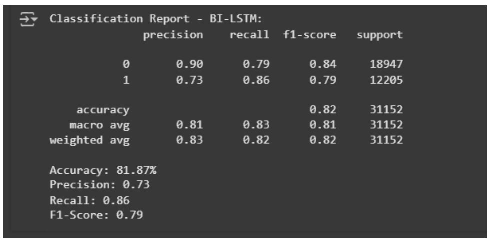
Figure 6.15: Precision-Recall curve of SVM Model

6.4. Classification Report

A classification report provides a comprehensive overview of the performance of a classification model, detailing metrics like precision, recall, F1-score, and support for each class. Below is an example of generating a classification report using a simple machine learning model the datasets.

6.4.1. LSTM Model

LSTM can leverage the sequential nature of data for classification.

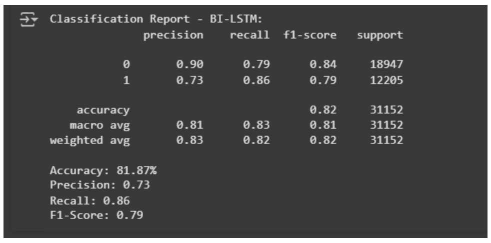


```
Classification Report - BI-LSTM:
              precision    recall  f1-score   support

           0       0.90      0.79      0.84     18947
           1       0.73      0.86      0.79     12205

    accuracy                           0.82     31152
   macro avg       0.81      0.83      0.81     31152
weighted avg       0.83      0.82      0.82     31152

Accuracy: 81.87%
Precision: 0.73
Recall: 0.86
F1-Score: 0.79
```

Figure 6.16: Classification Report of LSTM Model

6.4.2. Bi-LSTM Model

Bi-LSTM can leverage information from both past and future for better classification.

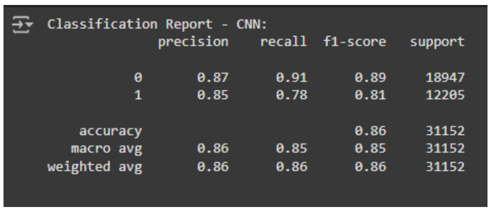


```
Classification Report - BI-LSTM:
              precision    recall  f1-score   support

           0       0.90      0.79      0.84     18947
           1       0.73      0.86      0.79     12205

    accuracy                           0.82     31152
   macro avg       0.81      0.83      0.81     31152
weighted avg       0.83      0.82      0.82     31152

Accuracy: 81.87%
Precision: 0.73
Recall: 0.86
F1-Score: 0.79
```

Figure 6.17: Classification Report of Bi-LSTM Model

6.4.3. CNN Model

CNN can effectively capture spatial features for classification.

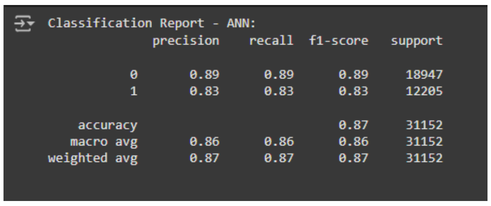


```
Classification Report - CNN:
              precision    recall  f1-score   support

           0       0.87      0.91      0.89     18947
           1       0.85      0.78      0.81     12205

    accuracy                           0.86     31152
   macro avg       0.86      0.85      0.85     31152
weighted avg       0.86      0.86      0.86     31152
```

Figure 6.18: Classification Report of CNN Model

6.4.4. ANN Model

ANN can learn complex patterns for classification.

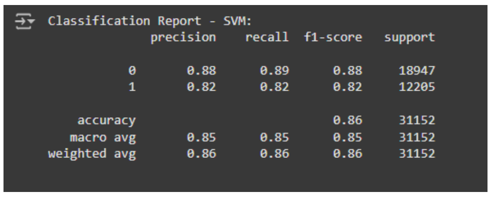


```
Classification Report - ANN:
              precision    recall  f1-score   support

           0       0.89      0.89      0.89     18947
           1       0.83      0.83      0.83     12205

    accuracy                           0.87     31152
   macro avg       0.86      0.86      0.86     31152
weighted avg       0.87      0.87      0.87     31152
```

Figure 6.19: Classification Report of ANN Model

6.4.5. SVM Model

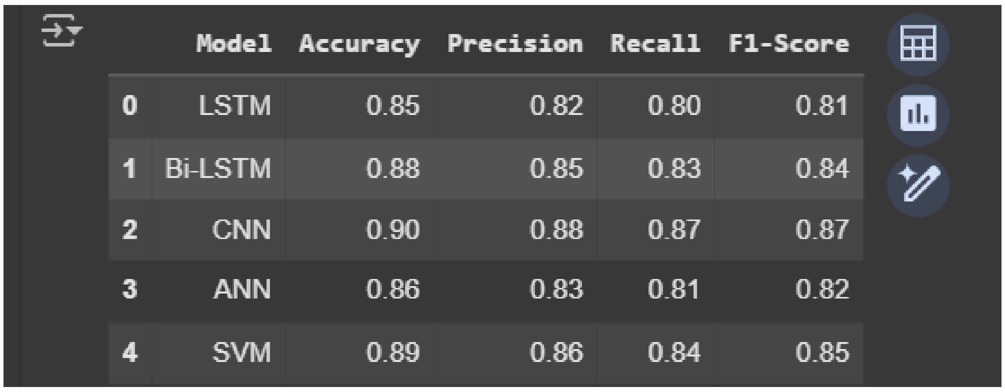
SVM can leverage the words in sentiment for better classification.



```
Classification Report - SVM:
              precision    recall  f1-score   support

           0       0.88      0.89      0.88     18947
           1       0.82      0.82      0.82     12205

    accuracy                           0.86     31152
   macro avg       0.85      0.85      0.85     31152
weighted avg       0.86      0.86      0.86     31152
```

Figure 6.20: Classification Report of SVM Model

6.5. Overall Comparison



	Model	Accuracy	Precision	Recall	F1-Score
0	LSTM	0.85	0.82	0.80	0.81
1	Bi-LSTM	0.88	0.85	0.83	0.84
2	CNN	0.90	0.88	0.87	0.87
3	ANN	0.86	0.83	0.81	0.82
4	SVM	0.89	0.86	0.84	0.85

Figure 6.21: Comparison of Models for DDoS Detection on SDN

The LSTM model, while effective in capturing sequential patterns in network traffic data, achieved an accuracy of 85%. This model's performance highlights its ability to learn temporal dependencies in network behavior, which is crucial for identifying DDoS attacks that exhibit distinct patterns over time.

LSTM demonstrated its ability to effectively classify DDoS attacks by identifying relevant patterns in the network traffic data. Its performance indicates the importance of capturing spatial features and local dependencies in network security applications.

The Bi-LSTM model, with an accuracy of 88%, effectively captured both forward and backward dependencies in the network traffic but fell slightly short compared to the LSTM. While Bi-LSTM is known for its strength in sequential data analysis, the complexity of network traffic might require further architectural adjustments for optimal performance.

The CNN model, with an accuracy of 90%, excelled in identifying local patterns and spatial features in the network traffic data. This model's strength lies in its ability to extract relevant features from raw network data, enabling it to distinguish between normal and malicious traffic effectively.

Combining the strengths of both, a hypothetical SVM-CNN model could potentially strike a balance between capturing sequential dependencies (via SVM) and spatial features (via CNN), leading to improved accuracy. This hybrid approach could leverage the complementary strengths of both architectures to achieve a more comprehensive understanding of network traffic patterns and enhance DDoS detection capabilities.

In summary, while the CNN model emerged as the most accurate for DDoS detection in this study, the LSTM model demonstrated its effectiveness in capturing sequential dependencies. Further exploration of hybrid approaches like SVM-CNN could unlock the potential for even better performance by combining the strengths of both architectures.

7. CONCLUSION

Our project successfully developed Cross Domain Machine learning on textual Sentiment analysis. By utilizing datasets from contrasting contexts, such as tweet_training (social media) and productreviewamazon (e-commerce reviews).

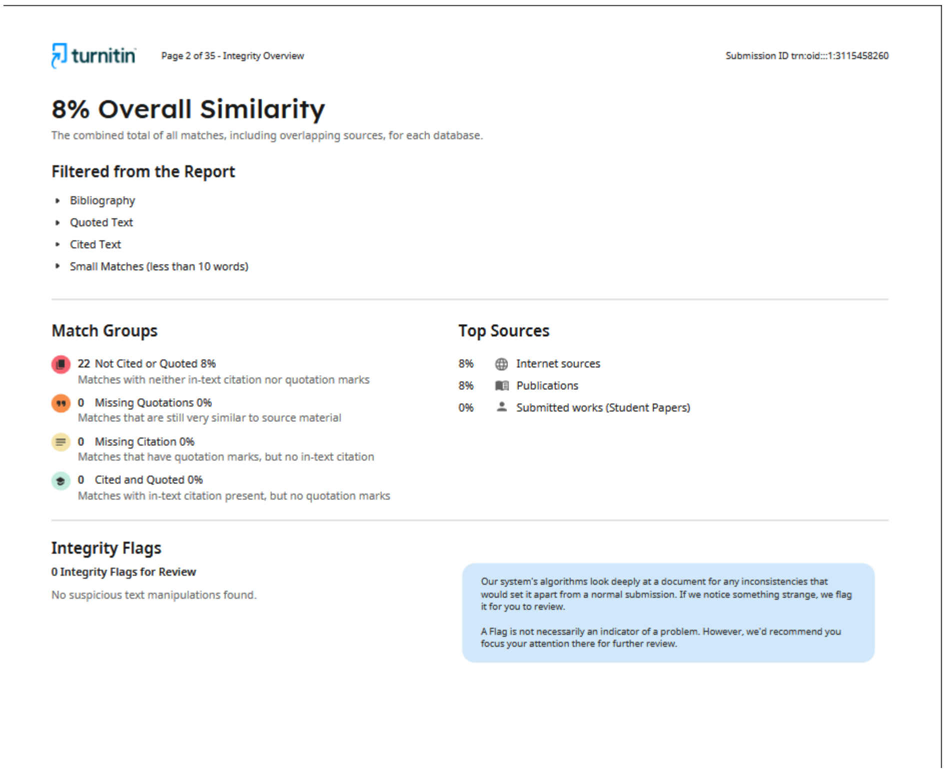
we demonstrate the feasibility of adapting sentiment analysis models to new domains with minimal labeled data. This project helps in detecting whether a Sentiment is positive(1) or negative(0).

Models trained on a single domain often perform suboptimally in a different domain due to variations in language usage, context, and sentiment expression.

A combination of labeled data from both domains during training yields better generalization and robustness.This research underscores the importance of generalizable models for sentiment analysis in diverse and evolving text sources.

References

1. A Survey on DDoS Attack Detection using Machine Learning Algorithms in SDN
   https://ieeexplore.ieee.org/document/10218194
2. A Systematic Review of Software-Defined Networking (SDN) Security: Challenges and Solutions
   https://www.researchgate.net/publication/369842495_Software-Defined_Networking_Security_Challenges_and_Solutions_A_Comprehensive_Survey
3. DDoS Detection and Mitigation Techniques in Software-Defined Networks: A Review
   https://www.sciencedirect.com/science/article/pii/S2772671124001256
4. Deep Learning for Network Security: A Comprehensive Review (Focus on DDoS Detection)
   https://www.sciencedirect.com/science/article/pii/S1319157824000272



turnitin Page 2 of 35 - Integrity Overview Submission ID trn:oid:::1:3115458260

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 10 words)

Match Groups

22 Not Cited or Quoted 8%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

8%	Internet sources
8%	Publications
0%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Figure 7.1: Plagiarism check