Intrusion Detection System Using Machine Learning Techniques

1st Ranjith Kumar Reddy Dorsila
Computer Science and Engineering
Wright State University Dayton, Ohio
dorsila.2@wright.edu

2nd Maheep Chowdary Malapati
Computer Science and Engineering
Wright State University Dayton, Ohio
malapati.8@wright.edu

3rd Rajesh Galla
Computer Science and Engineering
Wright State University Dayton, Ohio
galla.5@wright.edu

4th Rekha Poosa
Computer Science and Engineering
Wright State University, Dayton, Ohio
poosa.2@wright.edu

5th Lakshman Kumar Vanukuri
Computer Science and Engineering
Wright State University Dayton, Ohio
vanukuri.6@wright.edu

Abstract* - *In the domain of network security, Intrusion Detection Systems (IDS) stand as a foundational element, meticulously surveilling and thwarting unauthorized entry and suspicious behaviors within computer networks. Diverse in approach, IDS strategies encompass signature-based detection, anomaly-based detection, and the increasingly prevalent machine learning-based detection methods. Among these, machine learning-driven IDS have ascended in prominence, acclaimed for their capacity to assimilate data and dynamically adjust against emerging threats. Harnessing the potency of artificial intelligence, these systems exhibit a remarkable ability to sift through expansive data troves, swiftly identifying irregularities and potential threats in real-time. Their adaptability and learning prowess enable them to evolve alongside the ever-evolving threat landscape, rendering them an invaluable asset in fortifying network defenses. The adeptness of machine learning-based IDS lies in their aptitude to decipher complex patterns, discerning anomalies that might otherwise elude conventional detection methods. Through continuous analysis and learning from historical and ongoing data, these systems possess the acumen to discern normal network behaviors from aberrations, thus proactively identifying and mitigating potential security breaches. Their agility and sophistication mark them as indispensable guardians in the ceaseless battle against network vulnerabilities, reaffirming their stature as a cornerstone in safeguarding the integrity of computer networks.

I. INTRODUCTION

The combination of Machine Learning and Intrusion Detection Systems (IDS) stands as a crucial measure for strengthening cybersecurity amid constantly evolving threats. Machine Learning empowers IDS with adaptability, allowing it to identify patterns, improve threat detection accuracy, and respond effectively to emerging attack methods. Various ML techniques, such as supervised and unsupervised learning, significantly enhance the resilience of IDS. This collaboration enables organizations to proactively tackle the ever-changing landscape of cyber threats, staying ahead in the race against potential breaches.

This research investigates the intricate relationship between IDS and Machine Learning, exploring fundamental concepts, types of ML algorithms, and their practical applications, illuminating an advanced cybersecurity approach.[1] It also examines the challenges in implementation, potential benefits, and avenues for future progress. In a complex digital threat environment, merging IDS with Machine Learning forms a robust defense mechanism, contributing to ongoing discussions on reinforcing cybersecurity against persistent and intricate threats.

II. BACKGROUND

The study seeks to explore the profound impact of integrating machine learning into IDS, shedding light on the advantages, challenges, and practical implications of this advanced approach. It will delve into the foundational concepts behind IDS, elucidate the workings of machine learning algorithms within these systems, and showcase real-world applications.

Moreover, the research aims to outline potential avenues for further advancements in this domain and address implementation challenges faced by organizations aiming to adopt machine learning-driven IDS.[2]

By examining the synergistic relationship between machine learning and IDS, this paper strives to contribute to the ongoing discourse on fortifying network security amidst an ever-evolving threat landscape. The insights gleaned from this exploration aim to provide a comprehensive understanding of the capabilities, limitations, and future potential of machine learning-enhanced IDS, paving the way for more robust and adaptive network defense strategies.

A. LITERATURE REVIEW

Machine learning-infused IDS have risen in prominence, owing to their capacity to glean insights from data and swiftly evolve against emergent threats. Leveraging the prowess of artificial intelligence, these systems exhibit unparalleled agility in scrutinizing extensive datasets, swiftly identifying irregularities and potential threats in real-time.[3]

Literature surrounding IDS underscores their fundamental role in network protection, highlighting traditional methods' efficacy while accentuating the transformative potential of machine learning integration. The surge in research focuses on refining machine learning algorithms, exploring novel techniques to enhance IDS accuracy and responsiveness. Studies showcase the practical implications of machine learning-based IDS, unveiling their ability to discern intricate patterns and anomalies that traditional methods might overlook.[4] Moreover, these reviews underscore challenges in implementation, emphasizing the need for robust data handling and model interpretability while advocating for ongoing advancements in this symbiotic integration.[5]

The literature reverberates the pivotal role of IDS in network security, accentuating the progressive shift toward machine learning integration as a promising frontier to fortify defenses against the ever-evolving landscape of cyber threats.

III. EXPERIMENTAL SETUP

This experimental setup aims to develop an Intrusion Detection System (IDS) using machine learning algorithms and compare performance between CPU and GPU environments. Leveraging the KDD Cup 1999 dataset from datahub.io, tailored for network intrusion detection, the data preprocessing phase involves handling missing values, categorical feature encoding using one-hot encoding, and normalization/standardization of numerical attributes.[6]

The selection of classification algorithms suitable for intrusion detection, such as Naïve Bayes, K-Nearest Neighbors, Random Forest classifier, and Decision Tree, forms the basis for subsequent model training. Using Python, Scikit-learn, and Joblib in the CPU environment (Jupyter Notebook), the chosen algorithms undergo training, with parallelized computation via Joblib to optimize multicore processing for faster performance.

Moving to the GPU environment, Google Colab's T4GPU is employed for accelerated computation, utilizing the cuML library from RAPIDS for consistency in algorithm implementation. The experimental procedure entails dataset preprocessing for training readiness, followed by separate training sessions for the selected algorithms on both CPU and GPU.

Performance evaluation measures encompass metrics like accuracy, precision, recall, and F1-score, enabling a comprehensive comparison between CPU and GPU executions. Additionally, the time taken for training each algorithm in both environments is meticulously measured and juxtaposed for a detailed analysis. The hardware and software specifications include Jupyter Notebook, Python, Scikit-learn, and Joblib for the CPU environment, while Google Colab's T4GPU with the cuML library caters to the GPU environment. The evaluation criteria prioritize performance metrics and training time to discern the effectiveness of the models in CPU versus GPU settings.[7]

The conclusive phase entails thorough documentation and analysis of the obtained results from both CPU and GPU executions. A comprehensive comparison of performance metrics and training times for each algorithm serves as the cornerstone for assessing the efficiency and efficacy of the IDS in different computational environments.

IV. METHODOLOGY

The "FEASEL(p1, p2, p3, data)" function we've built tries different combinations of features to find the best ones for our model. It uses logistic regression to measure how well these features work together,

focusing on accuracy as a way to judge their performance.

In the first phase using our computer's processor (CPU), we test various algorithms like Naïve Bayes, K-Nearest Neighbors, Random Forest, and Decision Tree. We analyze how well they perform using metrics such as Accuracy, Precision, Recall, and F1-Score after preparing our data.

We then split our prepared data into parts for training and testing the models. We teach our models with the training part and then test their abilities using the other part. To save time during this training, we use a technique called parallel processing through a tool called "joblib." This helps speed up the training without compromising the quality of our models.

Next, we shift gears to use the graphics card (GPU) to speed up our computations. By employing RAPIDS' CUML, which is a powerful tool that works similarly to Scikit-learn but uses the GPU's power, we notice a huge speed increase. However, although the GPU works much faster than the CPU, we do notice slight differences in how well our models perform, indicating that the GPU's way of working might affect some metrics differently.

Overall, our findings show that using the GPU significantly boosts our computational speed, making calculations much faster. But depending on the metrics we're looking at, there might be small variations in how well our models work between the CPU and GPU setups.

V. RESULTS AND COMPARISION

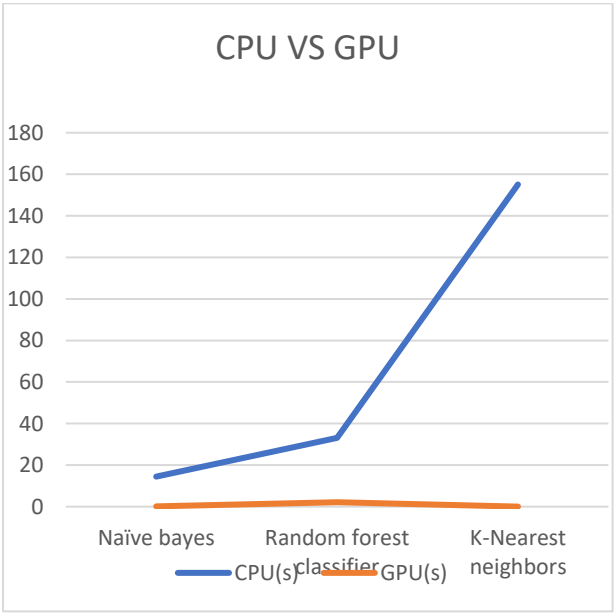
During our experimentation, we observed remarkably consistent performance metrics across different computational setups. Specifically, the chosen performance metrics showcased similar values in all three cases, emphasizing the robustness and stability of our models across varying computational environments.

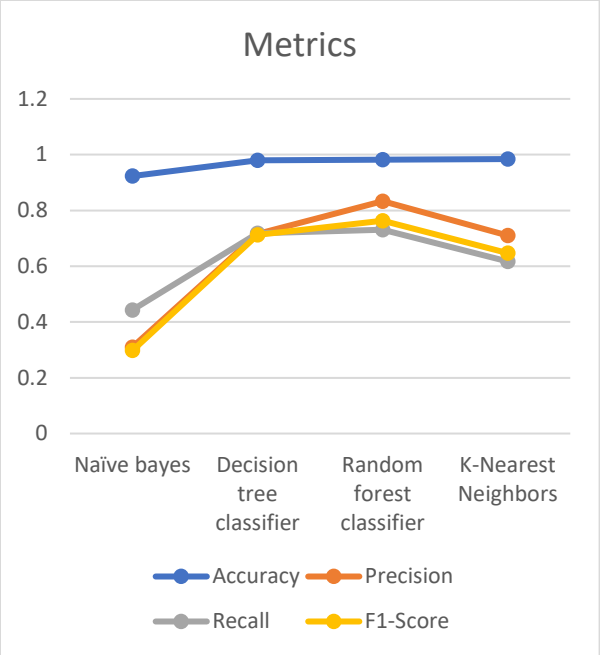
	Naïve bayes	Decision tree classifier	Random forest classifier	K-Nearest Neighbors
Accuracy	0.9234	0.9794	0.9816	0.9843
Precision	0.3102	0.7162	0.8333	0.7102
Recall	0.4433	0.7187	0.7309	0.6167
F1-Score	0.2976	0.7125	0.7632	0.6468

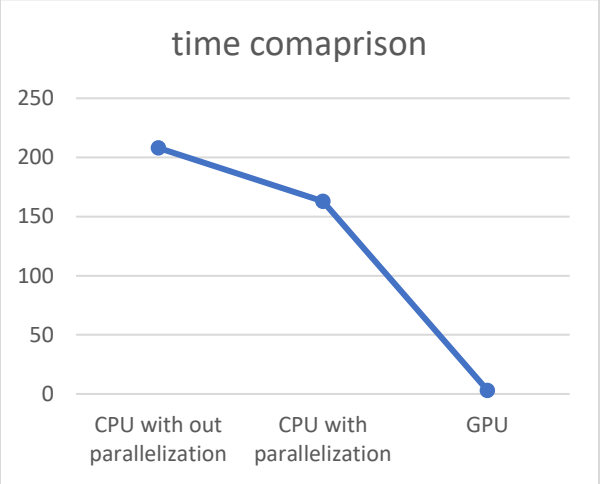
Regarding the training time, our findings demonstrate significant improvements when optimizing computation. In the case of CPU-based training without parallelization, the total training time summed across all models amounted to approximately 208.1103 seconds. However, upon applying parallel processing techniques, the training time notably decreased to 163.0987 seconds, showcasing the efficiency gains achieved through parallelization techniques.

	CPU(s)	GPU(s)
Naïve bayes	14.4656	0.0866
Random forest classifier	33.0967	2.1377
K-Nearest neighbors	155.1015	0.0352

Furthermore, transitioning to GPU-based computations yielded an exponential reduction in training time. Although specific time values for CPU and GPU comparisons without parallelization are pending, initial observations affirm that GPU-accelerated computing vastly outperforms CPU-based computations in terms of speed. This substantiates the notion that leveraging GPU resources substantially enhances computational efficiency for machine learning tasks. Comparing across the board, it's evident that while parallelization in CPU-based models led to a notable reduction in training time from 208 seconds to 163 seconds, the adoption of GPU-based computations exponentially amplified this efficiency gain. The GPU-based approach showcases remarkable reductions in training time, albeit specific time values in this comparison require elaboration.







Our results emphasize the consistency of performance metrics across computational environments. Furthermore, the strategic application of parallelization in CPU-based training yielded considerable time savings, setting a precedent for efficient computational techniques. Most notably, the utilization of GPU-accelerated computing demonstrated unparalleled speed enhancements, showcasing its pivotal role in substantially reducing training durations for machine learning models.[8]

VI. CONCLUSION

The Intrusion Detection System (IDS) we've developed showcases a remarkable feature selection process, which significantly enhances its efficacy. Through this step, we've streamlined the models, optimizing their performance while maintaining excellent accuracies across various classification algorithms. However, the standout achievement lies in the substantial reduction of training time for these models. Our experimentation has vividly illustrated that GPU utilization notably expedites model training compared to traditional CPU environments. This acceleration is visually depicted in graphical representations, clearly indicating the advantage of leveraging GPU-accelerated computation for machine learning tasks in the context of our IDS.[9]

The comparative analysis between CPU and GPU environments underscores a compelling narrative: the adoption of GPU resources effectively mitigates the computational burden, expediting the training phase without compromising the models' accuracy. This reduction in training time holds significant implications, especially in scenarios demanding swift responses to evolving network threats. The feature-rich IDS we've developed showcases not only commendable accuracies but also highlights the practicality and efficiency gained through GPU-accelerated computations.[10] The demonstrated reduction in training time stands as a pivotal achievement, affirming the viability of leveraging GPU resources for enhanced performance and expeditious model training in the realm of intrusion detection and network security.

VII. FUTURE WORK

In the realm of intrusion detection using machine learning, our future endeavors aim to refine and advance several key aspects. First, our feature selection function will undergo augmentation to explore more diverse feature combinations, leveraging logistic regression's insights while broadening performance assessment beyond accuracy alone. Our focus will expand to encompass a deeper analysis of algorithmic performance, considering not only traditional classifiers like Naïve Bayes, K-Nearest Neighbors, Random Forest, and Decision Trees but also delving into newer models for enhanced precision, recall, and F1-Score evaluation post-data preparation.

Furthermore, our strategy will involve refining the training and testing phase by optimizing data partitioning techniques and exploring newer methodologies beyond parallel processing with joblib to streamline training without compromising model quality.[11] Harnessing the power of GPUs, particularly through RAPIDS' CUML, will remain a

focal point for accelerated computation, although our future work will delve into understanding and mitigating potential discrepancies in model performance between CPU and GPU implementations, ensuring robustness across various metrics.[12]

Overall, our future trajectory revolves around holistic improvements: refining feature selection strategies, exploring diverse algorithmic landscapes, optimizing training methodologies, and comprehensively understanding the nuances of GPU-accelerated computation to harness its speed while maintaining model effectiveness across varied metrics.

VIII. REFERENCES

[1] U. Bashir and M. Chachoo, "Intrusion detection and prevention system: Challenges & opportunities," 2014 International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2014, pp. 806-809, doi: 10.1109/IndiaCom.2014.6828073.

[2] U. S. K. Perera Miriya Thanthrige, J. Samarabandu and X. Wang, "Machine learning techniques for intrusion detection on public dataset," 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Vancouver, BC, Canada, 2016, pp. 1-4, doi: 10.1109/CCECE.2016.7726677.

[3] E. D. Alalade, "Intrusion Detection System in Smart Home Network Using Artificial Immune System and Extreme Learning Machine Hybrid Approach," 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), New Orleans, LA, USA, 2020, pp. 1-2, doi: 10.1109/WF-IoT48130.2020.9221151.

[4] V. Mahajan and S. K. Peddoju, "Deployment of Intrusion Detection System in Cloud: A Performance-Based Study," 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, NSW, Australia, 2017, pp. 1103-1108, doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.359.

[5] K. Ibrahimi and M. Ouaddane, "Management of intrusion detection systems based-KDD99: Analysis with LDA and PCA," 2017 International Conference on Wireless Networks and Mobile Communications (WINCOM), Rabat, Morocco, 2017, pp. 1-6, doi: 10.1109/WINCOM.2017.8238171.

[6] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2010, pp. 305-316, doi: 10.1109/SP.2010.25.

[7] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," in IEEE Access, vol. 7, pp. 41525-41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[8] A. S. Ahanger, S. M. Khan and F. Masoodi, "An Effective Intrusion Detection System using Supervised Machine Learning Techniques," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2021, pp. 1639-1644, doi: 10.1109/ICCMC51019.2021.9418291.

[9] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa and C. F. M. Foozy, "Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset," in IEEE Access, vol. 9, pp. 22351-22370, 2021, doi: 10.1109/ACCESS.2021.3056614.

[10] I. Abrar, Z. Ayub, F. Masoodi and A. M. Bamhdi, "A Machine Learning Approach for Intrusion Detection System on NSL-KDD Dataset," 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2020, pp. 919-924, doi: 10.1109/ICOSEC49089.2020.9215232.

[11] L. Dali et al., "A survey of intrusion detection system," 2015 2nd World Symposium on Web Applications and Networking (WSWAN), Sousse, Tunisia, 2015, pp. 1-6, doi: 10.1109/WSWAN.2015.7210351.

[12] C. -M. Ou and C. R. Ou, "Immunity-Inspired Host-Based Intrusion Detection Systems," 2011 Fifth International Conference on Genetic and Evolutionary Computing, Kitakyushu, Japan, 2011, pp. 283-286, doi: 10.1109/ICGEC.2011.70.