LUCID++: Enhanced Deep-Learning Solution for DDoS Attack Detection

Syed M Minhaj Naqvi

Department of Computing and Mathematics, KFUPM, Dhahran, Saudi Arabia.

Corresponding author(s). E-mail(s): g202426160@kfupm.edu.sa;

Abstract

Distributed Denial of Service (DDoS) attacks are some of the most pervasive and consequential threats to the availability of online services. Although the lightweight deep learning model architectures, like LUCID, offer promising avenue for fast and low resource-hungry detections, their applicability on newer, evolved, and more complex datasets is largely unexplored. This paper addresses this gap by doing evaluation of the original LUCID architecture on the CIC-DDoS2019 dataset, which is a modern benchmark containing sophisticated reflective and amplification attacks that were not incorporated in previous studies. Additionally, we put forth an enhanced model, LUCID++, which includes Batch Normalization and a hidden dense layer in order to improve the training and classification performance. Both the baseline as well as the enhanced models were thoroughly tuned via a 5-fold cross-validation process using `GridSearchCV`. The baseline model was able to achieve a strong accuracy of 98.50%, but our enhanced version surpassed this benchmark, reaching 98.93% accuracy and an F1-score of 0.9903. Crucially, this performance gain comes from a more optimized decision boundary which reduces the False Positive Rate by almost 50% while keeping a 99.22% True Positive Rate. These outcomes reinforce that the lightweight 1D-CNN design still remains a largely effective means for detecting modern DDoS attacks. The results also confirm that targeted architectural refinements can give significant practical improvements in the reliability of detection.

Keywords: DDoS Detection, Deep Learning, CNN, LUCID, Cybersecurity

1 Introduction

Distributed Denial of Service (DDoS) attacks continue to be one of the most damaging and critical threats to the network availability. Such attacks have the capability to overwhelm and paralyze critical online services along with the potential to cause catastrophic loss of availability to vital systems, including Critical National Infrastructure (CNI). It can be very challenging to make out these attacks in a fast manner because of the fact that detection system must process a very big amount of network traffic in a real-time manner. Further compounding on this is the fact that attack vectors are by nature constantly evolving which makes static and rule-based methods truly not very effective. Because of these reasons, there is a urgent need for detection techniques that are very accurate and very computationally efficient, as they have to operate on line speed.

For tackling this, proposed lightweight Deep Learning (DL) methods shows promising results. LUCID model can be seen as the premier example of this. It is a DL model that uses Convolutional Neural Network (CNN) for getting state-of-the-art detection accuracy and the processing cost for this is 40 times lower that similar, but heavier models [1]. The LUCID study originally tested their method on several datasets, namely ISCX2012, CICIDS2017, and CSE-CIC-IDS2018, showing that it has viability in resource-constrained environments

Nonetheless, the threat landscape is always evolving and existing models' high performance on older data is not a guarantee of success on newer, more complex, and evolved attack patterns. The CIC-DDoS2019 dataset by UNB, featuring huge traffic and varying set of modern reflective and amplification-based attacks, was not part of the original LUCID study. Hence, its performance on this contemporary CIC-DDoS2019 dataset remains an open question.

This paper aims to bridge this gap through a threefold contribution. Firstly, we establish a comprehensive performance baseline by implementing and evaluating the original LUCID architecture on the CIC-DDoS2019 dataset. Secondly, we propose and implement an enhanced version of the original model architecture, named LUCID++, which integrates Batch Normalization and an additional dense layer to augment classification capability and training stability. In the end, we show an extensive evaluation for the performance of our modified architecture with the baseline. The results show us that even though the original model reaches a strong accuracy of 98.5% our enhanced model improves further on this to 98.93%. Most notably, it achieved a reduction of nearly 50% in the False Positive Rate on the test set, a critical metric for reducing alert fatigue in operational environments.

The rest of this paper is: Section 2 details the related work in the field. Section 3 details the methodology, including the dataset-specific characteristics, preprocessing steps, and model architectures. Section 4 details the experimental results and analysis, Section 5 provides a discussion of the findings, and Section 6 concludes with future work based on our results.

2 Background and Related Work

The detection of Distributed Denial of Service (DDoS) attacks is a well-established field within cybersecurity research. Over time, methodologies have shifted from simple statistical tests to high-performance deep learning models capable of automated feature extraction.

2.1 Traditional detection paradigms

Early work on this problem employed statistical as well as threshold-centric detection techniques [2, 3]. These early systems were engineered to monitor the network traffic for irregular patterns in packet headers, such as changes in the entropy of source IP addresses [4]. The logic behind this was, a volumetric DDoS attack would cause a sudden deviation from normal traffic behavior.The robustness of this process was not sound because of the fact that selecting effective threshold on different network environments was difficult [5]. Adversaries would still be able to trick the systems through various techniques like using low-and-slow or multi-vector techniques.

2.2 Classical machine learning for intrusion detection

Due to the above issue of fixed thresholds, there was a shift seen towards classical Machine Learning (ML). This new approach changed the focus from fixed rules to data-centric approach. Buczak and Guven survey the common models used for the end goal of intrusion detection, model like Support Vector Machines (SVMs), k-Nearest Neighbors (k-NN), and Random Forests [6]. While these models successfully raised detection accuracy, their performance remained very dependent on extensive, meticulous, and time-consuming feature engineering. Domain experts were needed for manually designing features like flow duration and packet inter-arrival times, consequently creating a bottleneck that slowed adaptation to novel attack types.

2.3 Deep learning for DDoS detection

In recent times, Deep Learning (DL) has come out as the dominant technique because it learns features directly from raw data with minimal to no manual intervention [1]. These models ingest low-level packet inputs and automatically learn very complex and non-linear patterns that distinguish benign traffic from malicious.

Two architectures have become dominant in this domain:

- Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTMs) are used to model the temporal sequence of packets in a flow, effectively capturing time-based attack patterns [7].
- Convolutional Neural Networks (CNNs) treat traffic flows as spatial structures, like 1D or 2D images, and learn from the spatial relationships between packet features within a flow [8].

2.4 The lightweight model imperative and LUCID

While effective, many deep learning models are computationally heavy, containing millions of trainable parameters. DeepDefense, which combined CNNs and RNNs, is one such example [7]. This high level of complexity raises processing cost and hinders real-time deployment on resource-constrained edge gateways or on-device IoT platforms [9, 10].

This gap in performance motivates the need for lightweight and accurate models. LUCID (Lightweight, Usable CNN in DDoS Detection) is a primary example of this philosophy [1]. Doriguzzi-Corin et al. designed LUCID as a practical deep model with exceptionally low overhead. Their network is a compact 1D-CNN with only 2,241 parameters. The authors showed that it matches the performance of much heavier models like 3LSTM, which has over one million parameters, while running approximately 40 times faster.

The desirable combination of the accuracy and the speed make LUCID a good baseline model for use in modern DDoS detection. We have use LUCID as baseline as well, for our study. We validate it against CIC-DDoS2019 dataset, which is a dataset that the original study did not include. We also try to do architectural enhancements in it for getting even better performance, beyond the baseline model.

3 Methodology

Our work uses comparative experimental approach, for evaluating rigorously, the efficiency of deep learning models that are lightweight. We test out the original LUCID model with our improved version, LUCID++. Training and the evaluation for both models was done on the CIC-DDoS2019 dataset which is a benchmark not originally included in the LUCID study. The main goal is to see how specific architectural changes, namely the addition of Batch Normalization and a dense hidden layer, influence the stability of training, the convergence speed, and overall detection accuracy.

3.1 Dataset and Preprocessing

The CIC-DDoS2019 dataset was selected for its relevance to the current threat landscape. It contains a broad mix of attack types, including sophisticated reflective and amplification-based flood attacks such as NTP, DNS, LDAP, and NetBIOS. It provides modern traffic patterns that reflect current DDoS techniques far better than older datasets.

Our preprocessing steps very faithfully uses the lightweight methodology which is described by the original LUCID study. This has been undertaken to ensure that comparison is fair. The whole traffic is arranged into non-overlapping windows of ten seconds. Packets within each of these windows are grouped together by bi-directional flow and each of the flows is then divided into samples containing at most ten packets. From each of the packets eleven specific features are pulled: relative timestamp, packet length, IP flags, protocol information, and transport-layer fields for TCP, UDP, and ICMP. Importantly, IP addresses and port numbers are not included because we wanted to ensure that the model learns generalized patterns, instead of simply just memorizing. Flows that are smaller than ten-packets are padded with zeros and all features are normalized between zero and one. At the end of the preprocessing step, the final data shape is three-dimensional tensor size (10, 11, 1). This means ten packets, eleven features, and one channel. This format is suited for a two-dimensional convolution.

3.2 Model Architectures

Two neural models were built and trained using identical optimization and loss settings to isolate the impact of architectural changes. Both models utilize the AdamW optimizer and binary cross-entropy as the loss function.

3.2.1 Baseline Model (LUCID)

The baseline model is a direct implementation of the original LUCID design [1]. As described in the paper, this architecture uses a convolutional layer where the filter's width, f, matches the width of the input feature matrix ($f = 11$). This design forces the filter to slide in only one dimension (along the n packets), thus behaving as a 1D convolution over the temporal axis.

In our implementation, this is built using a 2D Convolutional Layer ('Conv2D') with a kernel size of (h, f). This is followed by a ReLU activation, a 'GlobalMaxPooling2D' layer, a 'Flatten' operation, and a 'Dense' output layer with a sigmoid activation. The original architecture is depicted in Figure 1.

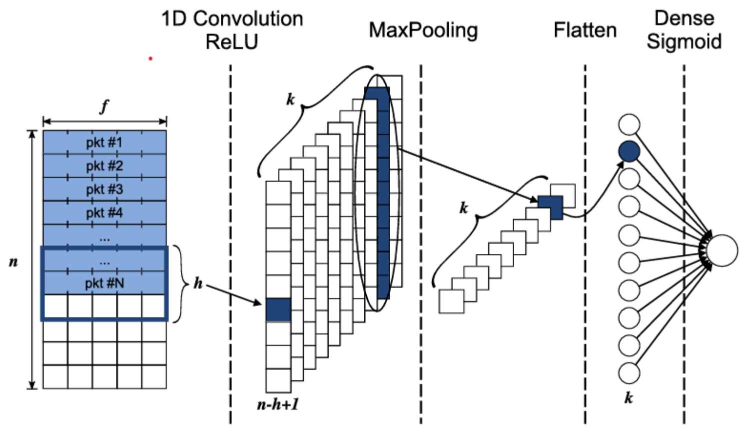

Fig. 1 The original LUCID model architecture (adapted from [1]).

3.2.2 Proposed Enhanced Model (LUCID++)

The proposed model, LUCID++, extends this design with two modifications. A **BatchNormalization** layer follows the 'Conv2D' layer to stabilize activations and improve convergence. A **Dense** hidden layer with n units and ReLU activation is added after the ‘Flatten‘ operation to learn richer combinations of extracted features. Dropout regularization is applied to this new layer to prevent overfitting. This enhanced architecture is shown in Figure 2.

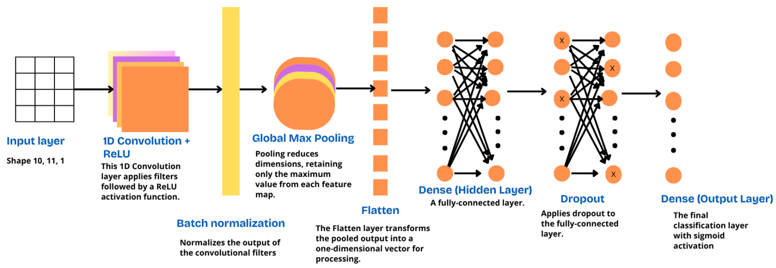

Fig. 2 The proposed LUCID++ model architecture.

3.3 Evaluation Setup

Model evaluation relies on a rigorous five-fold cross-validation process managed through ‘GridSearchCV‘. Each fold trains and validates models across a pre-defined hyperparameter grid, and the configuration yielding the highest F1-score on the validation folds is selected for retraining on the full training set. The grid explores a range of values for learning rate (0.01 and 0.001), batch size (1024 and 2048), convolution filter count (32 and 64), L1 regularization, convolutional dropout (0 or 0.2), and AdamW weight decay (0.01, 0.001, and 0). For the LUCID++ model, the size of the dense layer and its dropout rate are also varied.

The final, optimized model is tested on a strictly held-out subset of the dataset to ensure unbiased evaluation. For measuring the performance different metric are used. These include accuracy, F1-score, true positive rate, false positive rate, false negative rate, and the ROC-AUC curve. Together these metrics give a full view of the detection reliability of the model as well as the error balance.

3.4 Implementation and Reproducibility

The experiments we undertook were all performed on a Windows 11 workstation having NVIDIA 4060 GPU and 32GB RAM. The implementation of the model was done in Python 3 using TensorFlow 2.x and Keras. Scikeras was also used in order for interfacing Keras with Scikit-learn to use in hyperparameter search. Data handling and the I/O was done by NumPy and h5py. Also, PyShark was installed for parsing of the raw PCAP data, which requires the installing of TShark (Wireshark CLI).

In order to ensure the reproducibility of our results a fixed random seed of 42 was set for NumPy, TensorFlow, and Scikit-learn. Doing this guarantees that there is consistency in data splits and model initialization for each of the runs. Full source code, model definitions, and all of the hyperparameter settings are available publicly online in GitHub repository for our project [11]. This facilitates the exact replication of our experiments.

4 Results and Analysis

This section will give our experimental results which were gathered throughout our study. We begin by explanation of our extensive hyperparameter tuning process, then we proceed to delving in the training dynamics and convergence behavior of our final model, ending with an extensive evaluation of the classification performance on unseen test set.

4.1 Hyperparameter Tuning and Model Performance

For the optimization of the model's hyperparameter we used a comprehensive 5-fold cross-validation technique which we orchestrated using the 'GridSearchCV' technique. This extensive validation process was used on the entirety of the training dataset in order for ensuring that the selected parameters were not biased towards any of the specific subset of the data. In totality, our search space covered 1,920 unique combinations of the hyperparameter and each of these combinations was separately trained and evaluated so that we could ensure the final selected configuration was in actual the global optimum in our search space.

The searching process found the following specific configuration which was optimal for maximizing the F1-score of the enhanced LUCID++ model:

- **Batch Size:** 1024
- **Learning Rate:** 0.001
- **Weight Decay (AdamW):** 0.001
- **Kernels (Filters):** 64
- **Convolutional Dropout:** None
- **Regularization (L1):** None
- **Dense Units:** 16
- **Dense Dropout:** 0.5

Table 1 shows the comparison of the performance analysis between the original LUCID baseline model and our enhanced model LUCID++ on the test set. It shows that the LUCID++ model was able to achieve quantifiable improvement giving accuracy increase of 0.0043 (0.43%) and an F1-Score increase of 0.0037 (0.37%).

Although these improvements might seem a little bit marginal, they prove to be profound when looking at the balance between True Positives and False Positives. The baseline model was too much aggressive and achieved a near-perfect True Positive Rate (TPR) at the cost of False Positive Rate (FPR) of 2.86%. In comparison our enhanced model gave a little better equilibrium. It was able to successfully reduced FPR by 50% bringing it down to 1.43%, while only having a 0.78% cost in False Negative Rate (FNR). This shows a big practical improvement as it radically cuts down on the false alarms while not compromising on the ability of the system to detect actual threats.

Table 1 Performance Comparison on the CIC-DDoS2019 Test Set

Model	Accuracy	F1-Score	TPR	FPR	FNR	TNR
Baseline (LUCID)	0.9850	0.9866	0.9961	0.0286	0.0039	0.9714
LUCID++	**0.9893**	**0.9903**	0.9922	0.0143	0.0078	0.9857

4.2 Training Dynamics

The training behavior of LUCID++ model is seen in Figure 3. It plots accuracy and the loss over 300 epochs. These graphs give excellent insight for gauging the stability as well as the learning progression of the model. Looking at the loss curve on the right side in Figure 3, a consistent and a steady decrease is seen in the loss value for both training and validation part. The validation loss curve follows the training loss curve very closely and shows no divergence. This shows that the regularization and optimization techniques namely Dense Dropout, Batch Normalization, and the AdamW optimizer were very highly effective in overfitting prevention. On the left the accuracy curve shows interesting characteristics as well. At the start, there is much instability and high fluctuation particularly for the validation curve. Here the model explores the whole feature space and changes weights from random initialization. Around epoch 100 a sharp and a stable convergence is seen. After this, both the training and the validation accuracy curves climb fast to near perfect values and are tightly coupled till the end. This close coupling shows that the model successfully has learned a generalized representation from the very complex data patterns, not only just memorizing from the training set.

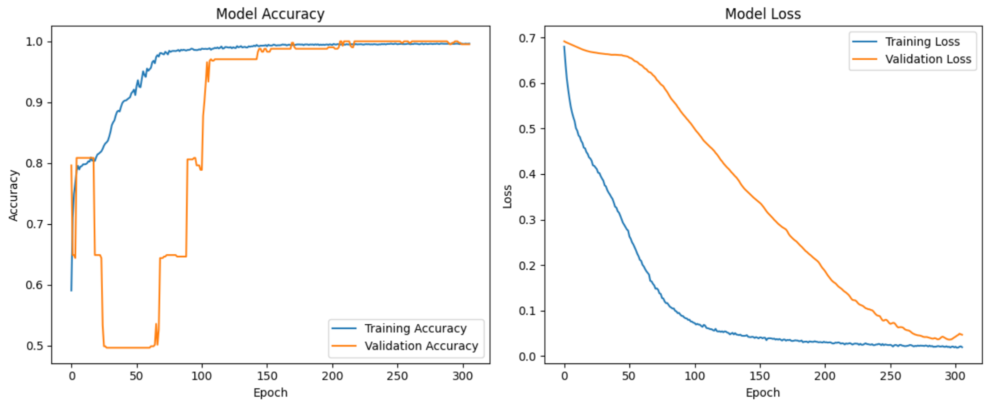

Fig. 3 Model Accuracy (Left) and Loss (Right) During Training (LUCID++ Architecture)

4.3 Classification Analysis

In order for providing a more granular and meticulous testing of the final model's performance, we generated a detailed classification report and a Receiver Operating Characteristic (ROC) curve using the unseen test dataset.

The classification report, shown in Table 2, breaks down the precision, recall, and F1-score for each specific class. The model showed exceptional performance, achieving scores of approximately 0.99 across all these three metrics for both the 'Benign' as well as the 'DDoS' class. Specifically, the model correctly identified 99.22% of all malicious DDoS attack samples (Recall/TPR), while simultaneously correctly identifying 98.57% of all benign traffic samples (Recall/TNR). This consistency in performance across classes confirms the balanced nature of the classifier, and confirms that it is not biased toward the majority class.

Table 2 Classification Report for LUCID++ on Test Set

	Precision	**Recall**	**F1-Score**	**Support**
Benign (0)	0.9904	0.9857	0.9881	210
DDoS (1)	0.9884	0.9922	0.9903	258
Accuracy			**0.9893**	**468**
Macro Avg	0.9894	0.9890	0.9892	468
Weighted Avg	0.9893	0.9893	0.9893	468

Figure 4 shows the critical trade-off that occurs between the True Positive Rate and the False Positive Rate across all possible decision thresholds. Our final model achieved an Area Under the Curve (AUC) of **0.9927**, a value that is very close to the theoretical maximum of 1.0. This extremely high AUC score indicates an outstanding level of class separability, meaning that the model can distinguish between benign and malicious traffic with a very high confidence. Visually, the curve shows a sharp and almost vertical rise toward the top-left corner of the plot. This confirms that the model is capable of achieving a very high True Positive Rate while maintaining an extremely low False Positive Rate, further supporting its suitability for deployment in high-security environments where minimizing false alarms is extremely important.

5 Discussion

The experimental results detailed in Section 4 provide a strong validation for the architectural enhancements put forth in our LUCID++ model. By achieving a final accuracy of 98.93% and an F1-Score of 0.9903, the enhanced model not only outperformed the baseline, but it also showed a significantly more robust and operationally desirable performance.

The most significant finding of this study lies in the analysis of the error types. The baseline LUCID model showed a high sensitivity identifying nearly all attacks with a 99.61% TPR. However, this sensitivity came at the cost of a 2.86% False Positive Rate. In the context of a vast-telemetry Security Operations Center (SOC), a 2.86%

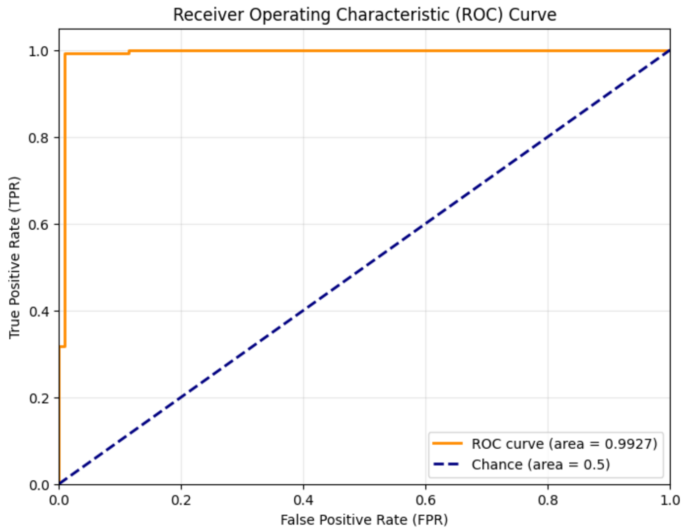

Fig. 4 ROC Curve for the Enhanced Model (LUCID++) on the Test Set (AUC = 0.9927)

false alarm rate can translate to a lot of incorrect alerts, leading to "alert fatigue" in analysts. Our optimized LUCID++ model successfully addressed this critical issue. By reducing the FPR to 1.43%, a decrease of nearly 50% and while maintaining a 99.22% detection rate, the enhanced model offers a more practical solution for real-world deployment.

We attribute these performance gains to the two primary architectural modifications we performed. First, the addition of a **BatchNormalization** layer played a central role in stabilizing the training process. By normalizing the activations output by the convolutional layer, this mechanism smoothed the optimization process, preventing the model from getting stuck in poor local minima during the early epochs which were very volatile as seen in Figure 3. Second, the addition of the hidden **Dense(16)** layer provided the network with increased representational capacity. Unlike the original architecture, which translated extracted features directly to the output, this intermediate layer allowed the model to learn more complex, non-linear combinations of the features before making a final classification decision.

It is worth noting that while our model achieved an F1-Score of 0.9903 on the CIC-DDoS2019 dataset, this value is not directly comparable to the 0.9987 F1-Score reported in the original LUCID paper for the CSE-CIC-IDS2018 dataset [1]. These datasets differ significantly in terms of attack vectors, traffic volume, and network topology. However, the fact that our lightweight model achieved such elite performance on a newer and a more complex dataset serves as a strong validation that the 1D-CNN approach is highly effective and deployable to modern threat landscapes.

Despite these successes, the study has its limitations. The primary limitation is the remaining 1.43% False Positive Rate. While a significant improvement, further reduction is needed for automated mitigation systems. Additionally, our evaluation was restricted to only a single new dataset, leaving the model's performance on wider variety of network environments an open question. Finally, while the model architecture remains relatively lightweight in terms of parameter count, a rigorous measurement of inference latency and memory usage on edge hardware was outside the scope of this study.

6 Conclusion and Future Work

6.1 Conclusion

This study showed a very extensive evaluation and the enhancement of the lightweight LUCID deep learning model for DDoS attack detection. We began by establishing a baseline model using the original LUCID model which gave 98.5% accuracy on the CIC-DDoS2019 dataset. After this we put forward LUCID++ which is an enhanced version of the original LUCID model, but it includes Batch Normalization, an additional dense hidden layer, and Dense dropout. Utilizing a extensive hyperparameter search our enhanced model achieved a better accuracy of 98.93% and F1-Score of 0.9903. This improvement showed a near 50% reduction in false positives and achieved a much more better balance between the sensitivity and the specificity. These results show us that the lightweight 1D-CNN architectures remain a very powerful and very efficient tool for the network defense and also demonstrate that their performance may be improved highly by using targeted modern architectural changes.

6.2 Future Work

Based on the findings of this study, we propose three directions for future research endeavors:

- **False Positive Reduction:** Future work should focus specifically on bringing down the remaining 1.43% FPR. Potential techniques for this include the use of advanced data augmentation to better represent the benign class or the integration of a secondary specialized anomaly filter designed to double-check borderline classifications.
- **Cross-Dataset Generalization:** For further validating the robustness of the LUCID++ architecture it should be evaluated across a broader spectrum of datasets, including the original ISCX2012, CICIDS2017, and CSE-CIC-IDS2018 datasets. This would help to determine if the performance gains we observed are universal or specific to the dataset we used.
- **Edge Deployment Benchmarking:** An important next step is to measure the real-world computational cost of the enhanced model. This would involve deploying LUCID++ on a resource-constrained edge device, such as the NVIDIA Jetson TX2 used in the original study [1] to rigorously compare its inference latency and memory footprint against the baseline model.

Declarations

- **Funding:** Not applicable.

- **Conflict of interest:** The authors declare that they have no conflict of interest.
- **Data availability:** The dataset analyzed during the current study is available in the CIC-DDoS2019 repository.
- **Code availability:** The code is available at the GitHub repository referenced in the text.

References

[1] Doriguzzi-Corin, R., Millar, S., Scott-Hayward, S., Martínez-del-Rincón, J., Siracusa, D.: Lucid: A practical, lightweight deep learning solution for ddos attack detection. IEEE Trans. Netw. Serv. Manag. **17**(2), 876–889 (2020)

[2] Bojović, P., Bašičević, I., Ocovaj, S., Popović, M.: A practical approach to detection of distributed denial-of-service attacks using a hybrid detection method. Comput. Elect. Eng. **73**, 84–96 (2019)

[3] Kalkan, K., Altay, L., Gür, G., Alagöz, F.: Jess: Joint entropy-based ddos defense scheme in sdn. IEEE J. Sel. Areas Commun. **36**(10), 2358–2372 (2018)

[4] Feinstein, L., Schnackenberg, D., Balupari, R., Kindred, D.: Statistical approaches to ddos attack detection and response. In: Proc. DARPA Inf. Survivability Conf. Expo., p. 303 (2003)

[5] Shah, S.B.I., Anbar, M., Al-Ani, A., Al-Ani, A.K.: Hybridizing entropy based mechanism with adaptive threshold algorithm to detect ra flooding attack in ipv6 networks. In: Computational Science and Technology, pp. 315–323. Springer, Singapore (2019)

[6] Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surveys Tuts. **18**(2), 1153–1176 (2016)

[7] Yuan, X., Li, C., Li, X.: Deepdefense: Identifying ddos attack via deep learning. In: Proc. SMARTCOMP, pp. 1–8 (2017)

[8] Wu, K., Chen, Z., Li, W.: A novel intrusion detection model for a massive network using convolutional neural networks. IEEE Access **6**, 50850–50859 (2018)

[9] Roopak, M., Tian, G.Y., Chambers, J.: Deep learning models for cyber security in iot networks. In: Proc. IEEE CCWC, pp. 452–457 (2019)

[10] Wang, N., Varghese, B., Matthaiou, M., Nikolopoulos, D.S.: Enorm: A framework for edge node resource management. IEEE Trans. Services Comput. (2017)

[11] Naqvi, S.M.M.: Github Repository for LUCID++: Enhanced Deep-Learning Solution for DDoS Attack Detection. https://github.com/BRAIN-Lab-AI/LUCID-Enhanced-Deep-Learning-Solution-for-DDoS-Attack-Detection