Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 251 (2024) 66–74

Procedia Computer Science

www.elsevier.com/locate/procedia

The 15th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2024)
October 28-30, 2024, Leuven, Belgium

Resource Consumption Analysis of Distributed Machine Learning for the Security of Future Networks

Md Muzammal Hoque[a,*], Ijaz Ahmad[a], Mohammad Tahir[b]

[a] *VTT Technical Research Centre of Finland, Tekniikantie 21, Espoo, 02150, Finland*
[b] *Department of Computing, University of Turku, Turku, FI-20014, Finland*

Abstract

As the network continues to become more complex due to the increased number of devices and ubiquitous connectivity, the trend is shifting from a centralized implementation to decentralization. Similarly, strategies to secure networks are increasingly leaning towards decentralization for its potential to enhance security in future networks with the help of Machine Learning (ML) techniques. In this regard, Distributed Machine Learning (DML) techniques, such as Federated Learning (FL) and Split Learning (SL), are at the forefront of this shift, offering collaborative learning capabilities across network nodes while maintaining data privacy. However, ML requires vast amounts of dedicated computing, memory, bandwidth, and as a consequence, energy resources. Moreover, resource consumption ML techniques used for network security have mostly been overlooked, which presents a glaring challenge for future networks in terms of overall resource utilization. This research emphasizes the importance of understanding the resource consumption patterns of two important DML techniques, i.e., FL and SL, to analyze the consumption of critical resources when deployed for network security. Furthermore, this research draws important insights from a practical comparative analysis of FL and SL in terms of resource consumption patterns and discusses their scope for future network security, such as in 6G, and stirs further research in this area.

Keywords: Green AI; machine learning; Security; 6G; distributed security; 6G security;

1. Introduction

The rapid growth in the number of communicating nodes and diverse services has increased network complexity. The increasing complexity and data volume in communications networks have increased the network security threat landscape. Machine Learning (ML) has become crucial across network layers, from access to backhaul and core

* Corresponding author. Tel.: +3-585-059-95038.
E-mail address: muzammal.hoque@vtt.fi

1877-0509
10.1016/j.procs.2024.11.085

networks, ensuring smooth operations of different tasks and services [1]. ML offers promising solutions for network security, including prevention, detection, response, and remediation. Traditionally, ML techniques are deployed in a centralized manner where all the data is aggregated, and the model is trained on the data. However, centralized deployment of ML techniques presents several challenges, which include increased latency, single point of failure, reduced availability, and security threats, such as Denial of Services (DoS) and Distributed DoS (DDoS) attacks. More recently, data privacy issues have also become a significant problem and this has raised serious concerns about the usage of data for training ML models which might contain private information of the users.

Distributed Machine Learning (DML) can address most of the constraints of traditional ML by enabling parallel learning across multiple nodes, thereby providing efficiency in terms of latency, resource consumption, reliability and performance [2]. Furthermore, the raw data is not required to be shared to build a global model. Only the local model built on the distributed nodes is shared, thereby providing data privacy for the users. DML is poised to play a pivotal role in securing future networks. It can manage large-scale learning on vast datasets within distributed environments. DML allows training on local devices, minimizes data transfer across the network, enhances privacy, and eliminates the time required to prepare and deliver data to a central device. However, some devices, such as IoT and devices with limited computational resources, may struggle to process complex ML algorithms [3]. Future networks, such as 6G are expected to be Internet of Everything (IoE) compliant and a sizeable decentralized system with the capability of making decisions at different levels [5]. Therefore, the resource demands of DML will be a key concern while deploying in future networks, and extensive research is needed to manage the resource consumption of DML. By measuring and analyzing the resource consumption of DML, it will be possible to understand the behavior of future networks, ensure proper resource allocation and utilization, and determine DML's scope of applications. This will aid in developing robust, scalable, and efficient future networks.

Federated Learning (FL) [6] and Split Learning (SL) [12] are two promising DML techniques. A comparative analysis of the resource consumption of these two techniques in network security will help understand their resource implications. Although some comparative analyses have been presented in [8] and [9], they do not pertain to network security. In this study, we explore these two DML techniques, with a focus on fingerprinting the resource consumption, in terms of memory, CPU utilization, power consumption, and potential CO_2 emissions, while also considering the accuracy of the models.

The remainder of the paper is structured as follows: Section II provides a brief overview of DML, FL, and SL. Section III outlines the experimental setup, use case, dataset used, and evaluation of the chosen techniques, while Section IV presents the results of our experiment. Section V discusses the implications of DML for future networks, especially for 6G security. Finally, Section VI presents the conclusion.

2. Distributed Machine Learning

DML allows training ML models without requiring data to be transmitted to a centralized server. DML is suitable for training large and complex models, which are difficult to train on a single machine. This is relevant to the requirements of future networks, such as 6G, since significant growth in data will make it challenging to centralize and process all the data on a single server [10]. Additionally, with a growing need for data privacy, sensitive data must be confined to devices or within certain geographical regions. In such scenarios, DML enables training models without centralising all the data, addressing privacy and data sovereignty concerns. From a communication point of view, DML results in lower overheads as the model parameters need to be shared without requiring the transmission of large amounts of data. For certain applications requiring low latency, such as autonomous vehicles, DML can enable faster response times by training and processing the data closer to the sources. It also aids in improving the scalability, accuracy, and efficiency of ML models. Therefore, various distributed paradigms are currently being researched, including data parallelism, model parallelism, hybrid parallelism, ensemble learning, etc., [11]. In this section, we discuss two popular DML techniques, FL and SL.

2.1. Federated Learning

FL trains ML models on data that is distributed on multiple devices without centralizing or sharing the raw data. Initially, each client receives a copy of the pre-trained foundation model. This model is then trained with the private

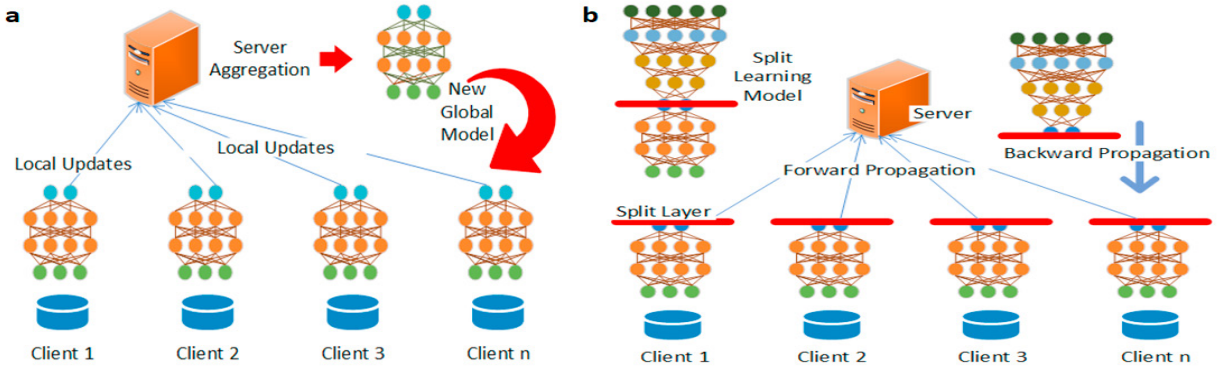

Fig. 1. (a) A high level presentation of federated learning mechanism; (b) A high-level presentation of split learning mechanism.

data of the clients. These model updates from the client devices are sent back to a central server, where the server aggregates the updated model parameters and creates a new global model. Each client then receives these global models, and the process continues iteration after iteration until the model is fully trained. Since there is no need to share the raw data with the server, the privacy of the client's data is protected. Additionally, FL enables parallel training for many clients together, which is considered an advantage. Fig. 1 (a) shows a generic overview of the Federated Learning process. However, since the clients need to train the whole model at their end, sometimes it becomes a challenge for the resource-constrained client devices [12].

2.2. *Split-Learning*

In SL, a deep neural network is divided into multiple parts, and each of these parts is trained in a different client. However, unlike FL, where the whole model is trained on the client device, a part of the neural network is trained privately at the user end, and the remaining part is trained on the server. Hence, the clients are relieved from the burden of the resource-intensive part of the computation, which is crucial for resource-constrained devices. In SL, the model architecture is divided into layers, and each client maintains the weights of what is learned until a certain layer, known as a split layer. The rest of the layers are held on the server. The SL reduces the size of the communication payload that needs to be sent during the distributed training. This is because the activation from only a single layer (split layer) is required to be sent to the server from each client. Additionally, the gradient from only one layer (the layer after the split layer) needs to be sent from the server to a client [13]. Fig. 1 (b) shows a generic training process for SL. However, unlike FL, training clients is done sequentially in SL. Only a single client trains with the server at one instance, which may lead to a significant training overhead for many clients. Nevertheless, some SL variants allow for parallel training, as documented in the literature.

3. Experimental Setup and Model Training

This section presents the experimental setup for measuring the performance of FL and SL using a dataset consisting of DDoS attacks to simulate a realistic attack scenario in the real world. The DDoS attack dataset was chosen as it is the most common form of attack on the network and targets the resources of the network resulting in increased consumption. These attacks are becoming more frequent, leading to significant financial and reputational damage for organizations. Generally, DDoS attacks flood a target server or network with excessive traffic, causing slowdowns or making it inaccessible to genuine users. Although many traditional and statistical model-based DDoS detection approaches exist, the prevalence of DDoS attacks continues to grow. In this section, we discuss the attack scenario, experimental setup, dataset preparation, model training and testing, and related steps.

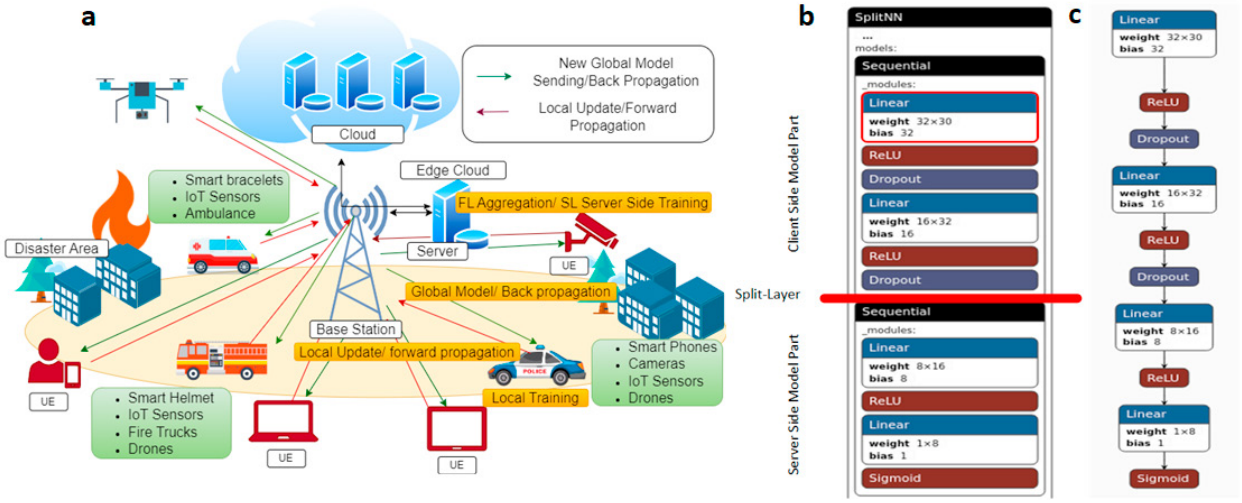

Fig. 2. (a) A generalized network with key relevant network and user segments in DDoS detection; (b) SL model structure; (c) FL model structure.

3.1. Attack Scenario: DDoS

Future networks will be the backbone of communications serving almost every sector, such as V2X, IoT, healthcare, public and emergency services. Therefore, it is important to protect crucial ubiquitous networks from various attacks. In particular, a DDoS attack, which is relatively easy to execute, and can result in wide-scale disruption of various critical services directly or indirectly. Furthermore, ML models themselves are susceptible to DoS attacks resulting from input flooding, resource depletion, model overload and data manipulation. These attacks can result in increased resource consumption, exhaust system resources, and cause unresponsiveness, posing significant challenges to the reliability and security of the networks. DML-based DDoS detection solutions like FL and SL can be used to detect and mitigate these malicious requests attempting to cause DDoS. In such cases, the DML model can use data and computation power from various devices participating in the process while the user's privacy is preserved since the raw data is not shared with the server. The devices where DML can be deployed may include drones, IoT devices, smartphones, smart helmets, cameras, autonomous vehicles and others. The resource consumption for devices for training should be kept as minimum as possible so that these devices do not face any difficulties performing their original purpose. Fig. 2 (a) shows a conceptual scenario depicting the use of a wide variety of devices for training a global model using DML for future networks. Furthermore, resource consumption must be minimised to reduce the carbon footprint and build a sustainable green network.

3.2. Experimental Setup

The comparison was conducted in a high-performance system with an x86-64 architecture and a 12th Gen Intel® Core™ i9-12900K CPU. It had 24 CPUs and 16 cores per socket. The CPU speed ranged from 800 to 6700 MHz, supporting VT-x virtualization. Additionally, it had an L1d cache of 384 KiB, an L1i cache of 256 KiB, and a L2 cache of 10 MiB. It also had 64GB RAM. The experiments were conducted using Anaconda PyTorch and Jupyter Notebook. Which are widely accepted in the ML community for their robustness, flexibility, and interactive capabilities in development, execution, and experiment sharing.

3.3. Dataset

This experiment made use of the CIC-DDoS2019 dataset, a standard dataset created by the Canadian Institute for Cybersecurity [14]. The dataset comprises over 50 million entries, with the majority (50,006,249 entries) representing DDoS attacks and a small fraction (56,863 entries) indicating normal behaviour. The dataset is divided into 11 pack-

ages and includes 12 different types of DDoS attacks, such as NetBIOS, MSSQL, DNS, LDAP, NTP, SNMP, SSDP, UDP, Syn, TFTP, UDP Lag, and WebDDoS. For the experiment, 50,000 instances were randomly selected from each of the 11 packages, totalling 550,000 instances. The selected dataset includes both attack and normal instances, with the majority (544,670 instances) being various types of attacks and a minority (5330 instances) being normal data.

3.4. Data Pre-Processing and Feature Extraction

Before any machine learning model can be trained, data preparation is an essential step. The data from the real world may contain errors, duplicates, outliers, missing numbers and inconsistency. To enhance the training process and reduce the data size, certain features deemed less informative (source and destination IPs, timestamps etc.) were excluded from the training process. Initially, the dataset had 88 features. In order to standardize the features of the dataset, *StandardScaler* was employed to remove the mean and scale to unit variance. Furthermore, the Principal Component Analysis (PCA) technique was employed to extract the most relevant features, resulting in 30 features. After feature extraction, the target variables were binary encoded, where all the attack data was assigned 0, while the normal data was assigned 1. The splitting ratio was 75:25 for the training and testing datasets. The training dataset was balanced using SMOTE (Synthetic Minority Oversampling Technique), which generates synthetic data using a linear interpretation between the minority class samples and their K-nearest neighbours. After balancing, the training set had a total of 816,928 instances with nearly equal numbers of DDoS and normal data. The test dataset was imbalanced, with 136,206 instances of attack data and 1,294 instances of normal data out of a total of 137,500 instances.

3.5. Model Training Setup

For the experiment, the FL and SL source codes were adapted from [15]. A total 32 numbers of client nodes were used, with 10, 16, and 24 randomly selected for each training round. Each client was trained with a separate dataset. A similar neural network structure was implemented for both SL and FL models, where the model was split into client and server parts for SL, as shown in Fig. 2 (b) and (c). Both models are feed-forward neural networks with dropout regularization and ReLU activation. The Adam optimization algorithm was used with a learning rate of 0.001. The models were trained for 5 epochs over 50 rounds, and performance was evaluated after each round. The process was repeated for all client batches to measure time, CPU usage, memory usage, power consumption, and CO_2 emissions.

4. Results and Discussion

The experiments were conducted with 32 clients, varying the training batch size of clients to 10, 16, and 24. The bath size is arbitrary and can take any number less than the total number of clients. The aim was to study the resource consumption of FL and SL algorithms in DDoS detection, their change in performance in highly imbalanced test datasets with various batch sizes of clients, and the impact of the dataset on various resource consumption. The findings are discussed below.

4.1. CPU Usage

The research used the ‘psutil‘ [16] Python library to monitor CPU and memory usage in real-time, employing a background thread for continuous data collection. It recorded the CPU and memory usage per second for both FL and SL throughout the entire training and testing process. As each setup required a different amount of time to complete, the resulting graphs varied in length. The ‘cpu_percent‘ method was used with no interval, ensuring immediate CPU usage readings. Results showed that the average CPU usage for FL with 10 clients was 8.10%, while for SL, it was slightly higher at 8.52%. FL’s peak CPU usage reached 17.4%, suggesting some operations are highly resource-intensive.

In contrast, SL’s peak was only 13.3%. Moreover, the standard deviation of FL and SL were 2.78 and 2.91, respectively, indicating that SL has more variability in CPU usage compared to FL. For 16 clients batch, The average CPU usage for FL is 7.78%, while for SL, it is significantly lower at 5.22%. This indicates that, on average, SL utilises less CPU resources than FL. However, the maximum CPU usage for FL and SL was 29.6% and 33.3%. Additionally, for

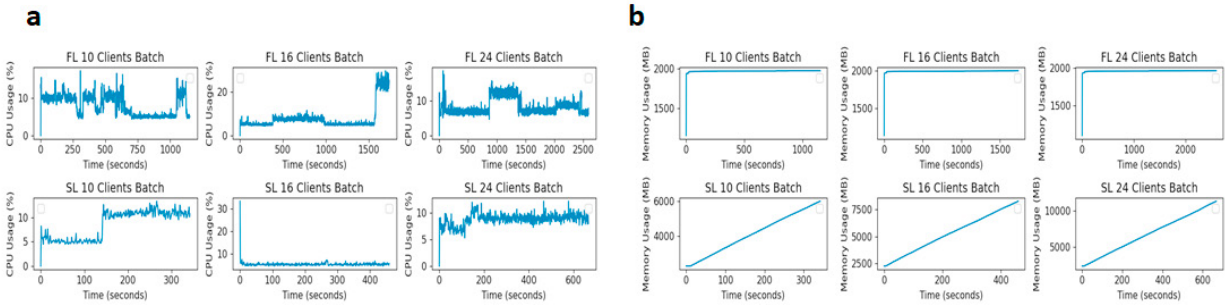

Fig. 3. (a) CPU usage over time; (b) Memory usage over time.

SL the highest spike to this maximum value was only for the first second, and then it stabilized and never went that up. After that, the maximum value for SL was 7.4%. The minimum CPU usage for FL was 0.0%, and for SL, it was 4.4%, indicating that SL has a higher baseline CPU usage than FL.

Furthermore, the standard deviation for FL was 5.36, and for SL was significantly lower at 1.39, indicating that FL's CPU usage had more variation than SL. Finally, for 24 clients, the average CPU usage for FL was 8.38%, while for SL, it was slightly higher at 8.81%. This indicates that, on average, SL utilises more CPU resources than FL. Whereas the maximum CPU usage for FL was 18.7%, and for SL it was 12.2%, and the standard deviation for FL was 2.28, and for SL, it was 1.10. Hence, while SL has high average CPU usage, FL had a higher peak (resource-intensive operations) and more variability. Fig. 3 (a) shows the CPU usage of the FL and SL techniques.

Table 1. The Performance and Resource Consumption Comparison of FL and SL

Models	Performance Metrics				Resource Metrics				
	Avg. Accu	Avg. Pre	Avg. Rec	Avg. F1 Sco	Avg. CPU (%)	Peak Memory (MB)	Power (Wh)	Time (Min)	CO2 (g)
Fed 10	0.9962	0.7109	0.9980	0.8302	8.10	1974.59	5.00	19.29	0.71
Split 10	0.9961	0.7098	0.9991	0.8298	8.52	5997.20	1.59	5.73	0.23
Fed 16	0.9965	0.7312	0.9990	0.8437	7.78	2000.15	7.56	29.05	1.07
Split 16	0.9962	0.7158	0.9992	0.8339	5.22	8286.378	2.19	7.67	0.31
Fed 24	0.9965	0.7314	0.9990	0.8444	8.38	1960.54	11.19	43.31	1.59
Split 24	0.9965	0.7276	0.9989	0.8419	8.81	11335.62	3.22	11.14	0.46

4.2. Memory Usage

The experiments used psutil's memory_info().rss to track the Resident Set Size (RSS) memory usage in MB every second. We found that the memory usage of SL increased linearly over time, while FL increased initially and then remained nearly constant. For example, for FL with 10, 16, and 24 client batches, the memory usage peaked at around 1974.59 MB, 2000.15 MB, and 1960.54MB, respectively, and stayed nearly constant thereafter. In contrast, SL's memory usage kept increasing, reaching 5997.21 MB, 8286.375 MB, and 11335.62MB at the end of 50 rounds for 10, 16, and 24 clients, respectively. This is 3.04, 4.14, and 5.78 times more than FL's memory usage. The memory usage comparison of FL and SL is shown in Fig. 3 (b). However, it is important to note that in the case of SL, the bulk of this memory usage is generally allocated to the server side. Thus, the clients are minimally affected by this increase in memory usage.

4.3. Power Consumption, CO2 Emission, and Turnaround Time for Execution

The experiments used the Eco2AI python library [17], which can accumulate statistics about power consumption and CO_2 emission while running a code. We found that FL consumed significantly more power and emitted more CO_2 than SL in all scenarios. For example, with 10 clients, SL consumed about 3.14 times less power and emitted

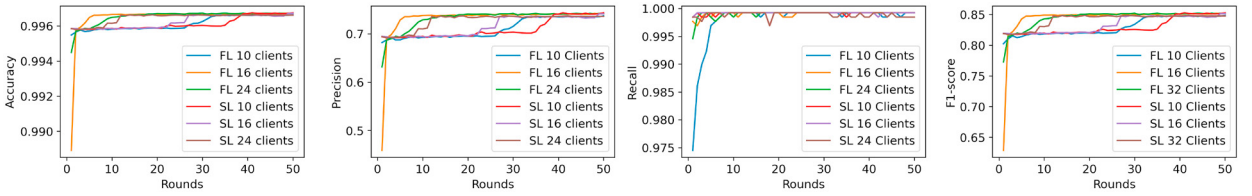

Fig. 4. Performance comparison of FL and SL in terms of accuracy, precision, recall, and F1-score.

3.16 times less CO_2 than FL. For 16 clients, these figures were 3.45 and 3.43 times less, respectively, and for 24 clients, they were 3.48 and 3.46 times less. The turnaround time was measured using the Python Time library [18], and found that FL required approximately 3.37, 3.79, and 3.89 times more time than SL for 10, 16, and 24 client batches, respectively.

4.4. Performance Evaluation

Along with the resource consumption, we also compared the performance of FL and SL for DDoS detection. Both methods showed high accuracy, exceeding (>0.996) shown in Table 1. FL slightly outperformed SL in terms of false positives and precision, with precision increasing alongside the client batch size and training rounds, as shown in Fig. 4. Both FL and SL had recall rates near 1, demonstrating their effectiveness in identifying true positive instances. Overall, FL had a marginally higher F1 score, indicating a better balance between precision and recall throughout the tests.

4.5. Recommendations from Analysis

SL and FL are two distributed machine learning approaches that train models on distributed data while maintaining data privacy. SL has been found to be more resource-efficient than FL, as it requires only intermediate representations to be exchanged between the client and server, not the raw data. This results in lower computational and bandwidth demands, making SL a better fit for large-scale networks and time-sensitive applications. On the other hand, FL may be preferable when data is unevenly distributed or when model complexity and communication costs are not critical issues. Ultimately, the decision on which distributed ML approach to use for the security of future networks will hinge on various factors, including computational resources, communication constraints, type of attack, data distribution, and privacy needs. A hybrid model combining SL and FL could offer a more resource-efficient and secure solution in certain cases [19], such as collaborative Intrusion Detection Systems (IDS) and threat intelligence.

4.6. Limitations

The research has certain limitations, such as the use of a single machine simulating the distributed environment for the experiments, which may not reflect the actual resource consumption in a physically distributed setting. The combined measurement of resources for clients and servers limits the understanding of individual client-server consumption in FL and SL. Additionally, CO_2 emissions vary with the energy source, and the study used a global average emission coefficient for calculations. The tools and library resource usage were also included in the measurements. Despite these constraints, the study offers valuable insights for future research on resource-efficient and sustainable future networks, contributing to efforts against climate change.

5. Implications for 6G Networks and Future Scope

6G networks will integrate an extensive array of devices, systems, and services, which will emerge with unique requirements. 6G is expected to improve on almost all key performance indicators (KPIs) of 5G standards. Some of these KPIs will require a radical shift from the way existing cellular network architectures are implemented. For

example, the existing 5G network is largely centralized, where most control network functions reside in the core network. Even though some of the core network functions can be pushed to the edge of the network, most functions still remain centralized. In order to meet the latency requirements of future services, most core network functions, such as the authentication function, will be pushed to the far edge of the network. The resulting network will be an extremely complex environment compared to its predecessors. Therefore, machine learning, with its ability to analyze and interpret vast amounts of data, will play a pivotal role in the management and operations of 6G networks, ensuring the necessary efficiency, reliability, and security of such networks [20]. Additionally, the integration of native AI into the next generation of mobile networks continues to evolve rapidly. Realizing the potential benefits of native AI requires significant investments in computing, memory, and energy resources within the 6G network. Training large AI models demands a significant amount of resources, and it is imperative to carefully consider the allocation and management of resources when designing and implementing native AI systems in the 6G network infrastructure. For example, training a large language model can consume hundreds of megawatts hours of electricity, several terabytes of memory, and hundreds of powerful GPUs. Even though the consumption of resources using ML or AI can be insignificant considering smaller tasks in 6G networks, the net impact can be much higher since 6G networks are considered more of an ecosystem than a simple packet-forwarding infrastructure. For instance, 6G will provide differentiated Quality of Service (QoS) for different verticals and services, such as digital healthcare and vehicular networks. An investigation is essential for resource utilization since, by considering the resource utilization of native AI in the 6G network design, we can pave the way for a robust and reliable native AI system that will drive innovation and be sustainable in years to come. From a network security perspective, ML is essential for developing various automated security solutions for threat detection, prevention, response, and remediation in 6G, where distributed ML such as FL and SL will play a significant role in the sustainable development of security in 6G networks.

6. Conclusions

DML techniques like FL and SL will play a major role in security decentralization for future networks. In this regard, the work presented in the paper examined their performance and resource consumption usage in DDoS detection. The finding shows that both models perform well, while FL appears to have a slight edge over SL in terms of precision and F1 score. However, the differences are quite small. In terms of resource consumption, it was observed that both of them have their advantages and shortcomings. In terms of CPU usage, SL had higher CPU usage, while FL had higher peaks and variability. In terms of memory usage, FL was more memory efficient than the SL. Finally, SL was more time and power-efficient and had lower CO_2 emission. Ultimately, the choice between the type of DML for security will depend on factors such as computational resources, communication constraints, type of attacks, data distribution, and privacy requirements. As a future work, we aim to test more extensively in a distributed environment covering other attacks to analyse and minimize the resource consumption of DML algorithms.

Acknowledgements

This work was supported by the Business Finland funded SUNSET-6G project.

References

[1] Peltonen, Ella, Ahmad, Ijaz, Aral, Atakan, Capobianco, Michele, Ding, Aaron Yi, Gil-Castiñeira, Felipe, Gilman, Ekaterina, Harjula, Erkki, Jurmu, Marko, Karvonen, Teemu, Kelanti, Markus, Leppänen, Teemu, Lovén, Lauri, Mikkonen, Tommi, Mohan, Nitinder, Nurmi, Petteri, Pirttikangas, Susanna, Sroka, Paweł, Tarkoma, Sasu, and Yang, Tingting. (2022) “The Many Faces of Edge Intelligence.” In *IEEE Access*, 10, 104769-104782.

[2] Naik, Dishita and Naik, Nitin. (2024) ”The Changing Landscape of Machine Learning: A Comparative Analysis of Centralized Machine Learning, Distributed Machine Learning and Federated Machine Learning.” In *Advances in Computational Intelligence Systems*, edited by Nitin Naik, Paul Jenkins, Paul Grace, Longzhi Yang, and Shaligram Prajapat, 18-28. Cham: Springer Nature Switzerland.

[3] Aygül, Mehmet Ali, Türkmen, Halise, Sağlam, Mehmet İzzet, Çirpan, Hakan Ali, and Arslan, Hüseyin. (2023) ”Centralized and Decentralized ML-Enabled Integrated Terrestrial and Non-Terrestrial Networks.” In *2023 IEEE Future Networks World Forum (FNWF)*, 1-6.

[4] Ahmad, Ijaz and Shahabuddin, Shahriar and Kumar, Tanesh and Okwuibe, Jude and Gurtov, Andrei and Ylianttila, Mika. (2019) ”Security for 5G and Beyond.” *IEEE Communications Surveys & Tutorials* **21** (4): 3682-3722.

[5] Wang, Minghao, Zhu, Tianqing, Zhang, Tao, Zhang, Jun, Yu, Shui, and Zhou, Wanlei. (2020) "Security and privacy in 6G networks: New areas and new challenges." *Digital Communications and Networks* **6** (3): 281-291.

[6] McMahan, Brendan and Moore, Eider and Ramage, Daniel and Hampson, Seth and Arcas, Blaise Aguera y. (2017) "Communication-Efficient Learning of Deep Networks from Decentralized Data." In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, edited by Singh, Aarti and Zhu, Jerry, 1273–1282. **54**. PMLR.

[7] Otoum, Safa and Guizani, Nadra and Mouftah, Hussein. (2023) "On the Feasibility of Split Learning, Transfer Learning and Federated Learning for Preserving Security in ITS Systems." *IEEE Transactions on Intelligent Transportation Systems* **24** (7): 7462-7470.

[8] Turina, Valeria and Zhang, Zongshun and Esposito, Flavio and Matta, Ibrahim. (2021) "Federated or Split? A Performance and Privacy Analysis of Hybrid Split and Federated Learning Architectures." *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)* : 250-260.

[9] Gao, Yansong and Kim, Minki and Abuadbba, Sharif and Kim, Yeonjae and Thapa, Chandra and Kim, Kyuyeon and Camtep, Seyit A. and Kim, Hyoungshick and Nepal, Surya. (2020) "End-to-End Evaluation of Federated Learning and Split Learning for Internet of Things." *2020 International Symposium on Reliable Distributed Systems (SRDS)* : 91-100.

[10] Okwuibe, Jude and Haavisto, Juuso and Kovacevic, Ivana and Harjula, Erkki and Ahmad, Ijaz and Islam, Johirul and Ylianttila, Mika. (2021) "SDN-Enabled Resource Orchestration for Industrial IoT in Collaborative Edge-Cloud Networks." *IEEE Access* **9**: 115839-115854. doi:10.1109/ACCESS.2021.3105944.

[11] Afzal, Muhammad Usman and Abdellatif, Alaa Awad and Zubair, Muhammad and Mehmood, Muhammad Qasim and Massoud, Yehia. (2023) "Privacy and Security in Distributed Learning: A Review of Challenges, Solutions, and Open Research Issues." *IEEE Access* **11**: 114562-114581.

[12] Otoum, Safa and Guizani, Nadra and Mouftah, Hussein. (2023) "On the Feasibility of Split Learning, Transfer Learning and Federated Learning for Preserving Security in ITS Systems." *IEEE Transactions on Intelligent Transportation Systems* **24** (7): 7462–7470.

[13] Vepakomma, Praneeth and Raskar, Ramesh. (2022) "Split Learning: A Resource Efficient Model and Data Parallel Approach for Distributed Deep Learning." In Ludwig, Heiko and Baracaldo, Nathalie (Eds.), *Federated Learning: A Comprehensive Overview of Methods and Applications*, Springer International Publishing, Cham, pp. 439–451.

[14] Sharafaldin, Iman and Lashkari, Arash Habibi and Hakak, Saqib and Ghorbani, Ali A. (2019) "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy." In *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8.

[15] Desai, P. R. (2021) "Credit Card Fraud Detection using Federated Learning and Split Learning." *GitHub Repository*, GitHub. Available at: https://github.com/PR-Desai2226/Credit-card-fraud-detection-using-Federated-Learning-and-Split-Learning

[16] Rodola', Giampaolo. (2024) "psutil." *Python Package Index* **5.9.8**: Available at: https://pypi.org/project/psutil/.

[17] Budennyy, S. A. and Lazarev, V. D. and Zakharenko, N. N. and Korovin, A. N. and Plosskaya, O. A. and Dimitrov, D. V. and Akhripkin, V. S. and Pavlov, I. V. and Oseledets, I. V. and Barsola, I. S. and Egorov, I. V. and Kosterina, A. A. and Zhukov, L. E. (2022) "eco2AI: Carbon Emissions Tracking of Machine Learning Models as the First Step Towards Sustainable AI." *Doklady Mathematics* **106** (1): S118–S128.

[18] Python documentation, "Time — Time access and conversions," Available online: https://docs.python.org/3/library/time.html, Accessed on: Jan. 19, 2024.

[19] Liu, Xiaolan and Deng, Yansha and Mahmoodi, Toktam. (2022) "Wireless distributed learning: A new hybrid split and federated learning approach." *IEEE Transactions on Wireless Communications* **22** (4): 2650–2665.

[20] Ahmad, Ijaz and Shahabuddin, Shariar and Malik, Hassan and Harjula, Erkki and Leppänen, Teemu and Lovén, Lauri and Anttonen, Antti and Sodhro, Ali Hassan and Mahtab Alam, Muhammad and Juntti, Markku and Ylä-Jääski, Antti and Sauter, Thilo and Gurtov, Andrei and Ylianttila, Mika and Riekki, Jukka. (2020) "Machine Learning Meets Communication Networks: Current Trends and Future Challenges." *IEEE Access* **8**: 223418-223460.