p-ISSN: XXXX-YYYY
e-ISSN: XXXX-YYYY

Convolutional Neural Network (CNN) for Attack Detection Based on Nginx Log Analysis

Muhammad Haikal Fikri As'ad[a1], Nadine Fatimah Az Zahra[a2], Maria Angela Permatasari[a3]

[a123]Faculty of Informatics, Telkom University Surabaya
Ketintang, Surabaya
[1]haikalasad@student.telkomuniversity.ac.id
[2]nadinefatimahzhr@student.telkomuniversity.ac.id
[3]mariaangelaprmtsari@student.telkomuniversity.ac.id

Abstract

The rapid evolution of web technologies has significantly increased the frequency and complexity of cyberattacks, making robust security systems a critical necessity. Web servers such as Nginx, which store crucial log data, are often targeted by sophisticated attacks like SQL Injection, XSS, and DDoS. This study focuses on leveraging Convolutional Neural Networks (CNN) for detecting attacks by analyzing Nginx logs. A dataset containing approximately 11 million log entries was preprocessed to extract meaningful features, including URL patterns, HTTP methods, and user-agent strings. To address class imbalance, oversampling and undersampling techniques were employed, ensuring equitable representation of attack types. The CNN model was trained using these processed logs and evaluated using precision, recall, and F1-score metrics. Experimental results demonstrated the model's effectiveness, achieving an accuracy of 81.27%. The study highlights CNN's capability to extract hierarchical features from log data and detect complex attack patterns more effectively than traditional intrusion detection systems. These findings contribute to the development of intelligent, scalable, and adaptive intrusion detection systems, laying a foundation for enhancing cybersecurity in real-world scenarios.

Keywords: *Convolutional Neural Network (CNN), Cybersecurity, Intrusion Detection, Nginx Logs, Attack Detection*

1. Introduction

The rapid evolution of web technologies has significantly enhanced communication, commerce, and education. However, the increasing reliance on web applications has also exposed them to diverse cybersecurity threats. Web applications are prime targets for malicious actors, deploying sophisticated attacks such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Command Injection (CMDi), and Path Traversal, which can lead to unauthorized data access, theft, or server compromise. These security challenges emphasize the critical need for robust intrusion detection systems (IDS) [1].

One particularly vulnerable area is web servers like Nginx, which generate log files critical for analyzing network activities. Server log analysis plays an essential role in identifying and mitigating potential cyberattacks [2], [3]. Traditional rule-based IDS, which rely on predefined signatures to detect known threats, often fail to address novel and evolving attack patterns, necessitating frequent updates and incurring significant overhead [4], [5].

To overcome these limitations, machine learning (ML) and deep learning (DL) techniques have emerged as promising alternatives. Among DL approaches, Convolutional Neural Networks (CNNs) stand out for their ability to analyze structured and unstructured data, such as server logs, and detect complex cyberattack patterns [6], [7]. CNNs excel at extracting hierarchical and discriminative features, making them particularly suitable for identifying anomalies indicative of potential threats [8], [9].

Several studies have demonstrated the effectiveness of CNN-based intrusion detection systems. For instance, CNNs have been shown to outperform traditional methods in detecting Distributed Denial of Service (DDoS), SQL Injection, and XSS attacks through log analysis, achieving detection accuracies exceeding 80% [7], [10]. Additionally, integrating CNNs with advanced preprocessing techniques, such as tokenization and embedding, further enhances detection accuracy and adaptability [11], [12]. Recent advancements have also explored hybrid architectures, such as combining CNNs with LSTM or autoencoders, to address multi-dimensional attack patterns [13], [14].

This study focuses on leveraging CNNs for detecting web application attacks using Nginx logs. By preprocessing log data, extracting meaningful features, and addressing class imbalance issues, this research aims to evaluate the efficacy of CNNs in classifying log entries as normal or malicious. The objectives include enhancing recall for underrepresented attack types and providing practical recommendations for deploying CNN-based intrusion detection systems in real-world scenarios. Building upon recent advancements in deep learning and log analysis, this research aspires to contribute significantly to the field of cybersecurity by offering scalable, efficient, and accurate solutions for intrusion detection [15], [16].

2. Research Methods

2.1. Dataset

The dataset used in this study is a raw web log file named old.log, containing unprocessed Nginx server logs. These logs provide detailed information about server requests, including IP addresses, timestamps, HTTP methods, requested URLs, status codes, and user agents. The dataset consists of approximately 11 million entries, representing a comprehensive collection of server activity. Below are examples of log entries from the dataset:

Table 1. Old.log Example

Old.log Example
66.249.79.63 - - [16/Dec/2022:17:04:10 +0700] "GET /robots.txt HTTP/1.0" 404 146 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
10.220.34.198 - - [16/Dec/2022:17:04:12 +0700] "GET / HTTP/1.0" 200 138 "-" "Mozilla/5.0 (compatible; PRTG Network Monitor (www.paessler.com); Windows)"[i]
216.118.234.34 - - [24/Dec/2022:13:17:34 +0700] "GET /user.php?act=login HTTP/1.0" 404 146 "45ea207d7a2b68c49582d2d22adf953aads|a:2:{s:3:\x22num\x22;s:297:\x22*/SELECT 1,0x2d312720554e494f4e2f2a,2,4,5,6,7,8,0x7b24617364275D3B61737365727428626126659556 30704F79412F506963702729293B2F2F7D787878,10-- -\x22;s:2:\x22id\x22;s:11:\x22-1' UNION/*\x22;}45ea207d7a2b68c49582d2d22adf953a" "Mozilla/5.0 (Windows; U; Windows NT 5.1; pt-BR; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13"

These logs were collected from an active web server and represent a mix of normal access and potentially malicious requests. The raw format of the logs allows for comprehensive preprocessing to extract meaningful features and label data for training the detection model.

2.2. Type Of Research

This research employs an experimental methodology to evaluate the efficacy of the CNN model in detecting web attacks based on Nginx log analysis. The approach involves preprocessing raw server logs, extracting relevant features, applying machine learning techniques for classification, and validating the model's performance using established metrics. This type of research allows for controlled testing of hypotheses and provides quantitative insights into model effectiveness.

2.3. Research Design

The research design comprises the following steps, each illustrated in the diagram:

p-ISSN: XXXX-YYYY
e-ISSN: XXXX-YYYY

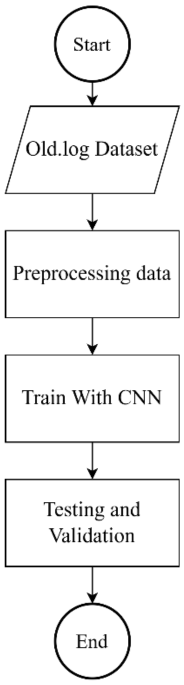


Figure 1. Research Method

A. Dataset

The raw Nginx log file (old.log) containing approximately 11 million entries is collected and prepared as the primary data source. This file includes details like IP addresses, timestamps, HTTP methods, requested URLs, and user agents.

B. Preprocessing Data

- **Regex for Labelling:** Specific attack patterns were identified and labeled using the following regular expressions:

Table 2. Regex Patterns

Type of Attack	Attack Patterns
Sql Injection	(UNION|SELECT|INSERT|DELETE|UPDATE|DROP|AND|OR|CONCAT)(?=.*[=;\'"])
XSS	<script>|%3Cscript%3E
Directory Traversal	\.\./
Malicious URL	Slot|slot|Zeus|zeus|Jackpot|JP|WD|JOIN
Brute Force	Repeated failed attempts (e.g., HTTP status codes 401 or 403) from the same IP within a short timeframe.

- **Feature Extraction:** To extract meaningful attributes from the raw logs and prepare the dataset for training, various features were derived. The length of each URL in the logs was calculated to provide insights into potential malicious patterns, such as excessively long URLs that might indicate SQL injection attempts. The number of parameters in the URL was determined by counting occurrences of '?' and '&', while the length of the user-agent string was computed to identify unusual or suspicious behaviors. HTTP methods, such as GET and POST, were one-hot encoded to represent their significance in training the model. Status codes were categorized based on their hundreds digit (e.g., 2xx, 4xx) to indicate successful or failed requests, with one-hot encoding applied to these categories. The frequency of IP addresses in the logs was calculated to identify patterns of repeated access, potentially indicating brute-force attempts. The timestamp was parsed to extract the hour of access, helping to identify abnormal access patterns. Finally, labels representing attack types or normal activity were encoded using a label encoder to prepare them for classification. The extracted features were stored in a structured CSV format to facilitate efficient loading and analysis, while irrelevant columns such as the original URL, user-agent string, HTTP method, and raw timestamps were dropped to streamline the dataset. Columns irrelevant to the model, such as the original URL, user-agent string, HTTP method, and raw timestamps, were dropped to streamline the dataset.

- **Data Resampling:** To address the issue of class imbalance, oversampling and undersampling techniques are applied. Minority class samples, which represent rare attack cases, are duplicated using the RandomOverSampler method to ensure that they are adequately represented in the training set. Conversely, majority class samples, which typically consist of normal log entries, are reduced using the RandomUnderSampler technique to achieve proportionality while maintaining data diversity. After resampling, the dataset is divided into 80% training data and 20% testing data. This split ensures that the model is trained on a large portion of the dataset while retaining a separate, unbiased subset for performance evaluation.

C. Train

The processed dataset is input into the Convolutional Neural Network (CNN). Key steps include:

- Building the CNN architecture with convolutional layers, batch normalization, dropout, and dense layers.
- Training the model using the Adam optimizer and categorical cross-entropy loss function over multiple epochs.
- Monitoring training progress with metrics like loss and accuracy.

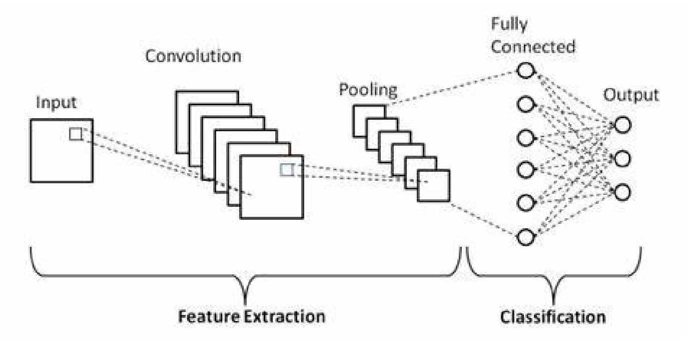


Figure 2. CNN Architecture

D. Testing and Validation

After training, the model is evaluated using:

- Confusion Matrix: A visual representation of true positives, false positives, true negatives, and false negatives.

p-ISSN: XXXX-YYYY
e-ISSN: XXXX-YYYY

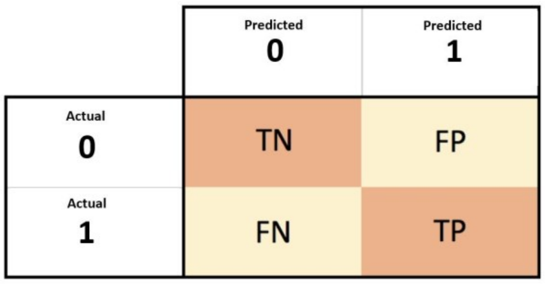


Figure 3. Confusion Matrix

- Evaluation Metrics: The evaluation of the CNN model is based on four key metrics: Precision, Recall, F1-Score, and Accuracy. These metrics provide a comprehensive assessment of the model's classification performance.

Precision:
Precision measures the proportion of correctly identified positive samples (true positives) out of all predicted positives. It reflects the model's ability to avoid false positives.
(1).

$$\Pr e\,cision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP)\ +\ False\ Positives\ (FP)}$$

Recall (Sensivity):
Recall, also known as sensitivity or true positive rate, calculates the proportion of actual positive cases that are correctly identified by the model. It highlights the model's ability to detect all relevant instances.
(2).

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP)\ +\ False\ Negatives\ (FN)}$$

F1-Score:
The F1-Score is the harmonic mean of precision and recall. It provides a balanced metric, particularly useful when dealing with imbalanced datasets where one class dominates over the other.
(3).

$$F1 - Score = 2\ \times \frac{\Pr e\,cision \times Recall}{\Pr e\,cision\ +\ Recall}$$

Accuracy:
Accuracy calculates the overall correctness of the model by measuring the proportion of correctly classified samples (both positive and negative) to the total number of samples.
(4).

p-ISSN: XXXX-YYYY
e-ISSN: XXXX-YYYY

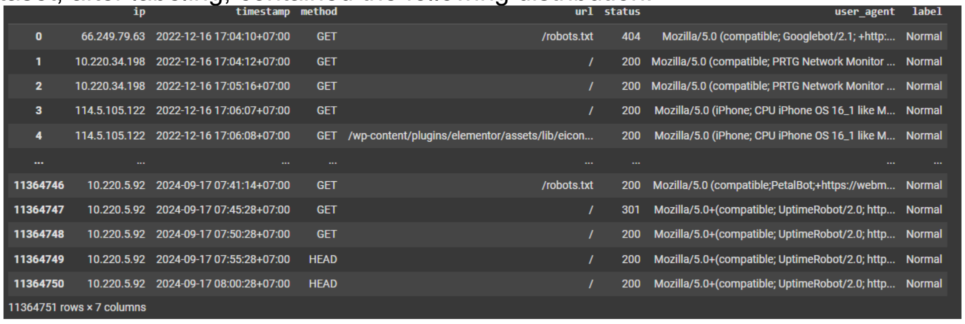
$$Accuracy = \frac{True\ Positives\ (TP)\ +\ True\ Negatives\ (TN)}{Total\ Samples\ (TP + TN + FP + FN)}$$

3. Result and Discussion

This section contains the result and discussion of the research and can be presented as description, charts or figures.

3.1. Data Characteristics

The dataset, after labeling, contained the following distribution:



	ip	timestamp	method	url	status	user_agent	label
0	66.249.79.63	2022-12-16 17:04:10+07:00	GET	/robots.txt	404	Mozilla/5.0 (compatible; Googlebot/2.1; +http:...	Normal
1	10.220.34.198	2022-12-16 17:04:12+07:00	GET	/	200	Mozilla/5.0 (compatible; PRTG Network Monitor ...	Normal
2	10.220.34.198	2022-12-16 17:05:16+07:00	GET	/	200	Mozilla/5.0 (compatible; PRTG Network Monitor ...	Normal
3	114.5.105.122	2022-12-16 17:06:07+07:00	GET	/	200	Mozilla/5.0 (iPhone; CPU iPhone OS 16_1 like M...	Normal
4	114.5.105.122	2022-12-16 17:06:08+07:00	GET	/wp-content/plugins/elementor/assets/lib/eicon...	200	Mozilla/5.0 (iPhone; CPU iPhone OS 16_1 like M...	Normal
...	...	...	...	...	...	...	...
11364746	10.220.5.92	2024-09-17 07:41:14+07:00	GET	/robots.txt	200	Mozilla/5.0 (compatible;PetalBot;+https://webm...	Normal
11364747	10.220.5.92	2024-09-17 07:45:28+07:00	GET	/	301	Mozilla/5.0+(compatible; UptimeRobot/2.0; http...	Normal
11364748	10.220.5.92	2024-09-17 07:50:28+07:00	GET	/	200	Mozilla/5.0+(compatible; UptimeRobot/2.0; http...	Normal
11364749	10.220.5.92	2024-09-17 07:55:28+07:00	HEAD	/	200	Mozilla/5.0+(compatible; UptimeRobot/2.0; http...	Normal
11364750	10.220.5.92	2024-09-17 08:00:28+07:00	HEAD	/	200	Mozilla/5.0+(compatible; UptimeRobot/2.0; http...	Normal

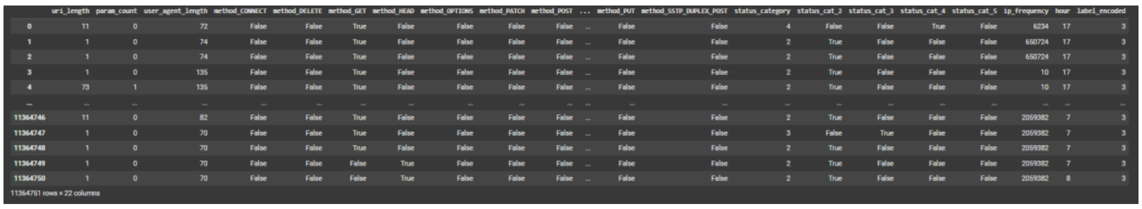
11364751 rows × 7 columns

Figure 4. Post Labeling

Table 3. Labeled Count

Attack Type	Count
Normal	11134440
Malicious URL	219564
Brute Force	5030
Directory Traversal	3426
SQL Injection	1663
XSS	628

During the feature extraction process, various attributes were derived from the raw logs to prepare the dataset for training. These attributes included URI length, parameter count, user-agent length, HTTP methods (one-hot encoded), HTTP status categories, IP frequency, and access time (hour). These features were crucial for distinguishing between normal and malicious log entries.



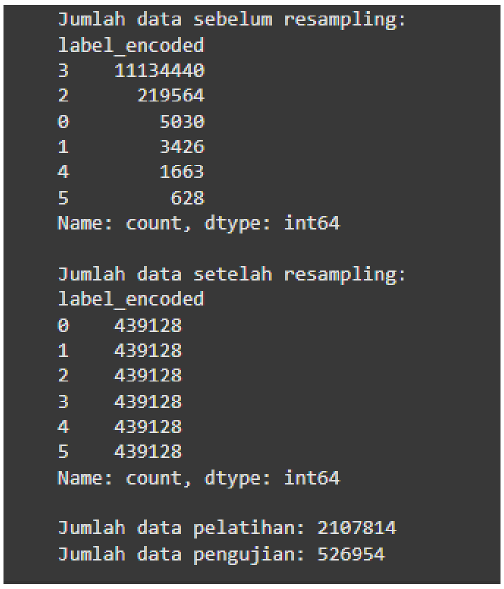
	uri_length	param_count	user_agent_length	method_CONNECT	method_DELETE	method_GET	method_HEAD	method_OPTIONS	method_PATCH	method_POST	...	method_PUT	method_SSTP_DUPLEX_POST	status_category	status_cat_2	status_cat_3	status_cat_4	status_cat_5	ip_frequency	hour	label_encoded
0	11	0	72	False	False	True	False	False	False	False	...	False	False	4	False	False	True	False	6234	17	3
1	1	0	74	False	False	True	False	False	False	False	...	False	False	2	True	False	False	False	650724	17	3
2	1	0	74	False	False	True	False	False	False	False	...	False	False	2	True	False	False	False	650724	17	3
3	1	0	135	False	False	True	False	False	False	False	...	False	False	2	True	False	False	False	10	17	3
4	73	1	135	False	False	True	False	False	False	False	...	False	False	2	True	False	False	False	10	17	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11364746	11	0	82	False	False	True	False	False	False	False	...	False	False	2	True	False	False	False	2059382	7	3
11364747	1	0	70	False	False	True	False	False	False	False	...	False	False	3	False	True	False	False	2059382	7	3
11364748	1	0	70	False	False	True	False	False	False	False	...	False	False	2	True	False	False	False	2059382	7	3
11364749	1	0	70	False	False	False	True	False	False	False	...	False	False	2	True	False	False	False	2059382	7	3
11364750	1	0	70	False	False	False	True	False	False	False	...	False	False	2	True	False	False	False	2059382	8	3

11364751 rows × 22 columns

Figure 4. Features Extraction

The dataset initially exhibited significant class imbalance. The majority class (label 3) contained 11,134,440 entries, whereas the minority classes ranged from 628 to 219,564 entries. To address this imbalance, a combination of undersampling and oversampling techniques was applied. First, the majority class was undersampled to 439,128 entries, ensuring manageable representation. Next, the minority classes were oversampled to match the target count of 439,128 entries per class using the RandomOverSampler technique.

After resampling, all six classes were equally represented, with 439,128 entries each, resulting in a balanced dataset with a total of 2,634,768 entries. The final dataset was then split into training and testing subsets, with 80% (2,107,814 entries) allocated for training and 20% (526,954 entries) for testing.



```
Jumlah data sebelum resampling:
label_encoded
3    11134440
2      219564
0        5030
1        3426
4        1663
5         628
Name: count, dtype: int64

Jumlah data setelah resampling:
label_encoded
0    439128
1    439128
2    439128
3    439128
4    439128
5    439128
Name: count, dtype: int64

Jumlah data pelatihan: 2107814
Jumlah data pengujian: 526954
```

Figure 5. Resampling and split

This preprocessing and balancing process ensured that the model could learn effectively from a balanced dataset, improving its ability to detect all types of attacks while reducing the risk of bias toward the majority class.

3.2. CNN Training and Performances

The CNN model was trained on the processed dataset to classify log entries into six different categories, including both normal and attack classes. Before training, the target labels were one-hot encoded to ensure compatibility with the softmax output layer of the model. The input features were reshaped to include an additional dimension, enabling compatibility with the 1D convolutional layers. The training process utilized the Adam optimizer with categorical cross-entropy as the loss function, reflecting the multi-class nature of the classification task.

The CNN architecture consisted of two convolutional layers with 32 and 64 filters, respectively, each followed by batch normalization and dropout layers to improve generalization and prevent overfitting. These layers were followed by a flattening layer and two dense layers, the last of which applied the softmax activation function to generate class probabilities. The model was trained over five epochs with a batch size of 64, using a training dataset of approximately 2.1 million samples and evaluated on a validation set.

The training results demonstrated a steady improvement in accuracy and loss across epochs. By the end of the fifth epoch, the model achieved a training accuracy of 79.99% and a validation accuracy of 81.27%. Validation loss reduced consistently, reaching 0.4935, indicating effective learning and generalization. Figure 6 illustrates the training and validation accuracy and loss curves over the epochs, showcasing stable convergence.

p-ISSN: XXXX-YYYY
e-ISSN: XXXX-YYYY

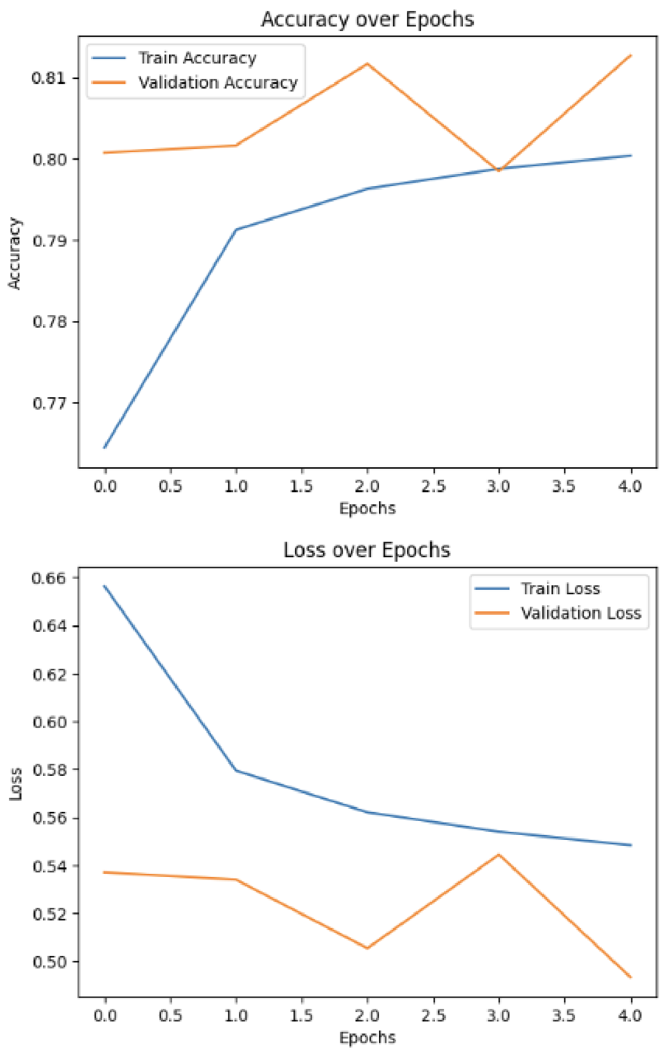


Figure 6. Resampling and split

3.3. Analysis of Results

The CNN model's performance on the testing dataset (526,954 samples) was evaluated using a confusion matrix, classification report, and accuracy score. The confusion matrix (Figure 2) reveals the detailed distribution of true positives, false positives, true negatives, and false negatives across all six classes. The classification report provided precision, recall, and F1-score metrics for each class, highlighting the model's ability to classify log entries effectively.

The model achieved an overall accuracy of **81.27%** on the testing dataset. Precision across classes ranged from 68% to 93%, with class 1 achieving the highest precision. Recall varied between 53% and 98%, with class 5 attaining the highest recall. The weighted average F1-score of 81% indicates a good balance between precision and recall, even in the presence of minor class imbalances.

The confusion matrix indicates areas for improvement. For example, class 4 (malicious URLs) showed lower recall (53%), suggesting the model struggled with detecting certain patterns in this class. However, the high recall for class 5 (98%) reflects the model's strength in identifying these cases accurately. These results suggest the potential for further enhancements, such as fine-tuning the model architecture or incorporating additional features.

Figures 7 and 8 visually present the training dynamics and confusion matrix, respectively, providing deeper insights into the model's learning process and classification performance.

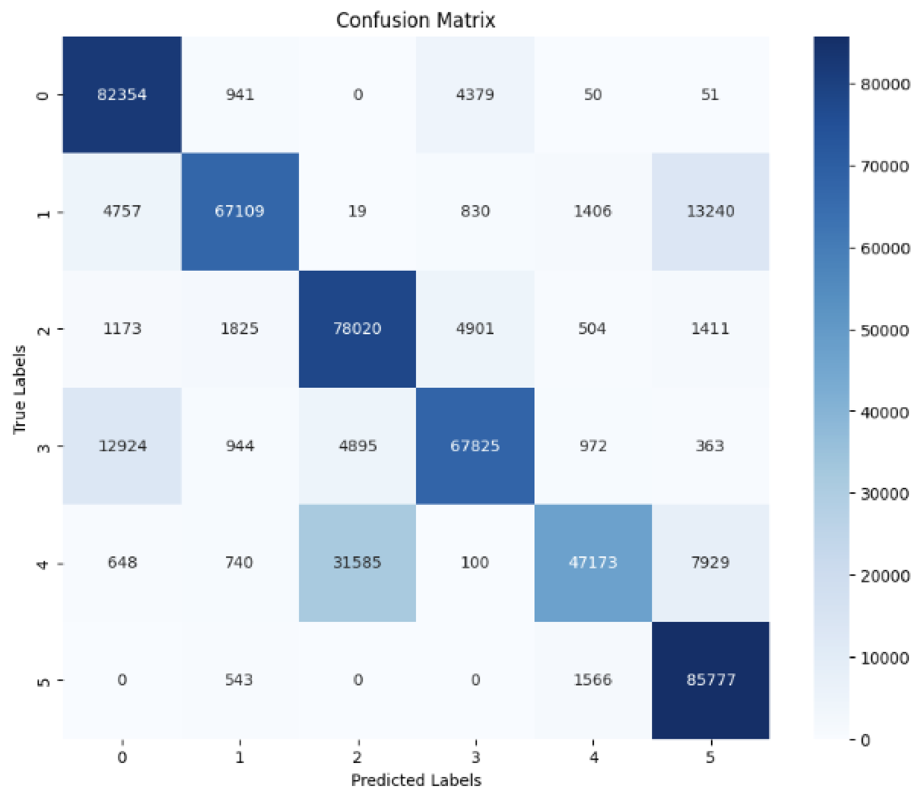
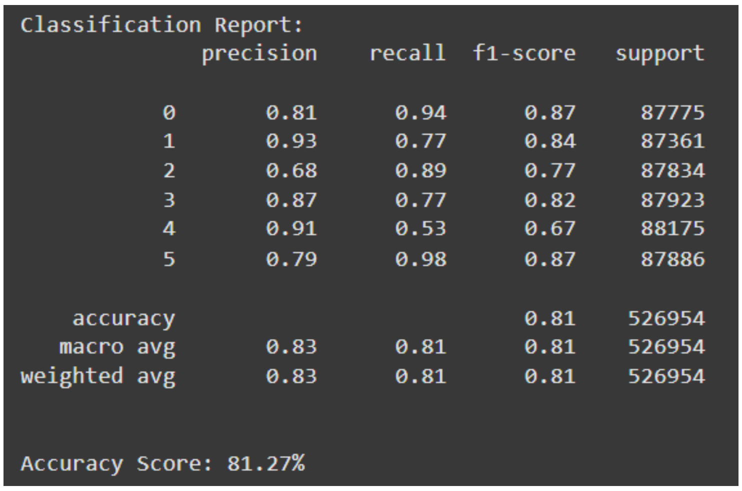


Figure 7. Confusion Matrix



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

           0       0.81      0.94      0.87     87775
           1       0.93      0.77      0.84     87361
           2       0.68      0.89      0.77     87834
           3       0.87      0.77      0.82     87923
           4       0.91      0.53      0.67     88175
           5       0.79      0.98      0.87     87886

    accuracy                           0.81    526954
   macro avg       0.83      0.81      0.81    526954
weighted avg       0.83      0.81      0.81    526954


Accuracy Score: 81.27%
```

Figure 8. Evaluation Metric

4. Conclusion

This study demonstrated the effectiveness of a Convolutional Neural Network (CNN) in detecting web attacks based on Nginx log analysis. By preprocessing raw server logs, deriving meaningful features, and applying resampling techniques, the dataset was balanced to ensure equal representation of normal and attack classes. The CNN model, trained on over two million samples and evaluated on a separate testing dataset of 526,954 samples, achieved an accuracy of **81.27%**.

The classification report highlighted the model's strengths, with precision ranging from 68% to 93% and recall between 53% and 98%. High F1-scores for classes such as normal entries (0.87), SQL Injection (0.84), and XSS (0.77) demonstrated the model's capability to detect complex attack patterns effectively. However, lower recall for certain classes, such as malicious URLs (53%), indicated areas where the model struggled to identify specific attack patterns.

Despite these limitations, the confusion matrix and classification metrics revealed the CNN's ability to balance precision and recall across most classes, achieving a macro-average precision, recall, and F1-score of 83%, 81%, and 81%, respectively. The results suggest that the model can serve as a reliable tool for identifying web attacks while highlighting opportunities for improvement.

Future work could focus on enhancing the model's recall for specific attack types by refining feature engineering processes or employing advanced architectures such as CNN-LSTM hybrids. Additionally, testing the model's performance in real-time scenarios on live server environments would provide valuable insights into its practical applicability. This research establishes a foundation for improving intrusion detection systems through deep learning and log-based analysis.

This study demonstrated the feasibility of using CNN for web attack detection based on Nginx logs. With an accuracy of 81.27%, the model effectively identified attack patterns and offered balanced performance across key evaluation metrics. Future enhancements in feature engineering and model architecture could further improve its efficacy, making it a robust solution for modern intrusion detection systems.

References

[1] N. N. Hoa and D. P. Van On, “ENHANCING WEBSHELL DETECTION WITH DEEP LEARNING-POWERED METHODS”.

[2] W. Li, B. Zhang, and J. Zhang, “Malicious Requests Detection with Improved Bidirectional Long Short-term Memory Neural Networks,” *arXiv preprint arXiv:2010.13285*, 2020.

[3] S. Wang, R. Jiang, Z. Wang, and Y. Zhou, “Deep learning-based anomaly detection and log analysis for computer networks,” *arXiv preprint arXiv:2407.05639*, 2024.

[4] T. Alma and M. L. Das, “Web application attack detection using deep learning,” *arXiv preprint arXiv:2011.03181*, 2020.

[5] C.-N. Nhu and M. Park, “Two-phase deep learning-based edos detection system,” *Applied Sciences*, vol. 11, no. 21, p. 10249, 2021.

[6] S. Jacob, Y. Qiao, Y. Ye, and B. Lee, “Anomalous distributed traffic: Detecting cyber security attacks amongst microservices using graph convolutional networks,” *Comput Secur*, vol. 118, p. 102728, 2022.

[7] R. A. Widodo, M. K. Delimayanti, and A. Wulandari, “DDoS Attacks Detection With Deep Learning Approach Using Convolutional Neural Network,” *Journal of Applied Informatics and Computing*, vol. 8, no. 2, pp. 235–240, 2024.

[8] P. Balasubramanian, S. Nazari, D. K. Kholgh, A. Mahmoodi, J. Seby, and P. Kostakos, “TSTEM: A Cognitive Platform for Collecting Cyber Threat Intelligence in the Wild,” *arXiv preprint arXiv:2402.09973*, 2024.

[9] H. P. Pham, V. H. Nguyen, N. A. Tran, and M. H. Nguyen, “Log Analysis For Network Attack Detection Using Deep Learning Models,” in *Proceedings of the 12th International Symposium on Information and Communication Technology*, 2023, pp. 731–738.

[10] M. A. Ridho and M. Arman, “Analisis Serangan DDoS Menggunakan Metode Jaringan Saraf Tiruan,” *Jurnal Sisfokom (Sistem Informasi Dan Komputer)*, vol. 9, no. 3, pp. 373–379, 2020.

[11] H. Liu and P. Patras, “NetSentry: A deep learning approach to detecting incipient large-scale network attacks,” *Comput Commun*, vol. 191, pp. 119–132, 2022.

[12] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, “Network traffic anomaly detection using recurrent neural networks,” *arXiv preprint arXiv:1803.10769*, 2018.

[13] X. Wang, T. Hui, L. Zhao, and Y. Cheng, “Input-Driven Dynamic Program Debloating for Code-Reuse Attack Mitigation,” in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, pp. 934–946.

[14] T. Ahmad, D. Truscan, J. Vain, and I. Porres, “Early detection of network attacks using deep learning,” in *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, IEEE, 2022, pp. 30–39.

[15] H. Liang *et al.*, “Generative pre-trained transformer-based reinforcement learning for testing web application firewalls,” *IEEE Trans Dependable Secure Comput*, vol. 21, no. 1, pp. 309–324, 2023.

[16] P. Wu and H. Guo, “LuNET: a deep neural network for network intrusion detection,” in *2019 IEEE symposium series on computational intelligence (SSCI)*, IEEE, 2019, pp. 617–624.