**Original Research Article**

**Brute Force Attack Detection Using Deep Learning Model**

**ABSTRACT**

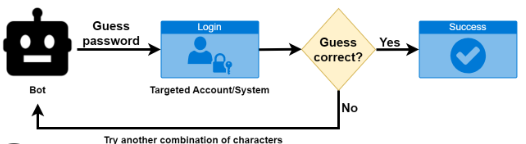
|  |
| --- |
| **Introduction:** This study presents a deep learning-based approach for detecting brute force attacks in network traffic using a Convolutional Neural Network (CNN) model.  **Methodology:** Flow-based data from the NF-UQ-NIDS dataset was preprocessed and balanced using the Downsampling and Synthetic Minority Over-sampling Technique (SMOTE) techniques. The CNN architecture was designed to extract temporal and spatial features from the input data, enabling accurate binary classification between brute force attacks and benign traffic.  **Results:** Evaluation of the model using standard performance metrics including accuracy, precision, recall, F1-score, and AUROC, revealed exceptional results from the CNN-SMOTE configuration achieving an accuracy of 99.82% and a recall of 99.95%. Comparative analysis against benchmark models from previous studies confirmed the superiority of the proposed approach, particularly in handling class imbalance.  **Conclusion:** The results demonstrate that deep learning models, especially when trained with the appropriate data balancing technique, can significantly enhance intrusion detection systems. Recommendations for further improvement include exploring hybrid models and integrating explainable AI components. |

*Keywords: Deep learning, Brute force attack, Convolutional Neural Network, Network intrusion detection, Flow-based traffic analysis, SMOTE, Downsampling, Cybersecurity.*

# INTRODUCTION



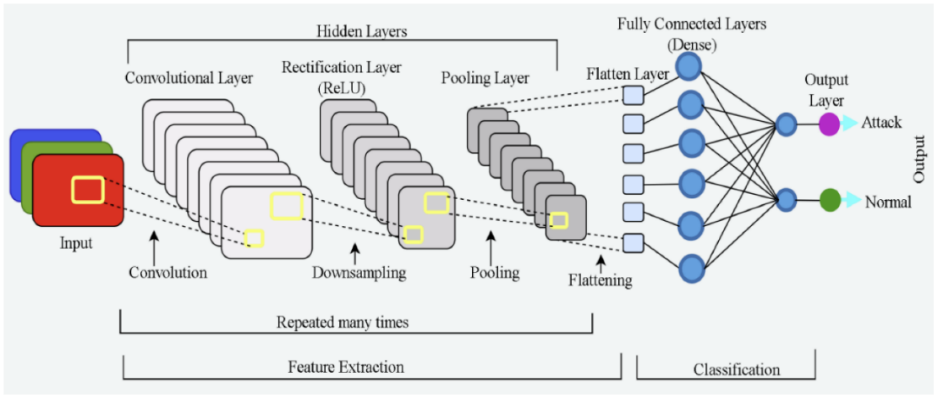
The swift growth of cyber-attacks has spontaneously amplified the need for enhanced model for intrusion detection system proficient of efficiently classifying and mitigating several forms of brute force attacks (Agghey et al., 2021). Brute force attack being a prominent cyber-attack is an automated in which attackers attempt to gain unauthorized access to systems by applying technical measures and utilization of salient encryption key combinations (Sulochana, 2019; Vijayan & Sundar, 2024). These attacks pose a significant threat as they can take advantage of weak or recycled passwords, often remaining unnoticed until substantial harm is caused. With the increasing shift of operations to online platforms, protecting against brute force attacks has become a critical focus for cybersecurity experts.

  
Figure 1: Brute Force Repetitive Architect (Ali Hamza & Jumma surayh Al-Janabi, 2024)

Brute force attacks are characterized by their repetitive nature (Ali Hamza & Jumma surayh Al-Janabi, 2024; Sarkunavathi & Srinivasan, 2018). Attackers typically make repeated login attempts within a short timeframe. Detecting these attacks manually leads to time wastage and unproductive results, particularly, when it involves high volumes of data. This indicates a growing need for automated detection methods that can correctly identify such attacks with timely results (Yashaswi et al., 2023), minimizing the risks associated with them.

Moden growth in deep learning have explore novel avenues for detecting security threats, including brute force attacks (Alotibi et al., n.d.; Zhang et al., 2020). Traditional detection techniques, such as rule-based systems or simple anomaly detection, have great limitation due to their inability to scale or adapt to new attack strategies. However, deep learning models, particularly Convolutional Neural Networks (CNNs), have shown great potential in handling sequential data, such as login attempts, due to their ability to capture temporal and spatial patterns within the data.

Figure 2 offers a structured breakdown of a Convolutional Neural Network (CNN), highlighting its key components and workflow with clear distinction between feature extraction through convolutional, ReLU, pooling layers, and classification which is via dense and output layers (Pujari et al., 2025). The repetition of convolutional and pooling operations underscores the hierarchical learning process, where the network progressively extracts complex features from raw input data.

  
Figure 2: Learning Architecture of CNN (Gebresilassie et al., 2023)

The inclusion of terms like "Attack" and "Normal" suggests that a likelihood of exploring CNNs in the context of detecting attack. The visual simplicity effectively conveys CNN’s two-phase paradigm which are, transforming input data into actionable insights through layered abstraction (Nzenwata et al., 2024).

The use of deep learning for brute force attack detection is an emerging area of research. By leveraging large datasets of authentication logs, deep learning models can learn complex patterns and identify brute force attempts with high accuracy. These models can be trained to distinguish between normal login behavior and suspicious patterns indicative of a brute force attack.

Furthermore, the ability of deep learning models to process large amounts of data in real-time makes them an ideal candidate for deployment in live systems, where they can continuously monitor login attempts and trigger alerts when an attack is detected.

This study aims to explore the feasibility and effectiveness of using a Convolutional Neural Network (CNN) to detect brute force attacks from authentication logs. By focusing on sequential data such as login attempts, time between attempts, and IP addresses, the model will be trained to classify each login attempt as either "normal" or "brute force." The goal is to develop a deep learning-based system capable of automatically identifying brute force attacks, providing an efficient solution for enhancing cybersecurity defenses.

# METHODOLOGY

This study employs a deep learning-based approach for detecting brute force attacks through the analysis of authentication log data. The methodology is divided into several key stages: data collection, data preprocessing, feature engineering, model design, training and validation, and performance evaluation.



## Data Collection

The training and testing phases of this study were based on the NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems (NF-UQ-NIDS) (Sarhan et al., 2021). This dataset was selected due to its comprehensive collection of labeled NetFlow records simulating both benign and malicious traffic, including brute force attacks. It includes flow-based network traffic data captured in a controlled environment, providing an ideal foundation for training intrusion detection models.

## Model Development

The model for brute force attack detection will be developed using the Python programming language, leveraging widely used deep learning libraries such as TensorFlow, Keras, and NumPy. The extensive ecosystem of python programming language provides robust support for data preprocessing, model training, evaluation, and deployment.

A Convolutional Neural Network (CNN) architecture is selected due to its ability to automatically learn complex patterns from sequential data (Jyothi et al., 2022). CNNs are particularly effective in recognizing local temporal patterns in network traffic data, which is crucial in identifying brute force attack behaviors such as repeated login attempts within short time intervals.

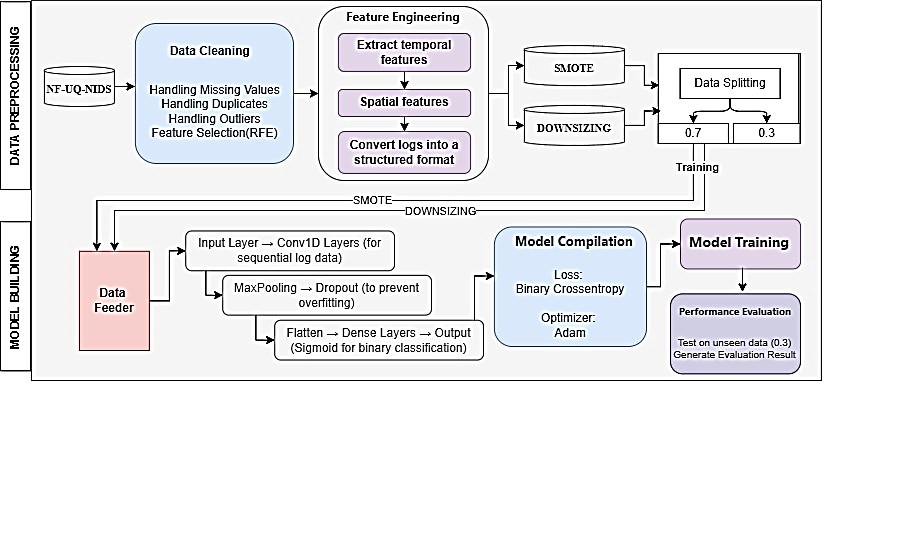


Figure 3: Architecture of the CNN Model

The CNN model illustrated in Figure 3 will be trained using the NF-UQ-NIDS dataset, which includes labeled flow-based network traffic data, allowing the model to distinguish between normal and brute force activities. The dataset will first undergo preprocessing and feature extraction, and the resulting structured input will be fed into the CNN.

* + 1. **Data Preprocessing**

The data preprocessing stage is significant for preparing the dataset before training the deep learning model. In this study, the raw data was sourced from the NF-UQ-NIDS dataset, which comprises labeled NetFlow traffic records representing both benign and malicious network activities, including brute force attacks. The preprocessing phase began with data cleaning, where missing values were handled, duplicate entries were removed, and outliers were identified and treated.

Important features were selected using Recursive Feature Elimination (RFE) to improve model efficiency and reduce noise (Jeon & Oh, 2020). Following the cleaning phase, feature engineering was performed to extract meaningful patterns from the data. This involved the generation of temporal features, such as the number of login attempts within specific time intervals, and spatial features, including patterns in IP address or port usage. These features helped to convert the raw network logs into a structured format suitable for input into the deep learning model.

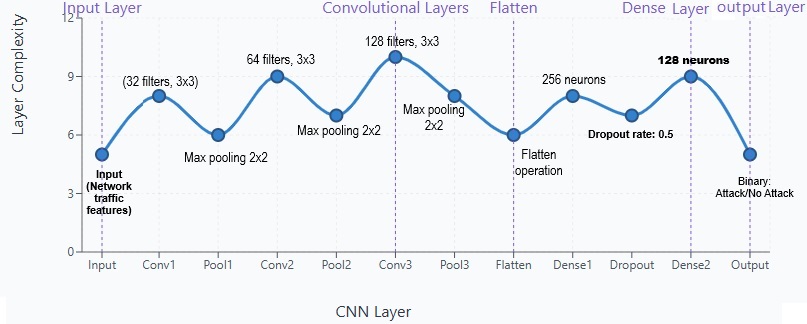
  
Figure 4: Class Distribution Across Sampling Techniques

To address the issue of class imbalance, two data balancing techniques were employed, the SMOTE (Synthetic Minority Over-sampling Technique) and downsampling of the majority class. The default sample size of the dataset includes 291,955 Brute Force samples and 9,208,048 Benign samples. For this research study, 583,910 Benign samples which is double of the size of the Brute Force samples will be used. Synthetic samples will then be generated to complement the SMOTE balancing technique, as illustrated in Figure 4.

The last stage of the data preprocessing is the splitting of the dataset for training (70%) and testing (30%) subsets to facilitate model training and performance evaluation.

* + 1. **Model Building**

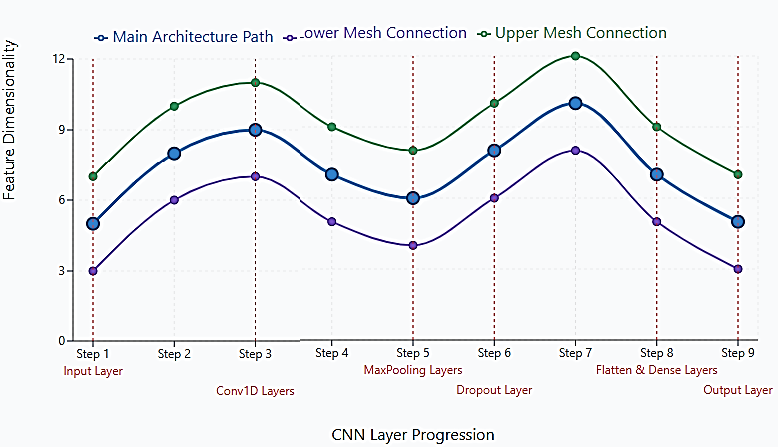
The model will be trained over multiple epochs. Early stopping will be applied to avoid overfitting if validation performance no longer improves. Binary cross-entropy will be used as the loss function, while the Adam optimizer will be employed for efficient gradient-based optimization.

  
Figure 5: Models’ Convolutional Architecture

The convolutional layers (Conv1D) in the CNN model visualized in Figure 5 and 6 play a key role in identifying patterns that indicate brute force attacks. The 1D convolution operation is illustrated below;

Where: yi is the output feature map, wj are the weights of the filter, x{i+j} are the input values  
b is presents the bias term, f is represents activation function (ReLU), k is the kernel size.

These layers scan through sequences of network flow features to detect repetitive behaviors, such as multiple failed login attempts within a short timeframe or repeated access from the same IP address.

  
Figure 6: Models’ Mesh visualization

In the context of network logs, spatial features refer to how different attributes (like time, IP, and port numbers) interact with one another. The CNN captures localized patterns across these features, which may not be obvious in raw form. For example, an increase in the number of attempts to login from a specific source combined with repeated port access can be a strong signal of malicious activity.

MaxPooling layers reduce the size of the feature maps generated by convolutional layers as expressed in Equation 2, helping the model to give attention on the most essential patterns. This not only speeds up computation but also ensures that the most prominent signs of an attack, such as spikes in traffic or bursts of failed logins, are retained for further analysis in the model’s deeper layers. Where p is the pooling size.

The Dropout layer is introduced to enhance the model’s robustness by preventing overfitting, especially when dealing with imbalanced data where normal behavior dominates. By randomly deactivating a fraction *p* of the neurons during training as expressed in Equation 3, Dropout ensures that the model learns to generalize well and does not rely too heavily on any single pattern or feature.

… (3)

The Flatten and Dense layers integrate the spatial and temporal features extracted earlier and perform the final classification. These fully connected layers allow the model to interpret the learned patterns and decide whether a particular flow indicates normal activity or a brute force attack.

Flattening reshapes the multi-dimensional input into a 1D vector in Equations 4 before the Dense layer applies a linear transformation followed by activation in Equation 5.

… (4)

… (5)

Where:  
W is the weight matrix; x is the input vector; b is the bias; and f is an activation function (ReLU)

The Output layer, using a sigmoid activation function for classification as illustrated in equation 6, provides a binary prediction, making the CNN highly effective for intrusion detection tasks.

Where σ is Sigmoid activation function and z is Output of the final dense layer

Once the model is trained and evaluated, it will be saved in an appropriate format and prepared for integration into a real-time intrusion detection system.

## Performance Evaluation

Evaluating the model's performance is a vital part of the system's development, as it reveals how effectively the model can identify and mitigate brute force attacks. This phase focuses on assessing the classification capabilities of the proposed system through various statistical metrics. These measurements are essential for identifying both the strengths and potential limitations of the model. Several performance indicators will be used, including accuracy, precision, recall, F1 score, and the Receiver Operating Characteristic - Area Under Curve (ROC-AUC). Graphical tools like the confusion matrix, precision-recall curves, and ROC curves will be used to visually represent the model’s performance and improve interpretability.

# RESULTS AND DISCUSION

The results for the CNN model will be expressed using two techniques: downsizing and SMOTE (Synthetic Minority Over-sampling Technique). These approaches were employed to address class imbalance in the dataset and to improve the model's generalization capabilities.

The model consists of three Conv1D layers with increasing filter sizes (32, 64, 128), each followed by max-pooling layers to progressively reduce spatial dimensions. A dropout layer is included for regularization, followed by a flattening layer and three dense layers, culminating in a single output neuron for binary classification. The model architecture is designed to extract hierarchical temporal features from sequential input data, making it well-suited for tasks like time-series analysis or intrusion detection.

Both techniques will be evaluated using standard performance metrics; accuracy, precision, recall, F1-score, and ROC-AUC, to uncover the strengths and limitations of the CNN model in detecting brute force attacks.

## Evaluation with Downsampling

Firstly, downsampling was applied to reduce the number of normal instances, balancing them with the minority class (brute force attacks). The CNN model trained on this balanced dataset achieved respectable scores across all metrics as seen in table 1.

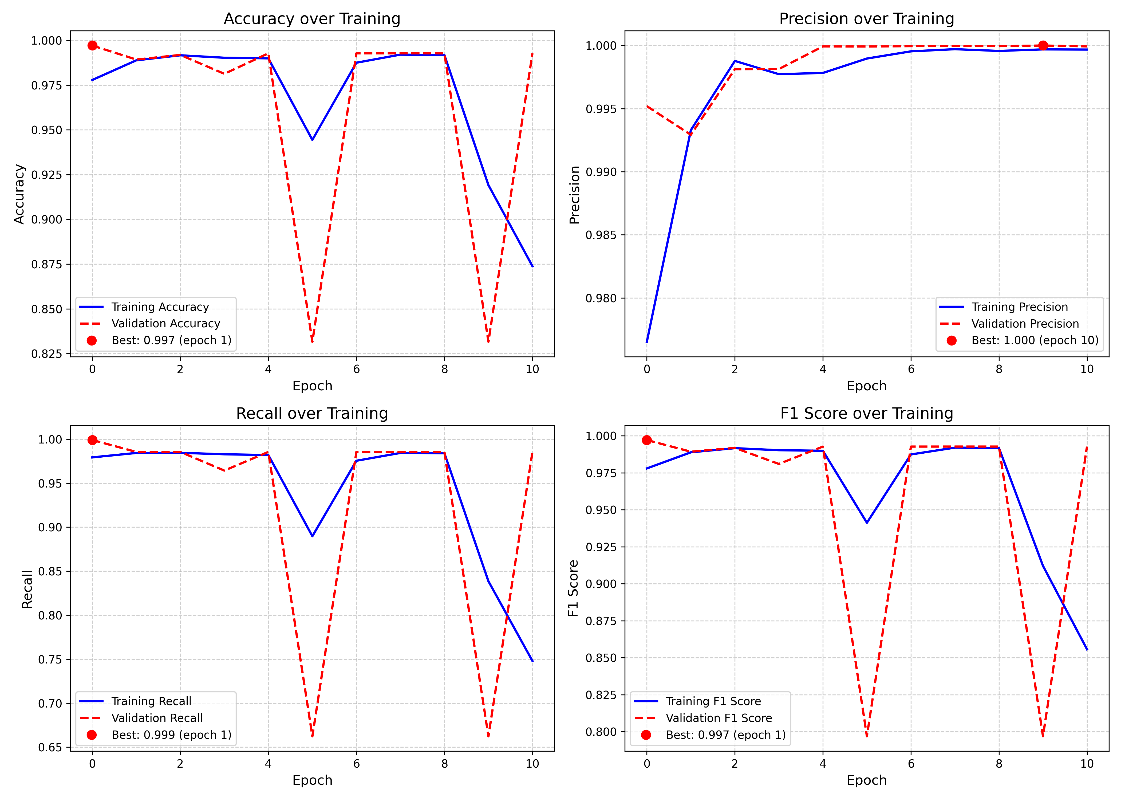
Table 1: CNN Model’s Result in Downsizing Technique

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accuracy (%)** | **Precision (%)** | **Recall (%)** | **F1-Score (%)** | **AUROC** |
| 99.73 | 99.54 | 99.92 | 99.73 | 99.97 |

The CNN model achieved impressive performance using the downsampling technique, with an accuracy of 99.73% and an exceptionally high recall of 99.92%, indicating its strong ability to detect nearly all brute force attacks. The precision of 99.54% suggests that most of the flagged attacks were indeed malicious, while the F1-score of 99.73% reflects a balanced performance between precision and recall. The AUROC value of 99.97% further confirms the model's excellent discriminative capability, making it highly reliable for intrusion detection tasks even under class-balanced conditions.

### Training and Validation Metrics Analysis (Downsizing Technique)

Figure 7 illustrates the performance of the Convolutional Neural Network (CNN) model across 10 training epochs using four key evaluation metrics: Accuracy, Precision, Recall, and F1 Score. Each subplot presents both training and validation trends for the respective metric. The visualization highlights several key observations as list below:

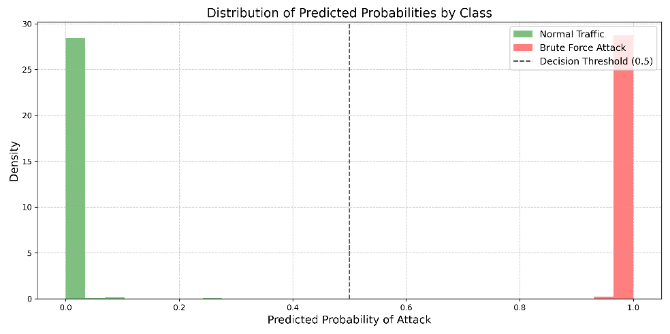
  
Figure 7: Based on the training vs. validation metrics charts

1. Accuracy: The training accuracy remains consistently high, while validation accuracy fluctuates, notably dropping at epochs 6 and 10, suggesting potential overfitting or instability in learning generalization.
2. Precision: Both training and validation precision are remarkably high, with the validation precision peaking at epoch 10 (1.000). This indicates the model is highly effective at minimizing false positives.
3. Recall: Training recall is strong and stable, but validation recall drops significantly at certain epochs (notably epoch 6), hinting at instances where the model may miss actual positive cases during validation.
4. F1 Score: The F1 Score mirrors the trends in precision and recall. Despite fluctuations, the high overall scores reflect the model’s robust balance between precision and recall.

The model demonstrates strong learning capabilities with high metric values. However, the dips in validation metrics, especially recall and accuracy, suggest the need for further regularization, additional data tuning, or more epochs with early stopping to stabilize performance and improve generalization.

### Predicted Probability Distribution Analysis (Downsizing Technique)

The distribution of predicted probabilities by Class of Figure 8 presents a histogram of the CNN model’s predicted probabilities for classifying network traffic as either normal traffic or brute force attack for the downsized technique.

  
Figure 8: Predicted Probability Distribution Analysis (Downsized Technique)

The predicted probabilities are highly polarized, with normal traffic predictions clustered near 0.0 and brute force attack predictions concentrated near 1.0. Using a decision threshold of 0.5, the model confidently distinguishes between the two classes, as most predictions fall well beyond this threshold. The minimal overlap between the distributions indicates excellent model separability and strong discriminatory capability.

This distribution confirms that the CNN model is effectively distinguishing between normal and brute force traffic. The sharp separation of predicted probabilities indicates a high level of certainty in predictions, aligning well with the high precision and recall metrics observed during training and validation.

## Evaluation with SMOTE

To address the class imbalance challenge in brute force attack detection, the Synthetic Minority Oversampling Technique (SMOTE) was applied. This method synthetically generated new brute force samples, enriching the training data and exposing the CNN model to a wider variety of attack patterns. The results from the SMOTE techniques is presented in Table 2.

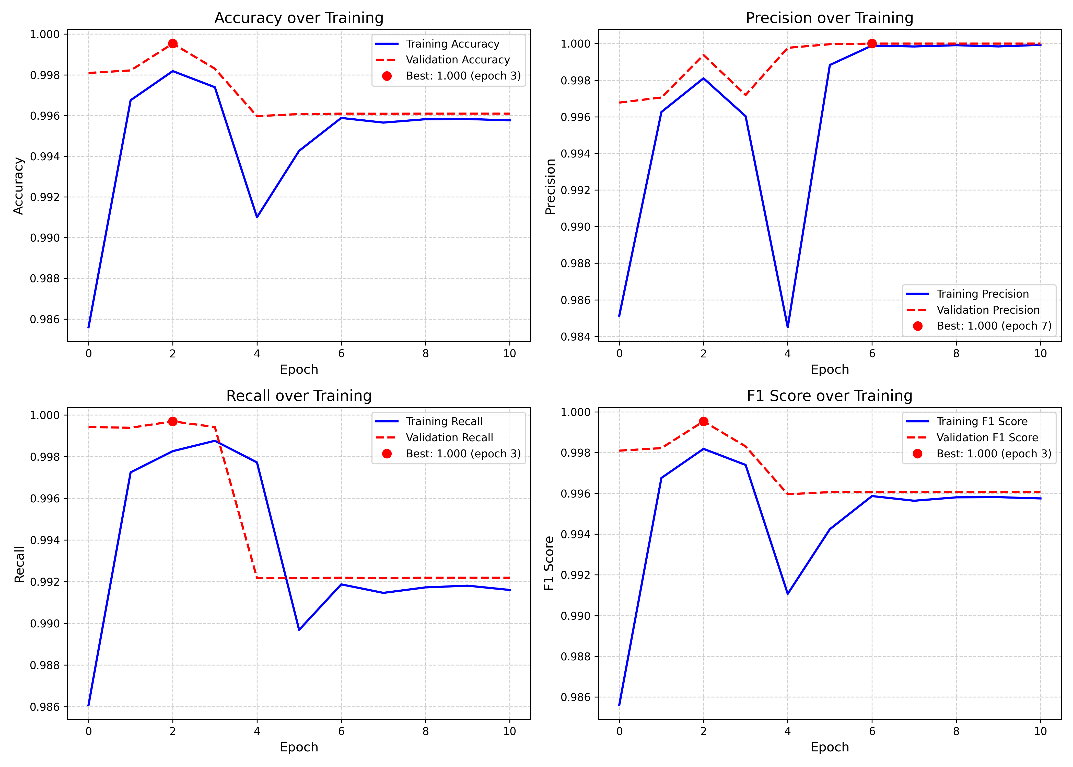
Table 2: CNN Model’s Result in SMOTE Technique

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accuracy (%)** | **Precision (%)** | **Recall (%)** | **F1-Score (%)** | **AUROC** |
| 99.82 | 99.70 | 99.95 | 99.82 | 99.98 |

These results demonstrate the model’s exceptional classification capability with SMOTE, particularly the high recall (99.95%), indicating almost all brute force attacks were correctly detected, and an AUROC of 99.98%, showing excellent separation between classes.

### Training and Validation Metrics Analysis (Downsizing Technique

Figure 9 displays the training and validation performance of the CNN model over 10 epochs using four key metrics: Accuracy, Precision, Recall, and F1 Score. Key finding from the metric analysis are;

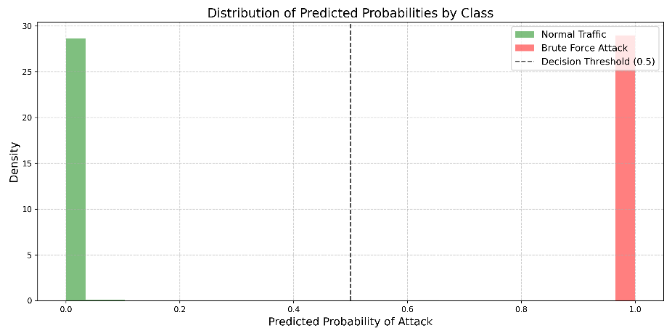
  
Figure 9: Predicted Probability Distribution Analysis (SMOTE Technique)

1. Accuracy: Training accuracy fluctuates significantly, especially around epochs 3 to 6, indicating the model was still adjusting during early training. Validation accuracy remains consistently high, suggesting that the model generalizes well despite training instability.
2. Precision: Validation precision stays near perfect (peaking at 1.000 in epoch 7), showing the model is highly reliable in minimizing false positives. The training precision shows some variance but aligns with validation by later epochs.
3. Recall: Validation recall is consistently high with minimal drop, reflecting the model's excellent sensitivity in identifying brute force attacks. The training recall curve shows slight volatility but stabilizes over time.
4. F1 Score: The F1 score, balancing precision and recall, mirrors both their patterns. While training F1 improves gradually, the validation F1 remains high throughout, confirming robust model performance.

This analysis confirms that the CNN model, trained with SMOTE, not only achieves high accuracy and generalization but also maintains strong predictive balance between false positives and false negatives. It highlights the model’s readiness for deployment in real-world intrusion detection environments.

### Predicted Probability Distribution Analysis (SMOTE Technique)

Figure 10 displays the distribution of predicted probabilities for classifying network traffic as either normal traffic or brute force attacks, generated by the CNN model trained using the SMOTE technique. Key Observation include;

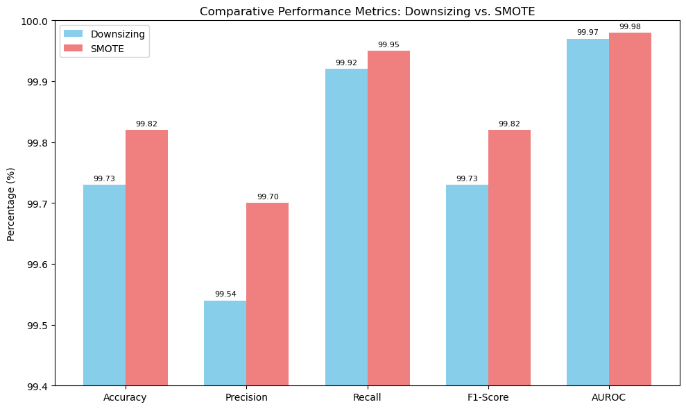
  
Figure 10: Predicted Probability Distribution Analysis (SMOTE Technique)

The predicted probability distribution demonstrates clear class separation, with normal traffic predictions tightly grouped near 0.0, indicating the model's strong confidence in identifying non-attacks, while brute force attack predictions are sharply concentrated near 1.0, reflecting a high certainty in detecting malicious activity. The vertical dashed line at 0.5 marks the classification threshold, and the pronounced polarization of probabilities shows that most predictions fall far from this point, reducing ambiguity in classification. Additionally, the minimal overlap between the two distributions highlights the model's excellent separability and high prediction certainty.

This histogram confirms the CNN model’s outstanding classification capability after applying SMOTE. The distinct clustering of prediction probabilities around the extreme ends (0 and 1) shows that the model has learned to clearly distinguish between attack and non-attack traffic, aligning with the high evaluation metrics. This level of confidence is crucial in intrusion detection scenarios where false negatives or false positives can have significant implications.

## Comparative Performance Metrics

To determine the most effective approach for handling class imbalance, Figure 11 presents a side-by-side comparison of the CNN model's performance using two balancing techniques: Downsizing and SMOTE. The comparison reveals that while both techniques yield exceptional results, SMOTE offers slightly improved metrics across all categories, suggesting enhanced generalization and robustness when synthetic minority samples are introduced.

  
Figure 11: Comparative Performance of Balancing Techniques’

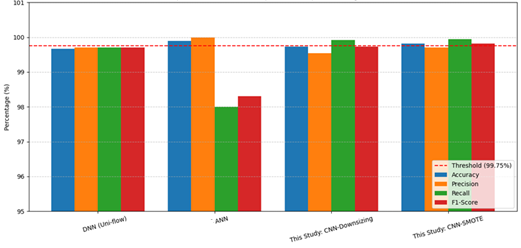
### 4.3.1. Benchmarked Metrics Comparison

To contextualize the performance of the proposed CNN model, Table 4 compares its accuracy against several benchmarked models from prior research. The studies referenced include classical and deep learning-based models such as traditional CNNs, DNNs with bi-directional and uni-directional flow, ANN, and LSTM architectures. Both versions of the CNN model from this study, trained using SMOTE and Downsizing techniques, surpass or closely rival existing models in accuracy. Notably, the CNN with SMOTE achieved 99.82%, outperforming other models, including DNN (Uni-flow) (Otoom et al., 2023) at 99.67% and ANN (Alotibi & Alshammari, 2023) at 99.9%, reflecting this study's advancement in network intrusion detection using robust class balancing strategies

**Table 3: Benchmarked Records of Related Works**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Ref** | **Year** | **Dataset** | **Model** | **Accuracy (%)** | **Precision (%)** | **Recall (%)** | **F1-Score (%)** |
| (Wanjau et al., 2021) | 2021 | CSE-CIC-IDS 2018 | CNN | 94.3 | 92.5 | 97.8 | 91.8 |
| (Otoom et al., 2023) | 2023 | MQTT-IoT-IDS2020 | DNN (Bi-flow) | 99.56 | 99.60 | 99.60 | 99.60 |
| DNN (Uni-flow) | 99.67 | 99.70 | 99.70 | 99.70 |
| (Alotibi & Alshammari, 2023) | 2023 | CSE-CIC-IDS 2018 | ANN | 99.9 | 100 | 98.0 | 98.3 |
| (Gouda et al., 2024) | 2024 | NF-UQ-NIDS-V2 | CNN | 98.7 | 99.0 | 98.8 | 99.0 |
| LSTM | 98.9 | 99.0 | 98.8 | 98.0 |
| This Study | 2025 | NF-UQ-NIDS | CNN-Downsizing | 99.73 | 99.54 | 99.92 | 99.73 |
| CNN-SMOTE | 99.82 | 99.70 | 99.95 | 99.82 |

Figure 12 presents this multi-metric comparison for the four top-performing models. This holistic analysis reveals not only the models' ability to correctly classify brute force attacks but also their effectiveness in minimizing false positives (precision) and capturing the majority of true attack instances (recall).

Figure 12: Multi-Metric Model Comparison of 99.6% Accuracy Threshold

The multi-metric comparative analysis of the top-performing models shows that ANN model achieved the highest accuracy at 99.90%, while this study’s CNN-SMOTE model closely followed with 99.82%. However, the ANN model showed a trade-off, with a lower recall of 98.0% despite perfect precision, indicating a higher likelihood of missing actual brute force attacks.

The CNN-SMOTE model offered the most balanced performance, achieving high precision (99.70%), near-perfect recall (99.95%), and a strong F1-score (99.82%). This balance suggests superior reliability and detection capability, especially in imbalanced datasets. Compared to traditional architectures like DNN and ANN, the CNN-based models in this study, particularly with SMOTE, demonstrate enhanced generalization and robustness, making them highly effective for brute force attack detection.

# 5.0. CONCLUSION AND RECOMMENDATION

This study successfully demonstrated the capability of a Convolutional Neural Network (CNN) to detect brute force attacks using flow-based network traffic data. By employing two balancing strategies, Downsampling and SMOTE. The CNN model was able to learn meaningful temporal and spatial features that distinguish malicious activity from normal traffic. The SMOTE-enhanced CNN achieved the highest performance with an accuracy of 99.82%, recall of 99.95%, and F1-score of 99.82%, outperforming traditional models and previous works in the literature. These results underscore the model's reliability, precision, and superior detection capabilities, particularly in handling class-imbalanced intrusion data.

In light of these findings, it is recommended that deep learning models, particularly CNNs integrated with synthetic oversampling methods like SMOTE, be adopted in modern intrusion detection systems. Their ability to automatically learn complex patterns from large-scale network traffic makes them well-suited for real-time security applications.

For future research, extending this approach to other types of cyber-attacks and exploring hybrid deep learning architectures such as CNN-LSTM could further enhance detection accuracy. Additionally, incorporating explainable AI methods can improve model transparency and trust in deployment scenarios.

# REFERENCE

Agghey, A. Z., Mwinuka, L. J., Pandhare, S. M., Dida, M. A., & Ndibwile, J. D. (2021). Detection of Username Enumeration Attack on SSH Protocol: Machine Learning Approach. *Symmetry*, *13*(11), 2192. https://doi.org/10.3390/sym13112192

Ali Hamza, A., & Jumma surayh Al-Janabi, R. (2024). Detecting Brute Force Attacks Using Machine Learning. *BIO Web of Conferences*, *97*, 00045. https://doi.org/10.1051/bioconf/20249700045

Alotibi, N., Advanced, M. A.-I. J. of, & 2023, undefined. (n.d.). Deep learning-based intrusion detection: A novel approach for identifying brute-force attacks on ftp and ssh protocol. *Search.Proquest.Com*. Retrieved February 17, 2025, from https://search.proquest.com/openview/e0e053ffe5b850bd93cfa58052d559f6/1?pq-origsite=gscholar&cbl=5444811

Alotibi, N., & Alshammari, M. (2023). Deep Learning-based Intrusion Detection: A Novel Approach for Identifying Brute-Force Attacks on FTP and SSH Protocol. *International Journal of Advanced Computer Science and Applications*, *14*(6). https://doi.org/10.14569/IJACSA.2023.0140612

Gebresilassie, S. K., Rafferty, J., Chen, L., Cui, Z., & Abu-Tair, M. (2023). Transfer and CNN-Based De-Authentication (Disassociation) DoS Attack Detection in IoT Wi-Fi Networks. *Electronics (Switzerland)*, *12*(17). https://doi.org/10.3390/electronics12173731

Gouda, H. A., Ahmed, M. A., & Roushdy, M. I. (2024). Optimizing anomaly-based attack detection using classification machine learning. *Neural Computing and Applications*, *36*(6), 3239–3257. https://doi.org/10.1007/s00521-023-09309-y

Jeon, H., & Oh, S. (2020). Hybrid-Recursive Feature Elimination for Efficient Feature Selection. *Applied Sciences*, *10*(9), 3211. https://doi.org/10.3390/app10093211

Jyothi, G. C., Prakash, C., Babitha, G. A., & Kiran Kumar, G. H. (2022). Comparison Analysis of CNN, SVC and Random Forest Algorithms in Segmentation of Teeth X-Ray Images. *Asian Journal of Computer Science and Technology*, *11*(1), 40–47. https://doi.org/10.51983/ajcst-2022.11.1.3283

Nzenwata, U.-J., Abiodun, A. G., Olayinka, A., Adeniyi, O. J., & Akwaronwu, B. G. (2024). Parkinson’s Disease Prediction Using Convolutional Neural Networks and Hand-Drawn Image Analysis. *Asian Journal of Computer Science and Technology*, *13*(2), 1–13. https://doi.org/10.70112/ajcst-2024.13.2.4270

Otoom, A. F., Eleisah, W., & Abdallah, E. E. (2023). Deep Learning for Accurate Detection of Brute Force attacks on IoT Networks. *Procedia Computer Science*, *220*, 291–298. https://doi.org/10.1016/j.procs.2023.03.038

Pujari, J., Wasim, J., Tripathi, A., & Bangarimath, S. (2025). Comprehensive Study on Binary Classification for CNN-LNet Based Urban Change Detection Using Satellite Images. *Asian Journal of Computer Science and Technology*, *14*(1), 1–5. https://doi.org/10.70112/ajcst-2025.14.1.4330

Sarhan, M., Layeghy, S., Moustafa, N., & Portmann, M. (2021). *NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems* (pp. 117–135). https://doi.org/10.1007/978-3-030-72802-1\_9

Sarkunavathi, A., & Srinivasan, V. (2018). A Detailed Study on Advanced Persistent Threats: A Sophisticated Threat. *Asian Journal of Computer Science and Technology*, *7*(S1), 90–95. https://doi.org/10.51983/ajcst-2018.7.S1.1797

Sulochana, V. (2019). Implementation of Chunks of Image Password in Cloud Computing System. *Asian Journal of Computer Science and Technology*, *8*(S1), 54–57. https://doi.org/10.51983/ajcst-2019.8.S1.1958

Vijayan, P. M., & Sundar, S. (2024). IoT intrusion detection system using ensemble classifier and hyperparameter optimization using tuna search algorithm. *Journal of Autonomous Intelligence*, *7*(2). https://doi.org/10.32629/jai.v7i2.962

Wanjau, S. K., Wambugu, G. M., & Kamau, G. N. (2021). SSH-Brute Force Attack Detection Model based on Deep Learning. *International Journal of Computer Applications Technology and Research*, *10*(01), 42–50. https://doi.org/10.7753/IJCATR1001.1008

Yashaswi, L., Mekala, S., & Prasad, M. D. (2023). Human Anomaly Detection using Deep Learning. *Asian Journal of Computer Science and Technology*, *12*(1), 35–40. https://doi.org/10.51983/ajcst-2023.12.1.3630

Zhang, S., Xie, X., & Xu, Y. (2020). A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity. *IEEE Access*, *8*, 128250–128263. https://doi.org/10.1109/ACCESS.2020.3008433