2025 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom) | 979-8-3315-3719-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/BlackSeaCom65655.2025.11193891

# OB-IDS: Optimized BERT-based Intrusion Detection System

Göktuğ Ateş*, Başar Çelebi*, Umut Aytuğ Semerci, Emircan Çapkan,
Batuhan Yıldırım, İlktan Ar*, Taner Arsan
Department of Computer Engineering, Kadir Has University, Istanbul, Turkey
*g.ates, basarcelebi02, ilktana@stu.khas.edu.tr

*Abstract*—**"OB-IDS (Optimized BERT-based Intrusion Detection System)" presents a lightweight IDS optimized for the resource-constrained environments. The proposed system which employs quantization, pruning, knowledge distillation and self distillation methods is developed in a multi-stage optimization manner. It was evaluated on UNSW-NB15 and CIC-IDS2017 datasets. The proposed system was found to significantly reduce the inference time and memory usage while maintaining high classification accuracy. The experimental results demonstrate the applicability of Transformer-based IDSs for real-time threat detection in resource-constrained environments.**

*Index Terms*—**intrusion detection, BERT, model optimization, network security, fine-tuning, quantization, machine learning, cybersecurity**

## I. Introduction

Network intrusion is defined as attempts of unauthorized access, distributing malware, and sophisticated attacks that have been successful and have caused damage to system integrity, security, and the convenience of our digital infrastructure [1]. The proposed system has focused on addressing this critical cybersecurity challenge, particularly in resource-constrained environments, inspired by the studies on the Bidirectional Encoder Representations from Transformers (BERT) [2].

According to the historical information, network security was primarily maintained by signature-based firewalls and IDS technologies that, without continuous improvement, eventually provided attackers with opportunities to overcome these defenses due to their restricted ability to recognize only previously identified attack patterns [3]. Rule-based intrusion detection systems were published by Denning between 1980 and 1990 [4]. Signature-based intrusion detection systems appeared such as SNORT in the mid-1990s [5]. The scope of statistical modeling in modern cybersecurity has expanded with the introduction of anomaly-based intrusion detection systems in the late 1990s and 2000s. Hybrid intrusion detection systems appeared with the combined use of signature-based and anomaly-based systems in the mid-2000s. The development of machine learning-based intrusion detection systems became widespread after the 2010s. At the beginning of the 2020s, transformer-based IDSs have become common. In the 2010s, studies primarily focused on the accuracy of the models, whereas they have focused on the efficieny and explanability of the models in the 2020s.

There are many different studies available in the field of network intrusion detection according to the literature research. Sayegh et al. [6] utilized an enhanced LSTM-based model to classify network packets in UNSW-NB15 [7] and CIC-IDS2017 [8] datasets, focusing on identifying anomalies and differentiating them from benign behaviors to establish an effective detection system. Halbouni et al. [9], aimed to detect network intrusion patterns more effectively and efficiently in the network packets using CIC-IDS2017 dataset. In a different study, Altunay et. al [10] utilized hybrid intrusion detection system (IDS) model using convolutional neural network (CNN) and long-short term memory (LSTM) architectures in UNSW-NB15 dataset. Sanmorino et al. [11], integrated the Deep Neural Network (DNN) pre-trained model, which was trained on a large-scale dataset, into the network intrusion system using transfer learning techniques. A metric mechanism was created to perform network intrusion tasks using the Network-based Security Lab - KDD dataset (NSL-KDD) dataset [12]. This article argues that, as a result of the research, the fine-tuned model gives much better results in many parameters, such as accuracy, precision, and F1-score, than the untuned model. Studies in the literature conducted with specific models or fine-tuned models in the field of network intrusion detection are very limited for fine-tuning. The purpose of this study is to add a new one to the research conducted in this field and to contribute to the scientific world.

Our work aims to enable machine learning-based intrusion detection systems, which traditionally require high processing power, to operate efficiently in resource-constrained environments. Our literature review has revealed that studies focusing on fine-tuning models, especially for resource-constrained IDS systems, are limited. However, this fine-tuning process is critical to significantly reduce computational costs while maintaining detection accuracy. Therefore, in our work, we developed and evaluated fine-tuned ML models that balance both performance and resource efficiency for network-based intrusion detection.

The rest of this paper is organized as follows. Firstly, methodology is represented in Section II. Section III describes the experimental results from the proposed system and their comparisons with related works. Finally, conclusions are drawn in Section IV.

979-8-3315-3719-7/25/$31.00 ©2025 IEEE

## II. Methodology

### A. Dataset

Two different network intrusion datasets, UNSW-NB15 and CIC-IDS2017, were used to evaluate the proposed system. Both datasets provide flow-level CSV files and raw packet data as PCAP. UNSW-NB15 dataset consists of four separate CSV files and 80 PCAP files. The CIC-IDS2017 dataset, however, consists of eight CSV files and five principal PCAP files.

In the preprocessing, the class imbalance problem was addressed in both datasets. Some types of attacks in both datasets contained millions of packets while others contained very few. To address this imbalance, undersampling was performed on overly occurring classes and oversampling on classes containing few samples. In addition, only TCP/IP-based packets with a payload size greater than zero were retained for analysis.

The packet consists of two different parts. These sections are IPv4 Header fields and TCP Header fields. IPv4 Header part consists of the version in which the IP version is specified, the internet header length (IHL) representing the length of IPv4 header, type of service indicating QoS level, total length expressing the total packet length of IPv4 Header and payload, header checksum providing error control in IPv4 Header, source address hosting IP address of the device sending the packet, and the destination address representing IP address of the target device, as shown in Fig.1.

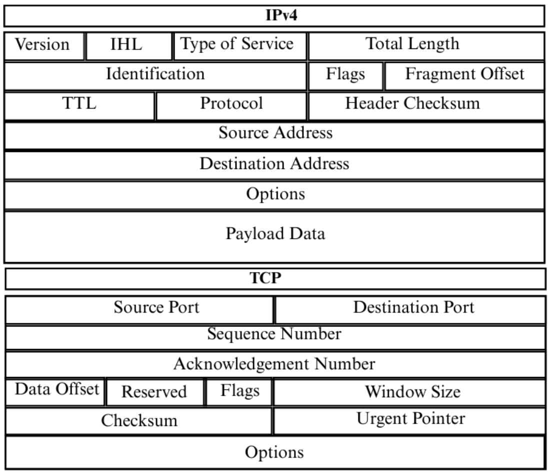

Fig. 1. TCP/IP Packet Structure [13]

TCP Header fields consist of parts such as the source port, which contains the port address of the sending application, the destination port, which represents the port address of the destination application, the sequence number, which indicates the order of the data, and the recognition number used for the confirmation of the transmitted data, as shown in Fig.1. [14]

After combining IPv4 and TCP structures, SYN Packets, which are utilized in establishing connections but do not carry data, have been removed so that they were not utilized in training the models. Similarly, unnecessary TCP checksums and hard-coded IPs used in closed network environments have been removed. To enable flow-level statistics to be backtracked to packet-level data, three packet-level subsets were constructed which are ”Packet Fields” (with only header information), ”Packet Bytes” (decimal values between 0–255 of headers and payloads), and ”Payload Bytes” (decimal value of the payload section only).

The final combined dataset is a dataset with 24 different classes as a result of combining UNSW-NB15 and CIC-IDS2017 datasets, 50,000 samples from each class, a total of 1.2 million packet data. This dataset consists of two different columns, including the packet data column and the attack category column. Accordingly, the final combined dataset is divided into 3 different subsets. These subsets are training, validation and test set. While the training set was created from 70% of the dataset, the validation and test sets shared the remaining 30% equally.

### B. The Proposed System

The proposed system, which is applied to the baseline model, is indicated, as shown in Fig.2. The experiment structure contains quantization, pruning, knowledge distillation, and self distillation respectively.

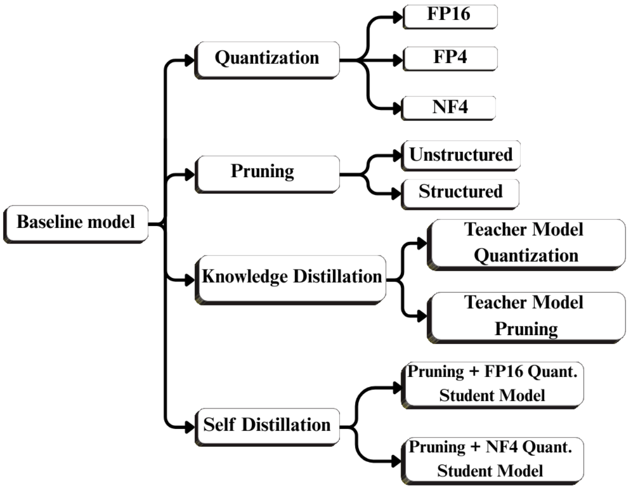

Fig. 2. The Experiment Structure

*1) Baseline Model:* The Baseline model has been fine tuned for the task of classifying packets on 24 different classes by converting the flow, header and payload information contained in packets to text using BERT architecture. It is important to emphasize that BERT is suitable for multi label classification task thanks to the multi head attention mechanism. After tokenization, [CLS] token, which serves as an input representation for the classification tasks, is added to the input sequence in the model. Accordingly, the bidirectional association of [CLS] token given as model inputs in a contextual way allows it to create original patterns for texts belonging to different labels. [15]

*2) Optimization Methods:* Quantization reduces the weights of the model using precision levels. It allows the model to consume less memory, inference time and energy in terms of the system resources at low system requirements. Accordingly, floating point (FP) and normalized floating point (NF) quantization methods at 32-bit, 16-bit and 4-bit precision levels were used in the experiment structure. [16]

Pruning aims to ensure that the model provides performance enhancement in terms of throughput by removing the weights of the model or the components such as neurons located in its intermediate layers. There are two main types of pruning: unstructured pruning and structured pruning. In the experiment structure, unstructured pruning aimed at pruning only on the weights of the model [17] and structured pruning aimed at pruning on the intermediate layers of the model. [18]

Knowledge distillation (KD) aims to use soft labels emerging from the teacher model and the ground truth labels contained in the dataset according to the loss function determined during the fine tuning process on the student model, without compromising accuracy and F1-score. The experiments were carried out as BERT-based teacher model and TinyBERT-based student model. The loss function is based on the mathematical distribution of cross entropy loss and KL (Kullback-Leibler) divergence loss functions. Accordingly, the ground truth labels are evaluated with cross entropy loss, while the soft labels are evaluated with KL divergence loss. The experiments were carried out as BERT-based teacher model and TinyBERT-based student model. [19]

Self distillation (SD), unlike knowledge distillation, employs the model as its own teacher and it has been shown to get better current performance by not adding parameters to the model or reducing overfittin. Also, it aims to improve generalization without additional fine tuning. The loss function used for knowledge distillation has also been used for self distillation. [20]

## III. Experimental Results

In the experiment structure, the fine tuning processes were performed on A100 GPU with 40 GB of high-bandwidth VRAM, and the testing processes were performed on L4 GPU with 22.5 GB of GDDR6 supported VRAM. The models were evaluated according to the model size, accuracy, F1-score and throughput (flows/s), which express the number of $m$ flows in each $n$ batch with the batch rate ($n$ batches per second), such as (1 batch/s x 32 flows).

TABLE I
Comparison of Quantization Precision Types

| Precision Type | Size (MB) | Accuracy (%) | F1-score (%) | Throughput (flows/s) |
|---|---|---|---|---|
| Baseline (FP32) | 251.01 | 99.5627 | 98.70 | 3702 |
| FP16 | 125.50 | 99.5627 | 98.70 | **3743** |
| FP4 | **21.11** | 99.5560 | 98.68 | 1466 |
| NF4 | **21.11** | **99.5639** | **98.74** | 1504 |

In order to evaluate BERT-based IDS across precision types, quantization in FP16, FP4, and NF4 precisions were applied and compared with the baseline model. NF4 quantization improved accuracy and F1-score but slightly reduced throughput in Table I. FP16 quantization matched the baseline model in accuracy while improving throughput. FP4 quantization performed worse in both metrics and was excluded from further evaluation. FP16 and NF4 quantization methods were employed for subsequent fine-tuning and testing.

TABLE II
Impact of Structured Pruning and Quantization on Model Size, Accuracy, F1-score and Troughput

| Method | Size (MB) | Accuracy (%) | F1-Score (%) | Throughput (flows/s) |
|---|---|---|---|---|
| Baseline Model | 251.01 | 99.5627 | 98.71 | **3702** |
| Unstructured Pruning | 251.01 | 99.5611 | 98.70 | 3261 |
| Structured Pruning | 251.01 | **99.5655** | **98.74** | 3242 |
| Structured + FP16 | 125.50 | **99.5655** | **98.74** | 3589 |
| Structured + NF4 | **84.44** | 99.5619 | **98.74** | 1452 |

In the workflows of pruning methods, structured pruning slightly improved both accuracy and F1-score compared to the baseline model. Structured pruning was preferred for further compression using FP16 and NF4 quantization. After applying structured pruning and quantization to the baseline model, it has been observed that employing NF4 quantization after structured pruning reduces the accuracy and throughput of the model.

TABLE III
Accuracy, F1-score, and Throughput for KD and SD under Different Compression Settings

| Method | Accuracy (%) | F1-score (%) | Throughput (flows/s) |
|---|---|---|---|
| Baseline Model | **99.56** | **98.71** | 3701.73 |
| FP16 Quant. (Teacher) + KD | 99.19 | 92.87 | 3725.99 |
| NF4 Quant. (Teacher) + KD | 99.19 | 94.02 | 3633.13 |
| Pruning + FP16 (Teacher) + KD | 99.24 | 95.32 | 3707.12 |
| Pruning + NF4 (Teacher) + KD | 99.24 | 94.38 | **3739.73** |
| Pruned-FP16 (Teacher) + KD + SD | 99.27 | 96.84 | 3694.71 |
| Pruned-NF4 (Teacher) + KD + SD | 99.23 | 94.63 | 3666.00 |

Four KD workflows and two SD workflows for student models that resulting from KD workflows were implemented for the model compression, as shown in Table IV. 512 as the padding length, AdamW [21] as the optimizer, 5e-5 as the learning rate were determined for the hyperparameters of KD and SD workflows. Loss function was applied with alpha (0.5) and temperature (2.0). Fine-tuning was performed within 15/20 epochs for KD workflows and within 5/10 epochs for SD workflows. Among KD workflows, both structured pruning and quantization applied workflows on the teacher model performed better in terms of accuracy and F1-score compared to other two workflows. In terms of throughput, they performed better than the baseline model. Due to slightly lower performance of the distilled models rather than the baseline model, two student models with the highest F1-score were used for self distillation in Table IV. When applying SD workflows, the distilled model from FP16 quantized teacher model performed much better than the distilled model from NF4 quantized teacher model in terms of accuracy, but it has performed below F1-score rather than the baseline model. All compressed models reduced the baseline model size from 251 MB down to 72.6 MB, a 71% reduction, regardless of the compression variant. As the quantization methods are applied only to the teacher, FP32-based TinyBERT student models keep the same size.

TABLE IV
COMPARISON OF THE EXPERIMENT & RELATED WORKS

| Model | Dataset | Accuracy (%) | F1-score (%) |
|---|---|---|---|
| Pruned-FP16 BERT | Combined | **99.56** | **98.74** |
| LSTM [6] | UNSW-NB15 | 98.31 | 98.32 |
| LSTM [6] | CIC-IDS2017 | 99.34 | 99.35 |
| CNN [9] | CIC-IDS2017 | 99.55 | 98.48 |
| CNN-LSTM Hybrid [10] | UNSW-NB15 | 92.90 | 92.59 |

Pruned and FP16 quantized BERT model performed well in UNSWB-NB15 and CIC-IDS2017 datasets compared to LSTM-based IDS. Additionally, All quantizations with/without pruning methods used in the experiment structure performed better rather than LSTM-based IDS. The best experimental model exhibited more successful F1-score performance although it performed as much accuracy performance as CNN-IDS in CIC-IDS2017. It has a more successful performance than CNN-LSTM Hybrid IDS in UNSW-NB15.

## IV. CONCLUSION

This study offers an effective solution to the urgent need for lightweight BERT-based IDS with the experimental results demonstrate substantial tradeoffs between model size, accuracy, F1-score and throughput in the resource-constrained environments. NF4 quantized model offers significant model size reduction by 91.6% decreasing from 251.01 MB to 21.11 MB, which is approximately 12 times smaller than the baseline model. Structured pruning with FP16 quantization method offers the highest performance for the environments where the accuracy and F1-score is most critical. For the environments where model size is of the highest concern, NF4 quantized model would be most appropriate despite lower throughput. Distillation based methods reduced the model size to 72.62 MB, but F1-scores decreased as a result. Further self-distillation had the highest F1-score among the distilled methods, but still fell behind the non-distilled methods in detection performance. The most important outcome was that using only structured pruning combined with FP16 quantization yielded the best compromise on all three metrics. These findings have significant implications. For the future work, the experiment structure which applied on the proposed system will generalize for IDS that can have contextual awareness with transformers in the resource-constrained environments.

## REFERENCES

[1] Sarika, "Investigation of machine learning-based software definition network for intrusion detection in iot," *Journal of Electrical Systems*, 2024.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[3] M. V. Deepa and D. N. Radha, "A survey on network intrusion system attacks classification using machine learning techniques," *IOP Conference Series: Materials Science and Engineering*, vol. 1022, 2021.

[4] D. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[5] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration*, ser. LISA '99. USA: USENIX Association, 1999, p. 229–238.

[6] H. R. Sayegh, W. Dong, and A. M. Al-madani, "Enhanced intrusion detection with lstm-based model, feature selection, and smote for imbalanced data," *Applied Sciences*, vol. 14, no. 2, 2024.

[7] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*. Canberra, Australia: IEEE, Nov. 2015, p. 1–6.

[8] A. Rosay, E. Cheval, F. Carlier, and P. Leroux, "Network intrusion detection: A comprehensive analysis of cic-ids2017," in *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP)*. SCITEPRESS, 2022, pp. 478–489.

[9] A. H. Halbouni, T. S. Gunawan, M. Halbouni, F. A. A. Assaig, M. R. Effendi, and N. Ismail, "Cnn-ids: Convolutional neural network for network intrusion detection system," in *2022 8th International Conference on Wireless and Telematics (ICWT)*, 2022, pp. 1–4.

[10] H. C. Altunay and Z. Albayrak, "A hybrid cnn+lstm-based intrusion detection system for industrial iot networks," *Engineering Science and Technology, an International Journal*, vol. 38, p. 101322, 2023.

[11] A. Sanmorino, S. Suryati, R. Gustriansyah, S. Puspasari, and N. Ariati, "Feature extraction vs fine-tuning for cyber intrusion detection model," *JURNAL INFOTEL*, vol. 16, no. 2, pp. 302–315, 2024.

[12] S. Choudhary and N. Kesswani, "Analysis of kdd-cup'99, nsl-kdd and unsw-nb15 datasets using deep learning in iot," *Procedia Computer Science*, vol. 167, pp. 1561–1573, 2020.

[13] P. RajkumarDheivanayahi, R. Berry, N. Costagliola, L. Fiondella, N. D. Bastian, and G. Kul, "Explainability of network intrusion detection using transformers: A packet-level approach," *IEEE Access*, 2024.

[14] Y. A. Farrukh, I. Khan, S. Wali, D. Bierbrauer, J. A. Pavlik, and N. D. Bastian, "Payload-byte: A tool for extracting and labeling packet capture files of modern network intrusion detection datasets," in *2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*. Vancouver, WA, USA: IEEE, Dec. 2022, p. 58–67.

[15] Y. Yang and X. Peng, "Bert-based network for intrusion detection system," *EURASIP Journal on Information Security*, vol. 2025, no. 11, 2025.

[16] S. Li, X. Ning, L. Wang, T. Liu, X. Shi, S. Yan, G. Dai, H. Yang, and Y. Wang, "Evaluating quantized large language models," in *Forty-first International Conference on Machine Learning*, 2024.

[17] Z. Liao, V. Quétu, V.-T. Nguyen, and E. Tartaglione, "Can unstructured pruning reduce the depth in deep neural networks?" in *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. IEEE, Oct. 2023, p. 1394–1398.

[18] E. J. Crowley, J. Turner, A. Storkey, and M. O'Boyle, "A closer look at structured pruning for neural network compression," 2019.

[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[20] S. Yang, X. Zheng, Z. Xu, and X. Wang, "A lightweight approach for network intrusion detection based on self-knowledge distillation," in *ICC 2023-IEEE International Conference on Communications*. IEEE, 2023, pp. 3000–3005.

[21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.