International Journal on Advanced Science Engineering Information Technology

Vol.15 (2025) No. 2
ISSN: 2088-5334

Comparative Study of Machine Learning Algorithms for DDoS Attack Detection in SDN Networks: A Carbon Emission Analysis with Hyperparameter Optimization Using Bayesian Optimization

Nadhir Fachrul Rozam [a,*], Handaru Jati [a], Eko Marpanaji [a]

[a] *Doctoral Program in Engineering, Faculty of Engineering, Universitas Negeri Yogyakarta, Indonesia*
Corresponding author: [*]nadhirfachrul.2024@student.uny.ac.id

Abstract*—The rising energy consumption of artificial intelligence (AI) models has sparked concerns about their environmental impact, particularly in high-computation fields like cybersecurity. As machine learning (ML) models become more complex and resource-intensive, optimizing their energy efficiency and sustainability has become a critical challenge. Bayesian Optimization has emerged as an effective approach for hyperparameter tuning, improving both model performance and energy efficiency. This study explores Tree-structured Parzen Estimators (TPE), a variant of Bayesian Optimization that models hyperparameter distributions using density estimation, to optimize the performance and environmental footprint of three widely used ML algorithms—Random Forest (RF), Support Vector Machine (SVM), and Extreme Gradient Boosting (XGBoost)—for DDoS attack detection in Software-Defined Networks (SDN). Evaluations on two datasets—Dataset 3 (binary classification) and Dataset 4 (multi-class classification)—analyze accuracy, precision, recall, and F1-score, alongside energy consumption and carbon emissions measured via the CodeCarbon. Results show that RF achieves the highest accuracy across both datasets (99.81%) while reducing carbon emissions by 44.6% after optimization of TPE. XGBoost, while slightly less accurate (99.77%), produces the lowest carbon emissions (0.0006 kg CO_2), demonstrating superior energy efficiency. SVM, despite a 35% reduction in emissions, remains the least efficient in energy consumption and exhibits lowest accuracy. These findings highlight the role of Bayesian Optimization in balancing predictive performance with sustainability. This study contributes by demonstrating a quantitative approach to evaluating the trade-off between accuracy and energy efficiency in ML-based DDoS attack detection in SDN, offering insights into selecting environmentally sustainable models.

***Keywords*— Carbon emission; energy consumption; DDoS attack detection; software-defined networks; machine learning; XGBoost.**

Manuscript received 18 Dec. 2024; revised 13 Jan. 2025; accepted 16 Mar. 2025. Date of publication 30 Apr. 2025.
IJASEIT is licensed under a Creative Commons Attribution-Share Alike 4.0 International License.

I. INTRODUCTION

Software-Defined Networking (SDN) has significantly transformed traditional network management by decoupling the control plane from the data plane, enabling more agile, scalable, and programmable networks. While this innovation offers substantial benefits in terms of flexibility and automation, it also introduces new security vulnerabilities, particularly in the form of Distributed Denial-of-Service (DDoS) attacks. DDoS attacks, which aim to overwhelm network resources and disrupt services, remain one of the most significant threats to SDN infrastructure [1]–[3]. These attacks can exploit the centralized nature of SDN control, making it particularly vulnerable to large-scale disruptions [4]–[6]. As SDN networks become more widely deployed in critical infrastructure, ensuring robust protection against such attacks has become paramount.

In recent years, machine learning (ML) algorithms have gained prominence for their ability to automate and enhance DDoS detection by analyzing patterns in network traffic. Various ML techniques, including Support Vector Machine (SVM), Random Forest (RF), and Extreme Gradient Boosting (XGBoost), have been explored for this purpose due to their proficiency in handling large, complex datasets and identifying hidden patterns in traffic behavior [7], [8] For example, SVM is known for its ability to perform binary classification tasks with high accuracy by constructing hyperplanes that separate different classes of network traffic [7], [9], [10]. On the other hand, Random Forest leverages an ensemble of decision trees to classify network traffic, which helps mitigate overfitting and enhances the model's robustness against varied attack patterns [11]–[13]. XGBoost, a gradient boosting algorithm, has become increasingly popular due to its efficiency in handling large datasets and its

superior predictive performance, particularly in scenarios involving imbalanced data like DDoS detection [14]. These algorithms have shown promise in improving the accuracy of DDoS attack detection, though each exhibits unique strengths and weaknesses depending on the specific characteristics of the network data.

While these traditional ML approaches have demonstrated promising results, recent advancements in deep learning have introduced new possibilities for improving DDoS detection in SDN. Deep learning techniques leverage their ability to automatically extract hierarchical features from raw network traffic, reducing reliance on manual feature engineering. Convolutional Neural Networks (CNNs) have been employed for traffic classification by capturing spatial dependencies within packet sequences [15]–[17], whereas Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have proven effective in modeling temporal attack patterns [18]–[20]. Additionally, Transformer-based architectures, including attention mechanisms and hybrid deep learning frameworks, are being explored for their scalability and robustness in handling high-dimensional SDN traffic data [21], [22].

However, as machine learning models continue to be adopted in practical applications, the environmental impact of these models has come under scrutiny. Training machine learning models, especially deep learning models, can be computationally intensive and energy-consuming, leading to a significant carbon footprint [23]–[25]. In the context of SDN, where models are often required to process vast amounts of network data in real-time, this concern becomes even more pressing. Studies have highlighted the substantial energy consumption associated with training these models and the need for sustainable AI practices that balance model performance with environmental responsibility [23], [26]–[29]. Reducing the energy consumption of machine learning models, particularly during training and hyperparameter optimization phases, is a critical step toward making these technologies more sustainable [30]–[32].

To address these dual challenges of detection accuracy and sustainability, this paper proposes the use of Bayesian optimization, implemented through the Optuna library [33], to optimize the hyperparameters of SVM, Random Forest, and XGBoost for DDoS attack detection in SDN networks. Bayesian optimization is particularly well-suited for optimizing complex, non-convex functions like those associated with machine learning models, as it intelligently explores the hyperparameter space and converges to optimal solutions more efficiently than traditional grid search or random search methods [33]. By incorporating Bayesian optimization, this study aims to enhance model performance while reducing computational cost, thereby mitigating the carbon footprint associated with model training. Furthermore, the carbon emissions of each algorithm will be measured during the training to evaluate their environmental impact, providing a holistic view of the trade-offs between model performance and sustainability.

Through this comprehensive analysis, we aim to identify the most efficient and sustainable approach for DDoS attack detection in SDN, offering valuable insights into how machine learning can be deployed more responsibly in real-world network security applications. This research contributes to the growing body of literature that emphasizes the importance of energy-efficient AI models, particularly in the context of cybersecurity, where both accuracy and sustainability are crucial for long-term success.

II. Materials and Methods

This section describes the materials, tools, and methodologies used to conduct the experiments. The study utilizes a dataset from previous research, including network traffic data relevant for identifying DDoS attacks, and applies advanced hyperparameter optimization techniques using Optuna. Additionally, the carbon footprint of the models is measured during both the training and evaluation to evaluate the environmental impact of the machine learning approaches used. The following subsections outline the dataset, algorithms, optimization procedures, and evaluation metrics used in this study.

A. Dataset

The dataset used for training and evaluating the machine learning models in this study is sourced from a previous research paper, which utilized network traffic data captured via the sFlow protocol. The dataset consists of four distinct datasets generated from real-time DDoS attack traffic and benign traffic in a Software-Defined Networking (SDN) environment. The traffic is sampled using the sFlow protocol to reduce network load and capture relevant features for attack detection. The datasets are categorized as follows:

- Dataset 1 and Dataset 2 contain 6,109 data points each, with traffic classified into two classes (normal and attack) and three classes (normal, Slowloris attack, and Hping3 attack), respectively.
- Dataset 3 and Dataset 4 contain 400,488 data points, categorized into two and three classes, respectively.

These datasets were preprocessed to handle missing values and normalized to ensure consistent scaling of input features. For this study, Dataset 3 and Dataset 4 were selected because they demonstrated the best performance in terms of classification accuracy and the ability to capture the diverse nature of DDoS attacks, as shown in the results of the previous research [2]. These datasets are more comprehensive and contain a greater variety of traffic patterns, making them ideal for training machine learning models in real-world SDN environments. The data from these datasets were preprocessed to handle missing values and normalized to ensure consistent scaling of input features. The data was then divided into training and testing sets with an 80/20 split, ensuring that the models could generalize well on unseen data. Each dataset contains various traffic flow characteristics such as source IP, destination IP, source port, destination port, IP protocol, and packet size, which are crucial for detecting DDoS attacks in SDN environments.

B. Machine Learning Algorithms

For this experiment, three machine learning models were selected to classify DDoS attacks in SDN environments: SVM, RF, and XGBoost. These algorithms were chosen due to their robustness and proven performance in classification tasks, particularly in high-dimensional data, such as network traffic. Below is a detailed explanation of each algorithm, along with the mathematical formulations used in their operation.

1) Support Vector Machine (SVM): Support Vector Machine (SVM) is a supervised learning algorithm used for classification tasks, including binary and multi-class classification. It works by finding the optimal hyperplane that maximizes the margin between two classes [34]. The goal of SVM is to find a decision boundary that best separates the data points of one class from those of the other class, thus ensuring optimal generalization.

Given a set of training data points $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, where $\vec{x}_i \in \Re^d$ are the feature vectors and $y_i \in \{-1, +1\}$ are the class labels, the SVM aims to find a hyperplane defined by Eq. (1)

$$w \cdot x + b = 0 \tag{1}$$

Where w is the normal vector to the hyperplane and b is the bias term.

The objective of SVM is to maximize the margin, defined as the distance between the hyperplane and the closest data points from both classes (also known as support vectors). This margin is given by Eq. (2).

$$\text{Margin} = \frac{2}{|w|} \tag{2}$$

The optimization problem is thus formulated Eq (3)

$$\min_{w,b} \quad \frac{1}{2}|w|^2 \tag{3}$$

subject to the constraints $y_i(w \cdot x_i + b) \geq 1$ *for all i=1,2,…,n*

This is a convex optimization problem, which can be solved using quadratic programming. In cases where data is not linearly separable, SVM uses a kernel trick to map the data into a higher-dimensional space where it becomes linearly separable. Popular kernels include the Radial Basis Function (RBF) kernel.

2) Random Forest (RF): Random Forest (RF) is an ensemble learning method that constructs multiple decision trees and aggregates their results to improve classification accuracy [35], [36]. Each tree is built using a random subset of the training data, and at each node, a random subset of features is considered for splitting. This randomization helps to reduce overfitting and increases the model's ability to generalize.

Random Forest builds an ensemble of T decision trees, where each tree T_k is trained using a random bootstrap sample D_k of the training data. The algorithm aggregates the results from all trees using majority voting for classification tasks. Given a new input x, the classification result is determined by Eq (4).

$$\hat{y} = \frac{1}{T}\sum_{k=1}^{T} h_k(x) \tag{4}$$

where:

- $h_k(x)$ is the prediction of the kkk-th tree,
- T is the total number of trees.

Each decision tree in the forest is constructed by recursively partitioning the feature space into regions where the data points in the same region are as homogeneous as possible in terms of the target variable. This is typically done by minimizing a criterion such as Gini impurity as in Eq (5) or entropy as in Eq (6) at each node.

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2 \tag{5}$$

where p_i is the proportion of the class i in the node t and K is the number of classes.

$$Entropy(t) = -\sum_{i=1}^{K} p_i \tag{6}$$

where $p_i$is the probability of class i at node t.

3) XGBoost (Extreme Gradient Boosting): XGBoost (Extreme Gradient Boosting) is an optimized gradient boosting algorithm that combines the predictive power of boosting with techniques to prevent overfitting. It builds an ensemble of decision trees sequentially, where each tree corrects the errors made by the previous ones. The model is trained by minimizing a loss function, and each subsequent tree adds weight to the mistakes of the prior trees [37].

The main objective of XGBoost is to minimize the following objective function in Eq (7).

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{7}$$

where:

- $l(y_i, \hat{y}_i)$is the loss function that measures the difference between the true label y_i and the predicted label $\hat{y}_i$,
- $\Omega(f_k)$ is a regularization term that penalizes the complexity of the model (i.e., the size of the trees) to avoid overfitting.

The regularization term is typically defined as Eq (8)

$$\Omega(f_k) = \gamma T_k + \frac{1}{2}||w_k||^2 \tag{8}$$

where:

- γ controls the complexity of the tree,
- T_k is the number of leaves in the k-th tree,
- w_k are the leaf weights.

The loss function is minimized using gradient descent. For each iteration, the model updates the weights of the trees to minimize the loss, making the model more accurate by correcting the errors of previous trees. XGBoost uses a second-order approximation to the loss function, improving the computational efficiency and enabling faster training compared to traditional gradient boosting methods [38]. This results in a significant performance boost, especially for large datasets.

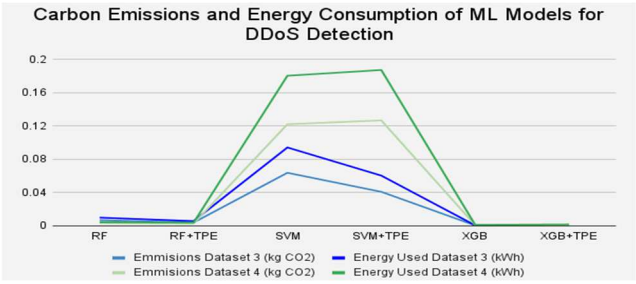

Fig. 1 Total Carbon Emissions and Energy Consumed for Each Method

C. Hyperparameter Optimization

To enhance the performance of each model, Optuna was utilized for hyperparameter optimization. It uses a probabilistic model to explore the hyperparameter space and efficiently identify the best configuration. The following hyperparameters were optimized for each algorithm:

1) SVM: The C, kernel, and gamma parameters were optimized using Optuna's search space, with a focus on the

radial basis function (RBF) kernel, as it showed the best results in preliminary experiments.

2) Random Forest: Hyperparameters such as n_estimators, max_features, and max_depth were optimized to find the optimal value.

3) XGBoost: Key parameters like n_estimators, learning_rate, and max_depth were tuned to improve classification accuracy while controlling overfitting.

Optuna implements Bayesian Optimization using Tree-structured Parzen Estimators (TPE). This optimization method minimizes the search space while maximizing the performance metrics. The optimization process also helps reduce computational time and resource usage, contributing to lower energy consumption and carbon footprint, which is an essential aspect of this study.

The Tree-structured Parzen Estimators (TPE) method is a probabilistic model-based optimization technique used for Bayesian Optimization [39]–[41]. It models the distribution of objective function values over different hyperparameter configurations in a way that allows efficient exploration of the search space. The key idea behind TPE is to model the hyperparameter distribution separately for "good" and "bad" configurations based on the objective function [42]The good configurations result in high performance (e.g., high accuracy or low error), and the bad configurations result in low performance.

The mathematical foundation of TPE relies on the ratio of the densities of good and bad hyperparameters [43], [44]. Maximizing this ratio for the next hyperparameter set selection is the main goal. The objective is to find the set of hyperparameters x that maximizes the following utility function in Eq (9).

$$\text{Utility}(x) = \frac{p(x|y > \theta)}{p(x|y \leq \theta)} \quad (9)$$

where:

- $p(x|y > \theta)$ is the estimated probability density of the hyperparameters given that the performance is greater than a threshold θ.
- $p(x|y \leq \theta)$ is the *estimated* probability density of the hyperparameters given that the performance is less than or equal to θ.

By modeling these distributions for each hyperparameter configuration, TPE efficiently searches the hyperparameter space and selects configurations that are likely to yield good results.

D. Carbon Footprint Measurement

The carbon footprint of each model was tracked during the training using the CodeCarbon library [45]. CodeCarbon calculates energy consumption based on the hardware resources (CPU, GPU) used during model training and estimates carbon emissions in kilograms of CO_2 equivalent. Energy consumption was measured for each training session, and total carbon emissions were calculated based on the electricity consumed.

The energy consumed during computation is converted into carbon emissions based on the energy grid's carbon intensity. Carbon intensity is the amount of CO_2 emitted per unit of energy produced (usually measured in g CO_2 per kWh). This value varies depending on the location, as different regions have different energy mixes (renewable vs. fossil fuels). The carbon emissions C_{CO2} can be calculated as in Eq (10).

$$C_{CO2} = E \times \text{carbon intensity} \quad (10)$$

where:

- C_{CO2} is the carbon emissions (in grams of CO_2, g CO_2),
- E is the energy consumed (in kWh),
- carbon intensity is the emission factor (in g CO_2 per kWh).

TABLE I
EVALUATION RESULTS

	RF		RF+TPE		SVM		SVM+TPE		XGB		XGB+TPE	
Dataset	3	4	3	4	3	4	3	4	3	4	3	4
Accuracy	99.81	99.81	99.81	99.81	99.45	99.46	99.47	81.79	99.77	99.76	99.77	99.76
Precision	99.79	99.83	99.78	99.83	99.21	99.50	99.26	54.43	99.68	99.78	99.70	99.78
Recall	99.82	99.84	99.83	99.84	99.67	99.55	99.67	66.26	99.86	99.82	99.84	99.82
F1	99.81	99.84	99.80	99.83	99.44	99.53	99.47	59.03	99.77	99.80	99.77	99.80

Since carbon emissions are often expressed in kilograms (kg) rather than grams, the value of C_{CO2} can be converted to kg by dividing by 1000. The goal of tracking the carbon footprint was to evaluate the environmental impact of machine learning models during their optimization. This allowed for a comparison between the performance of the models and their associated environmental costs. The results of this comparison are essential for identifying sustainable approaches to DDoS detection in SDN networks.

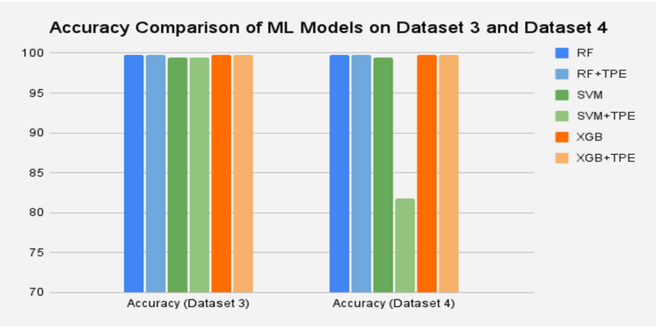

Fig. 2 Accuracy Comparison of ML Models

III. Results and Discussion

This section presents the results of the machine learning models used for DDoS attack detection in SDN networks. These models were trained using Datasets 3 and 4 from [2]We compare their carbon emissions, energy consumption, and training duration. Additionally, we analyze the impact of Optuna optimization on the models' performance and environmental sustainability.

A. Overview of Experimental Setup

This study evaluated three machine learning algorithms—SVM, RF, and XGBoost—for DDoS attack detection in SDN networks. These algorithms were selected for their diverse capabilities in classification tasks, and they were tested using two different datasets: Dataset 3 and Dataset 4. Dataset 3 and Dataset 4 contain 400,488 data points, categorized into two and three classes, respectively. Dataset 3 classes are Benign Traffic (label 0) and DDoS Traffic (label 1), while Dataset 4 classes are Benign Traffic (label 0), Slowloris attack (label 1), and Hping3 [46] attack (label 2)

The models were evaluated both with and without Optuna optimization. Optuna was used to tune the hyperparameters of the models, aiming to improve their performance while reducing computational costs, including carbon emissions and energy consumption. The key performance metrics used to assess the models included accuracy, precision, recall, and F1-score. In addition to these performance metrics, we also measured carbon emissions and energy consumption during the training of the models to evaluate their environmental impact.

There are different parameters for each algorithm. Each parameter combination will be searched for the best parameter with GridSearchCV (5 fold validation) and Optuna (60 trials). In the Random Forest, the hyperparameter used to find the best combination of parameters are as follows:

- N Estimator: 25,50,75, and 100
- Max Feature: sqrt, log2, and None

For Support Vector Machine, the hyperparameters used are as follows:

- Kernel: Linear, RBF, and Polynomial
- C: 0.1, 0.5, 1, 5, and 10

For Xtreme Gradient Boosting, the hyperparameters used are as follows:

- N Estimator: 25, 50, 75, and 100
- Learning Rate: 0.1, 0.5, and 1

To evaluate the energy consumption and carbon emissions during the training phase, we utilized CodeCarbon version 2.8.1 on a system with the following specifications: Intel® Core™ i5-10400 CPU @ 2.90GHz (12 cores), 32 GB RAM, and an NVIDIA GeForce GTX 1060 (6GB) GPU. The experiments were conducted on Windows 11 (version 10.0.26100-SP0) with Python 3.12.8 as the runtime environment. The carbon footprint measurements were obtained using CodeCarbon's estimation methodology, which considers power consumption from both the CPU and GPU while accounting for regional carbon intensity in Yogyakarta, Indonesia, where the experiments were conducted.

By conducting these experiments, we aimed to identify the most efficient model in terms of both performance and environmental sustainability, considering the trade-offs between computational resources and detection accuracy. The goal of the experiments was twofold:

- Comparing the performance of SVM, Random Forest, and XGBoost with and without Optuna optimization.
- Evaluating the environmental sustainability of these models by measuring their carbon emissions and energy consumption during training.

TABLE II
EMISSION RESULTS

	RF		**RF+TPE**		**SVM**		**SVM+TPE**		**XGB**		**XGB+TPE**	
Dataset	3	4	3	4	3	4	3	4	3	4	3	4
Emission (kg CO2)	0.0065	0.0027	0,0036	0.0023	0.0636	0.1221	0.0407	0.1267	0.0002	0.0006	0.0003	0.0009
Training Duration (s)	662.94	277.18	373.87	244.68	6503.68	12441.01	4157.41	12932.25	22.54	68.11	30.73	97.36
Energy Used (kWh)	0.0096	0.0040	0.0054	0.0035	0.0941	0.1805	0.0602	0.1874	0.0003	0.0009	0.0004	0.0014

B. Performance Comparison Across Models

This section compares the performance of RF, SVM, and XGBoost models on Dataset 3 and Dataset 4 using four key metrics: Accuracy, Precision, Recall, and F1-score. We also assess the impact of TPE optimization on model performance. Details of each model's performance results can be seen in Table I as a reference in the discussion.

For Dataset 3, RF demonstrated the best performance across all metrics, achieving 99.81% accuracy. TPE optimization had no significant effect on RF's performance, with the optimized model showing the same results. SVM performed slightly lower, with 99.45% accuracy. After applying TPE optimization, SVM showed a slight improvement in precision and recall. XGBoost performed well, with 99.77% accuracy for Dataset 3. TPE optimization had minimal impact on XGBoost's performance, as it maintained the same score.

On Dataset 4, RF continued to perform strongly, achieving 99.81% accuracy. TPE optimization again did not significantly affect RF's performance, with the optimized model maintaining the same score on all metrics. SVM showed 99.46% accuracy. After applying TPE optimization, SVM showed a significant drop in performance, with 81.79% accuracy. This indicates that TPE optimization caused a substantial decline in SVM's performance on Dataset 4. XGBoost initially performed well on Dataset 3, with TPE optimization providing slight improvements. On Dataset 4, XGBoost showed a noticeable improvement with TPE optimization, particularly in recall and precision.

When comparing Dataset 3 and Dataset 4, as shown in Fig. 2, RF consistently outperformed the other models in both

datasets, with minimal impact from TPE optimization. This indicates the robustness of RF in handling different types of data. SVM performed well on Dataset 3, but TPE optimization led to a significant drop in accuracy (81.79%) and precision (54.43%) on Dataset 4, indicating that TPE optimization hurt SVM for Dataset 4. This decline in accuracy also means that SVM might be less adaptable or sensitive to the variations in Dataset 4, possibly due to the inherent complexity of SVM's decision boundaries.

Overall, TPE optimization has not significantly improved the XGBoost in performance measurement. RF maintained excellent performance across both datasets, while SVM showed moderate improvements for Dataset 3 but experienced a performance decline in accuracy and precision on Dataset 4 after TPE optimization.

C. Carbon Emissions and Energy Consumption

In this section, we evaluate the carbon emissions and energy consumption during the model training. The results highlight the environmental impact of each model's training process and the effects of applying TPE optimization on the energy usage and carbon footprint. As previously explained, this emission calculation can be different for each region. This research was conducted in Yogyakarta, Indonesia so that carbon emissions were adjusted to the Carbon Intensity data in this area.

Table II shows a linear relationship between carbon emissions, energy usage, and training duration. The longer the training duration, the more energy consumption will increase, and of course, the carbon emissions will increase. Some algorithms that apply TPE can reduce training duration because they can find optimal hyperparameters more quickly.

As shown in Fig. 1, SVM is the highest contributor to carbon emissions, with a significant total of 0.0636 kg CO2. This is primarily due to the computational complexity of the SVM algorithm, which requires substantial processing power during training. Applying TPE on SVM reduces 35% of the carbon emission to 0.0407. However, it still becomes the biggest number among the others. On the other hand, RF shows much lower carbon emissions, at 0.0065 kg CO2, indicating a lower environmental impact than SVM. Similarly, applying TPE optimization to RF (RF+TPE) results in 44.6% reduced carbon emissions (0.0036 kg CO2), which becomes the most efficient decrease due to the optimization. For XGBoost, the emissions are even lower, at 0.0002070 kg CO2 for the base model, and 0.0002701 kg CO2 after applying TPE optimization (XGB+TPE), showing a similar trend of minimal impact on carbon emissions after the optimization.

Still in Fig. 1, SVM also consumes the most energy, totaling 0.0941 kWh during training. This high energy consumption is consistent with the large computational requirements for SVM model training. RF and RF+TPE show much lower energy consumption, with 0.009649 kWh and 0.005420 kWh, respectively. The TPE optimization applied to RF reduces energy use, which highlights the optimization's ability to enhance computational efficiency.

In short, as seen in Table II, the results indicate that SVM has the highest carbon emissions and energy consumption, significantly impacting the environment during training. On the other hand, RF and XGB models, especially with TPE optimization, show lower carbon emissions and energy consumption, with RF exhibiting the most notable reduction in both metrics after optimization.

D. Optimization Impact on Performance and Sustainability

TPE optimization generally reduced RF energy consumption and carbon emissions, while maintaining high performance metrics. These reductions show that TPE optimization can significantly improve the sustainability of RF by decreasing its environmental impact while maintaining performance. Despite showing performance drops, SVM benefited from TPE optimization by reducing carbon emissions and energy usage compared to its non-optimized version. TPE helps make SVM more efficient, but SVM still consumes substantial resources compared to RF and XGBoost. This decline suggests that SVM may not be as sensitive to the TPE optimization process, or the optimization process did not improve SVM's performance for the more complex Dataset 4. The results indicate that SVM might benefit from alternative optimization methods better suited to its structure. XGBoost showed consistently low emissions and energy consumption, with TPE optimization further enhancing recall and precision without significant environmental impact. TPE optimization is most effective for XGBoost and RF in improving both performance and sustainability.

TPE optimization plays a vital role in enhancing the performance and sustainability of machine learning models. RF and XGBoost showed positive improvements in carbon emissions and energy consumption due to TPE optimization, making them more efficient models in terms of performance and environmental impact. SVM, however, experienced a performance decline on Dataset 4 and showed limited gains in sustainability despite TPE optimization. This highlights the potential for further optimization research and techniques, particularly for models like SVM, which may benefit from other tuning methods.

IV. Conclusion

In this study, we compared the performance and environmental impact of three machine learning models: RF, SVM, and XGBoost, across two datasets (Dataset 3 and Dataset 4). The primary objective was to evaluate TPE optimization's effectiveness in improving model performance and sustainability by reducing carbon emissions and energy consumption.

RF consistently performed well across both datasets, with minimal impact from TPE optimization. While TPE did not significantly improve RF's performance, it helped reduce training time and energy consumption, making RF a highly efficient model for both performance and sustainability. XGBoost demonstrated high performance with 99.77% accuracy and low carbon emissions and energy consumption across both datasets. TPE optimization resulted in slight improvements in recall and precision, further enhancing the efficiency of XGB without negatively affecting its environmental footprint. SVM demonstrated the highest emissions and energy consumption, particularly on Dataset 4, where carbon emissions were significantly higher (0.1221 kg CO2 without optimization) than the other models. In terms of performance, SVM also has the lowest among the others. Therefore, it is not recommended to use SVM in case of

DDOS attack detection in SDN networks, or at least it can be tried with other datasets or other optimization methods.

This study suggests that a significant drop in performance for SVM on Dataset 4 after TPE optimization indicates that SVM may not fully benefit from TPE for certain types of datasets. Future research could explore alternative optimization techniques or a more tailored hyperparameter tuning approach to better suit the needs of SVM, especially for complex datasets. While TPE optimization showed some positive effects on energy consumption and carbon emissions, further studies could focus on comparing TPE with other optimization methods, such as Bayesian Optimization or Genetic Algorithms, to assess which method provides the most significant reduction in both training time and environmental impact.

Although XGBoost and RF showed good results with TPE optimization, exploring the effects of TPE on deep learning models or ensemble methods could provide new insights into the scalability and applicability of TPE optimization across different algorithms. Future work should also consider the real-world application of these findings in resource-constrained environments. Research could explore the potential for these optimized models to be deployed on edge devices or cloud infrastructures, focusing on the trade-off between model performance, environmental impact, and computational costs. Incorporating Sustainability Metrics into Model Selection: The growing importance of sustainable AI calls for integrating ecological impact into model selection criteria. Future research could propose frameworks that combine traditional performance metrics with carbon emissions and energy consumption, aiding researchers and practitioners in making more environmentally responsible decisions when choosing deployment models.

This study highlights the dual benefits of TPE optimization: improving the performance of models like XGBoost and RF while reducing their carbon emissions and energy consumption. However, the performance of models like SVM suggests that the impact of TPE optimization can vary significantly across different algorithms and datasets. As machine learning models become more widely used, the need for sustainable AI practices will only grow, and future research should continue to explore methods that enhance model performance and environmental responsibility.

REFERENCES

[1] S. Chattopadhyay, A. K. Sahoo, S. Jasola, and T. Choudhury, "Detection of DDoS Attacks in SDN Using Machine Learning Approaches: A Review," *Lect. Notes Networks Syst.*, vol. 1025, pp. 295–305, 2024, doi: 10.1007/978-981-97-3594-5_24.

[2] N. F. Rozam and M. Riasetiawan, "XGBoost Classifier for DDOS Attack Detection in Software Defined Network Using sFlow Protocol," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 13, no. 2, pp. 718–725, Apr. 2023, doi: 10.18517/ijaseit.13.2.17810.

[3] M. A. Setitra, M. Fan, I. Benkhaddra, and Z. E. A. Bensalem, "DoS/DDoS attacks in Software Defined Networks: Current situation, challenges and future directions," *Computer Communications*, vol. 222. Elsevier B.V., pp. 77–96, Jun. 01, 2024, doi:10.1016/j.comcom.2024.04.035.

[4] Y. Cui *et al.*, "Towards DDoS detection mechanisms in Software-Defined Networking," *J. Netw. Comput. Appl.*, vol. 190, no. November 2020, p. 103156, 2021, doi: 10.1016/j.jnca.2021.103156.

[5] N. S. S. Shaji, R. Muthalagu, and P. M. Pawar, "SD-IIDS: intelligent intrusion detection system for software-defined networks," *Multimed. Tools Appl.*, vol. 83, no. 4, pp. 11077–11109, Jan. 2024, doi:10.1007/s11042-023-15725-y.

[6] N. T. Cam and T. D. Viet, "uitSDD: Protect software defined networks from distributed denial-of-service using multi machine learning models," *Cluster Comput.*, vol. 28, no. 1, p. 11, Feb. 2025, doi:10.1007/s10586-024-04757-0.

[7] R. Ma, Q. Wang, X. Bu, and X. Chen, "Real-Time Detection of DDoS Attacks Based on Random Forest in SDN," *Appl. Sci.*, vol. 13, no. 13, 2023, doi: 10.3390/app13137872.

[8] H. Kheddar, D. W. Dawoud, A. I. Awad, Y. Himeur, and M. K. Khan, "Reinforcement-Learning-Based Intrusion Detection in Communication Networks: A Review," *IEEE Commun. Surv. Tutorials*, vol. PP, no. 00, pp. 1–1, 2024, doi:10.1109/comst.2024.3484491.

[9] Y. Wang, X. Wang, M. M. Ariffin, M. Abolfathi, A. Alqhatani, and L. Almutairi, "Attack detection analysis in software-defined networks using various machine learning method," *Comput. Electr. Eng.*, vol. 108, 2023, doi: 10.1016/j.compeleceng.2023.108655.

[10] W. Man, G. Yang, and S. Feng, "Joint Selfattention-SVM DDoS Attack Detection and Defense Mechanism Based on Self-Attention Mechanism and SVM Classification for SDN Networks," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E107.A, no. 6, pp. 881–889, 2024, doi: 10.1587/transfun.2023eap1057.

[11] R. Ma, Q. Wang, X. Bu, and X. Chen, "Real-Time Detection of DDoS Attacks Based on Random Forest in SDN," *Appl. Sci. 2023, Vol. 13, Page 7872*, vol. 13, no. 13, p. 7872, Jul. 2023, doi:10.3390/app13137872.

[12] A. A. Najar and S. Manohar Naik, "DDoS attack detection using MLP and Random Forest Algorithms," *Int. J. Inf. Technol.*, vol. 14, no. 5, pp. 2317–2327, 2022, doi: 10.1007/s41870-022-01003-x.

[13] H. Nurwarsito and M. F. Nadhif, "DDoS Attack Early Detection and Mitigation System on SDN using Random Forest Algorithm and Ryu Framework," in *2021 8th International Conference on Computer and Communication Engineering (ICCCE)*, 2021, pp. 178–183, doi:10.1109/iccce50029.2021.9467167.

[14] T. Arvind and K. Radhika, "XGBoost Machine Learning Model-Based DDoS Attack Detection and Mitigation in an SDN Environment," *Int. J. Eng. Trends Technol.*, vol. 71, no. 2, pp. 349–361, 2023, doi:10.14445/22315381/ijett-v71i2p237.

[15] R. Ben Said, Z. Sabir, and I. Askerzade, "CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking with Hybrid Feature Selection," *IEEE Access*, vol. 11, pp. 138732–138747, 2023, doi:10.1109/access.2023.3340142.

[16] A. A. Najar and S. Manohar Naik, "Cyber-Secure SDN: A CNN-Based Approach for Efficient Detection and Mitigation of DDoS attacks," *Comput. Secur.*, vol. 139, 2024, doi:10.1016/j.cose.2024.103716.

[17] O. Polat *et al.*, "Multi-Stage Learning Framework Using Convolutional Neural Network and Decision Tree-Based Classification for Detection of DDoS Pandemic Attacks in SDN-Based SCADA Systems," *Sensors*, vol. 24, no. 3, 2024, doi:10.3390/s24031040.

[18] R. Priyadarshini and R. K. Barik, "A deep learning based intelligent framework to mitigate DDoS attack in fog environment," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 3, pp. 825–831, 2022, doi:10.1016/j.jksuci.2019.04.010.

[19] N. M. Yungaicela-Naula, C. Vargas-Rosales, J. A. Perez-Diaz, E. Jacob, and C. Martinez-Cagnazzo, "Physical Assessment of an SDN-Based Security Framework for DDoS Attack Mitigation: Introducing the SDN-SlowRate-DDoS Dataset," *IEEE Access*, vol. 11, pp. 46820–46831, 2023, doi: 10.1109/access.2023.3274577.

[20] D. G. Narayan, W. Heena, and K. Amit, "A Collaborative Approach to Detecting DDoS Attacks in SDN Using Entropy and Deep Learning," *J. Telecommun. Inf. Technol.*, no. 3, pp. 79–87, 2024, doi:10.26636/jtit.2024.3.1609.

[21] H. Wang and W. Li, "DDosTC: A transformer-based network attack detection hybrid mechanism in SDN," *Sensors*, vol. 21, no. 15, 2021, doi: 10.3390/s21155047.

[22] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, "FlowTransformer: A transformer framework for flow-based network intrusion detection systems," *Expert Syst. Appl.*, vol. 241, p. 122564, May 2024, doi:10.1016/j.eswa.2023.122564.

[23] D. Patterson *et al.*, "The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink," *Computer (Long. Beach. Calif).*, vol. 55, no. 7, pp. 18–28, Jul. 2022, doi: 10.1109/MC.2022.3148714.

[24] Z. Fan, Z. Yan, and S. Wen, "Deep Learning and Artificial Intelligence in Sustainability: A Review of SDGs, Renewable Energy, and Environmental Health," *Sustain. 2023, Vol. 15, Page 13493*, vol. 15, no. 18, p. 13493, Sep. 2023, doi: 10.3390/SU151813493.

[25] T. Adem, A. Mccrabb, V. Goyal, and V. Bertacco, “Evergreen: Comprehensive Carbon Model for Performance-Emission Tradeoffs,” *Proc. - 2024 IEEE Int. Symp. Workload Charact. IISWC 2024*, pp. 132–143, 2024, doi: 10.1109/IISWC63097.2024.00021.

[26] D. Patterson *et al.*, “Carbon Emissions and Large Neural Network Training,” pp. 1–22, Apr. 2021, doi: 10.48550/arXiv.2104.10350.

[27] R. Nishant, M. Kennedy, and J. Corbett, “Artificial intelligence for sustainability: Challenges, opportunities, and a research agenda,” *Int. J. Inf. Manage.*, vol. 53, no. January, p. 102104, 2020, doi:10.1016/j.ijinfomgt.2020.102104.

[28] V. Mehra *et al.*, “Impacts of digital technologies and social media platforms on advocating environmental sustainability in sports sector,” *Discov. Sustain.*, vol. 6, no. 1, 2025, doi: 10.1007/s43621-025-00932-4.

[29] F. Abdelfattah, K. Dahleez, H. Al Halbusi, and M. Salah, “Strategic green alliances: Integrating green dynamic capabilities, AI, and electronic entrepreneurial innovation for sustainability,” *Sustain. Futur.*, vol. 9, no. January, p. 100433, 2025, doi:10.1016/j.sftr.2025.100433.

[30] X. Wang *et al.*, “A Novel Energy Saving Algorithm for Network Deep Learning Tasks,” *2024 6th Int. Conf. Next Gener. Data-Driven Networks, NGDN 2024*, pp. 339–343, 2024, doi:10.1109/ngdn61651.2024.10744084.

[31] T. S. Reddy and B. M. Beena, “Analysis of Green Computing Models on AWS Using Machine Learning Algorithms,” *2024 1st Int. Conf. Women Comput. InCoWoCo 2024 - Proc.*, pp. 1–6, 2024, doi:10.1109/InCoWoCo64194.2024.10863061.

[32] S. Sarkar *et al.*, “Carbon Footprint Reduction for Sustainable Data Centers in Real-Time,” *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 20, pp. 22322–22330, 2024, doi: 10.1609/aaai.v38i20.30238.

[33] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A Next-generation Hyperparameter Optimization Framework,” *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 2623–2631, Jul. 2019, doi: 10.1145/3292500.3330701.

[34] Z. Liu, Y. Wang, F. Feng, Y. Liu, Z. Li, and Y. Shan, “A DDoS Detection Method Based on Feature Engineering and Machine Learning in Software-Defined Networks,” *Sensors*, vol. 23, no. 13, 2023, doi: 10.3390/s23136176.

[35] S. Bhadauria, N. Aildasani, J. P. Kushwaha, and H. Gauttam, “DDoS Attacks Detection using Ensemble Learning,” *2024 15th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2024*, pp. 1–6, 2024, doi:10.1109/icccnt61001.2024.10724315.

[36] W. Zhang, G. Yang, and R. Zhang, “DDoS Attack Detection Based on Rényi-RF in SDN Environment,” in *2024 IEEE 7th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Sep. 2024, vol. 46, no. 4, pp. 1365–1369, doi:10.1109/itnec60942.2024.10733276.

[37] H. A. Alamri and V. Thayananthan, “Bandwidth control mechanism and extreme gradient boosting algorithm for protecting software-defined networks against DDoS attacks,” *IEEE Access*, vol. 8, pp. 194269–194288, 2020, doi: 10.1109/access.2020.3033942.

[38] G. B. N. Rao, A. Kumar, N. Kumar, and P. Raj, “Efficient Intelligent Network Intrusion Detection for SDN Using XGBoost,” in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, pp. 1–9, doi:10.1109/icccnt61001.2024.10723841.

[39] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi, “Multiobjective tree-structured parzen estimator for computationally expensive optimization problems,” *GECCO 2020 - Proc. 2020 Genet. Evol. Comput. Conf.*, pp. 533–541, Jun. 2020, doi:10.1145/3377930.3389817.

[40] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, “Taking the human out of the loop: A review of Bayesian optimization,” *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, 2016, doi:10.1109/jproc.2015.2494218.

[41] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, “Recent Advances in Bayesian Optimization,” *ACM Comput. Surv.*, vol. 55, no. 13s, Dec. 2023, doi: 10.1145/3582078.

[42] T. T. Khoei, S. Ismail, and N. Kaabouch, “Boosting-based Models with Tree-structured Parzen Estimator Optimization to Detect Intrusion Attacks on Smart Grid,” in *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2021, pp. 165–170, doi:10.1109/uemcon53757.2021.9666607.

[43] S. Watanabe, "Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance," *arXiv*, preprint arXiv:2304.11127, 2023. [Online]. Available: https://arxiv.org/abs/2304.11127

[44] J. Li *et al.*, “Water quality soft-sensor prediction in anaerobic process using deep neural network optimized by Tree-structured Parzen Estimator,” *Front. Environ. Sci. Eng.*, vol. 17, no. 6, 2023, doi:10.1007/s11783-023-1667-3.

[45] B. Courty *et al.*, “mlco2/codecarbon: v2.4.1.” Zenodo, May 2024, doi:10.5281/zenodo.11171501.

[46] A. Mehra and S. Badotra, “A Novel Framework for Prevention against DDoS Attacks using Software Defined-machine Learning Model,” *Int. J. Performability Eng.*, vol. 18, no. 8, pp. 580–588, 2022, doi:10.23940/ijpe.22.08.p6.580588.