Patryk Paszko
Marek Konieczny
Sławomir Zieliński
Bartosz Kwolek

FL-MEC: FEDERATED LEARNING FOR NETWORK TRAFFIC CLASSIFICATION ON THE NETWORK EDGE

Abstract *Nowadays, two technological trends, Federated Learning (FL) and Edge Computing (EC), are increasingly important and influential. FL is a decentralized machine learning strategy that allows learning on distributed data. It primarily allows performing learning operations close to the user, where the data is gathered. This approach belongs to the EC domain, where the main goal is to move computation closer to the end user (e.g., from the centralized cloud). In our work, we apply the FL and EC in the context of network flow classification. We achieved an accuracy of 0.957 with the FL model, compared to 0.924 for the best local model. We achieved these results thanks to the federated averaging performed on neural network layers. To verify our approach, we executed all our experiments on a virtualized environment that emulates existing mid-scale EC network infrastructure, including limitations related to resource constraints on edge nodes.*

Keywords network traffic classification, edge computing, federated learning

Citation Computer Science 26(4) 2025: 181–201

1. Introduction

We increasingly rely on a variety of connected devices, from standard desktops and mobile devices to IoT devices. We also rely heavily on Internet services. It is especially true in the post-pandemic period, when many activities have shifted to remote domains, including, but not limited to, remote work and teaching. The increase in usage puts significant pressure on infrastructure as demand for high-quality networks grows.

From a networking perspective, one of the crucial factors is proper network traffic classification, as it enables us to understand the types of traffic that flow through a network and how the network is utilized. Next, we can utilize these insights to optimize network performance, enhance security, and ensure that critical applications and services receive priority and appropriate quality of service. There are several essential use cases where network traffic classification is crucial.

First, network traffic classification can help us identify and block malicious traffic. By understanding the types of traffic flowing through a network, we can identify patterns and behaviors indicative of attacks or malware. Based on that, we can develop appropriate security measures and block malicious traffic before it causes harm. Network data analysis is crucial for intrusion detection systems [35].

Second, different types of traffic have their specific performance requirements. For example, interactive video and voice traffic requires low latency, low jitter, and high throughput, while file transfers and email traffic are less delay-sensitive. By classifying traffic, we can prioritize different types of traffic based on their performance needs, ensuring that critical applications and services receive the resources they need to function correctly.

Third, network traffic classification can help us optimize network performance. By understanding which types of traffic consume the majority of available bandwidth, we can identify potential bottlenecks, reduce congestion, and improve performance. The analysis of network usage (services, flow types, upload / download) facilitates both the introduction of optimizations and the establishment of baselines, which serve as a basis for detecting anomalies in network traffic [2].

Fourth, network traffic classification can inform network capacity and resource allocation decisions. By understanding the types of traffic flowing through a network, we can better predict future demand and plan for capacity upgrades and resource allocation accordingly.

Typically, a dedicated central monitoring infrastructure performs network traffic analysis. All traffic information collected by network devices is sent to central units for analysis. The scenario often requires specialized software and equipment, leading to a vendor lock-in. Access to all the data in place is a great benefit of centralization, as it allows us to use machine learning on big data sets. However, this raises questions related to data privacy and concerns about the efficiency of the utilization of network resources [29]. To address those issues, we propose FL on the network edge to solve problems related to: (1) data distribution on the network edge devices, (2) resource limitations of the network edge devices, and (3) protection of data and privacy when analyzing network traffic.

In the following sections, we introduce the design and implementation of our real-time data acquisition system in the existing network. We also describe a scalable federated environment, which we created, that could be used for any federated learning task. Finally, we characterize a federated learning neural network solution that we built and measured its performance, runtimes, and data usage. Our results are promising - the federated model performs better than several local models trained on varying amounts of data.

2. Related work

FL is a decentralized machine learning technique that allows training on a massive corpus of data distributed across many devices [7, 25]. It exemplifies the broader strategy of "taking the code to the data" instead of "taking the data to the code", which tackles data ownership, privacy, and locality issues at their core.

Classic FL architectures are mainly based on a central server to aggregate model updates. In contrast, decentralized or peer-to-peer FL approaches eliminate this dependency by exchanging updates among nodes directly. In that way, the nodes participating in the FL learning process demonstrate that global models can be trained across multiple clients without data centralization. In addition, the nodes often operate on resource-constrained devices, such as IoT devices or mobile phones, with limited computational and communication capacity [1, 3, 16].

Decentralized Federated Averaging (DFedAvg) [39] and earlier decentralized SGD formulations [21] show that under realistic network topologies, decentralized learning can achieve convergence rates comparable to server-based FL. Recent surveys [4,11,28] and communication-efficient FL frameworks [8, 27, 33] further explore different techniques of compression, sparsification, and quantization to improve communication and reduce bandwidth overhead. These findings are particularly relevant to our use case, where resource-constrained Raspberry Pi nodes and edge network device nodes exchange neural network weights over local networks, leading to significant communication costs. However, they do not consider constrained memory and computational capabilities of FL nodes.

Network Traffic Classification remains an essential element in network management and traffic analysis. Recent surveys [5,34] emphasize the growing importance of encrypted traffic analysis, where statistical and temporal flow features replace payload inspection. Deep neural networks have shown strong performance in these tasks, yet their deployment is often centralized and requires significant data volumes. In contrast, federated and edge-based learning approaches enable privacy-preserving analysis of local traffic traces.

Some recent network traffic studies have focused on the classification problem on the network edge [32, 36, 37]. Recent work also reviews several supervised learning algorithms: naÛve Bayes classifier, C5.0 decision tree, neural network based on radial base function, Bayesian network [26]. Surprisingly, despite its complex implementation, the Bayesian network allowed for satisfying results.

One of the most interesting trends we have observed recently is the increased attention paid to integrating FL with the network traffic classification domain. Comprehensive reviews [10, 13, 18] demonstrate that FL can effectively aggregate security insights from distributed nodes while preserving privacy and without revealing sensitive user data. However, most existing approaches still depend on a centralized node that serves as a coordinator or utilize cloud infrastructure. In contrast, FL-MEC adopts a fully peer-to-peer architecture, performing federated averaging directly among nodes and evaluating performance under realistic network constraints.

To conclude, prior work has separately addressed federated and decentralized learning, communication efficiency, and traffic classification. To the best of our knowledge, no prior study has evaluated a fully peer-to-peer FL setup performing network traffic classification directly on the network edge, using real-world medium-scale educational infrastructure and resource-constrained hardware. FL-MEC fills this gap by combining decentralized model averaging, real network data, and runtime communication analysis across approximately one hundred nodes.

3. Experimental environment

In this section, we present our system designed for experiments with federated learning, and the real-life environment from which we gathered our dataset. Because of the name of the original environment (Małopolska Educational Cloud, MEC), we call the experimental system FL-MEC.

To understand the architecture of the presented solution, let's start with the protocol we use in the learning process. All operations are executed close to the network users, on edge devices (see 1 in Figure 1). In our setup, we preconfigured network devices (Ethernet switches) to mirror all users' traffic to specific ports. We attached our edge processing units (Raspberry Pis) directly to the mirroring ports. The devices analyze all the traffic and participate in the learning protocol. Such approach allows us to program and reconfigure all needed system components flexibly, because although we cannot program network devices directly, we still have full control over the data processing. Moreover, the processing scheme is repeatable – it is very easy to reuse it with network devices from different vendors, i.e., heterogeneous networking environments. It also allows us to update the software and data used in experiments dynamically without disturbing the network users, and gives us access to all necessary libraries.

Once the processing unit finishes booting up, it starts participating in the learning process. First, it trains the local model, and later redistributes the model parameters (see Section 4) to all other nodes (see 3 in Figure 1), and also accepts updates from them. Once a device receives the updates from all nodes, it computes the average of weights following the federated averaging algorithm (see Section 4.1). That concludes a single iteration of the algorithm. Details regarding data processing are presented in Section 3.2.

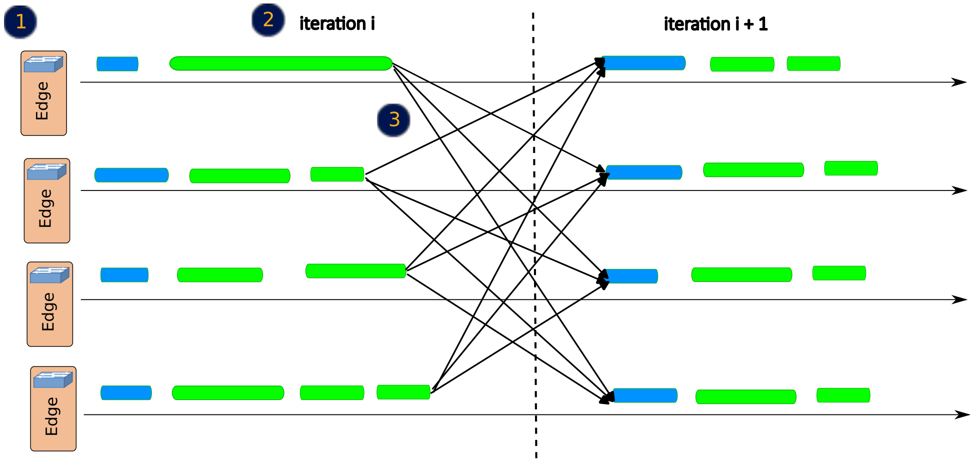

Figure 1. Edge devices in FL-MEC include an edge network device and an edge processing unit (1). The learning process consists of iterations (2). Each iteration (save the initial one) contains three phases: averaging of the received weights (blue boxes), model training (green boxes), and sending model weights. Communication to nodes occurs at the end of an iteration (3). In our case, we send the data to all nodes using a fully peer-to-peer approach

3.1. Data source: Małopolska Educational Cloud (MEC)

In our setup, we utilize Małopolska Educational Cloud (MEC)[1] infrastructure [12, 43]. The project was developed in 2014-2021 to interconnect the Małopolska region high schools and universities. The infrastructure, which was installed in more than 120 schools, 20 university departments, and about 20 other supporting institutions, provided a basis for multiple cooperation scenarios [42]. Most often, high schools in the region use the infrastructure to conduct remote courses organized by universities. Such courses typically include one or two remote classes a week, and their typical duration is one or two semesters. There are a few (typically five) schools participating in the same course, each of them represented by about 20-25 students. In addition to the video sessions, the courses can be supported by additional topic-specific services, hosted in the private cloud, which was also implemented by the project.

MEC infrastructure also provides Internet access for participating schools in a unified way. The schools' infrastructures consist of several (typically 11) access points, a switch that is a central point of the whole network, and a boundary security device, which integrates the functionalities of a router, VPN termination point, and L7

[1] MEC https://e-chmura.malopolska.pl/

firewall and content filter. The primary provider of infrastructure devices (i.e., both networking and multimedia equipment) is Cisco.

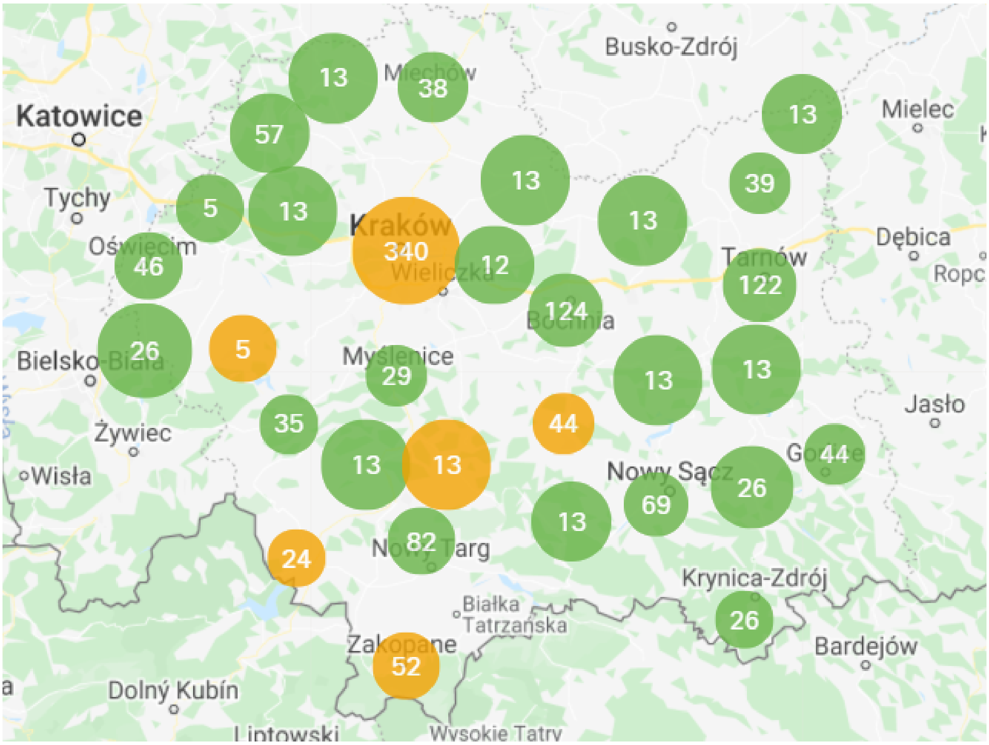

Figure 2. MEC coverage of Małopolska region. The integers indicate the number of devices installed in respective locations

Because the devices installed in respective schools are monitored and managed in a unified and centralized manner and are interconnected by a single device (the switch), the MEC infrastructure provides an opportunity to conduct synchronized experiments in many geographically dispersed locations. The schools are sources of many diversified data flows (towards the Internet) and a few periodic priority flows, which include video and data flows towards the MEC cloud. It is noteworthy that in most cases the development of MEC infrastructure was accompanied by upgrades in Internet connections; the goodput both towards the MEC and Meraki[2] clouds reaches 100 Mbps, and the values of latency and jitter are relatively low (see Figure 3).

[2] Cisco Meraki (meraki.cisco.com), the cloud-based management platform used in MEC

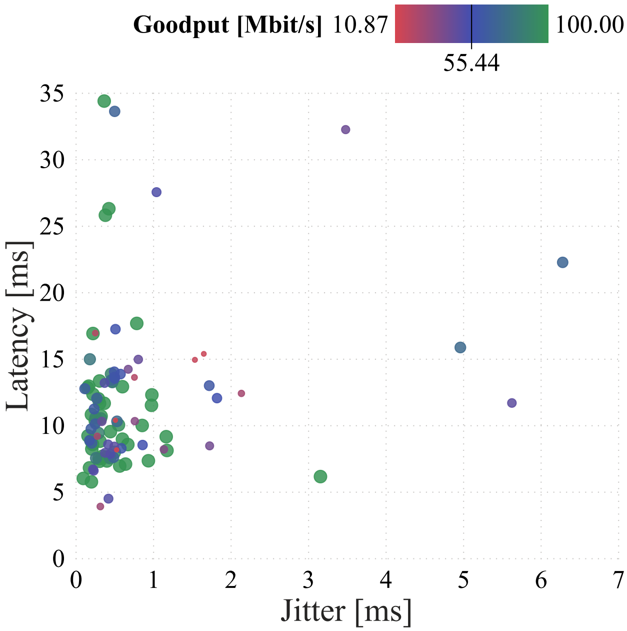

Figure 3. Distribution of most important network metrics

3.2. Data aggregation, detection of flows

The first step in processing that occurs on the edge is the aggregation of individual packets that arrive from the mirroring interface into meaningful flows. It was also essential to identify the source (host in MEC network) and destination (external address) addresses to determine the destination of the flow – whether it was inside or outside MEC infrastructure. For that, we consulted the MEC IP address management system.

We utilize Tshark[3] TCP stream indexer to deal with packet flows. Regarding feature engineering, we calculated additional features that could be helpful in both machine learning and further data analysis (see Table 1).

Thanks to those aggregations, we reduced the data volume significantly (from about 1GB of filtered packet data per batch to about 1MB of aggregated flows). It was essential due to the limited resources of our edge processing nodes.

Finally, we ended up with over 3 million data samples ready for further processing, collected for the duration of over one year.

[3] TShark https://tshark.dev/capture/capture_filters/

Table 1

Aggregated flows data columns description and example entries

Column name	Description	Example entry
tcp_stream_index	TCP stream index provided by Tshark	5
ip_src	Source IP address in packet	**.***.**.***
ip_dst	Destination IP address in packet	***.**.**.*
src_comp	Host IP address in MEC	**.***.**.***
dst_srv	Service IP address	***.**.**.*
tcp_src_port_comp	Host port	5060
tcp_dst_port_srv	Service port	54404
tcp_port	Thsark concatenation of source & destination port	54404,5060
up_len	Upload length (from host in MEC)	263424
up_size	Upload size (from host in MEC)	52836736
down_len	Download length (to host in MEC)	265961
down_size	Download size (to host in MEC)	70529848
flow_start	Flow start time	2024-05-01 00:00:30.679705+02:00
flow_end	Flow end time	2024-05-01 07:55:01.121280+02:00
flow_time	Flow duration	28470
total_data	Total data exchanged	123366584
up_size_by_time	Upload size by flow time (from host in MEC)	1855.874113
down_size_by_time	Download size by flow time (to host in MEC)	2477.339234

3.3. Flows segmentation

As we wanted the dataset to be practical, we decided to create a simple segmentation scheme based on flow times and upload/download data sizes, and ended up with nine categories (see Table 2).

Table 2

Flow segmentation based on established thresholds

No.	Time	Upload	Download	Share
1	Short	Low	Low	37.5%
2	Long	Low	Low	17.6%
3	Long	High	High	11.8%
4	Short	High	High	11.4%
5	Short	Low	High	8.4%
6	Long	Low	High	8.2%
7	Short	High	Low	2.8%
8	Long	High	Low	1.2%
9	Irrevelant	Irrevelant	Irrevelant	0.9%

Next, we adjusted the threshold parameters to have appropriate class sizes. We did not want to have very unbalanced data, but we also did not want to have a perfectly balanced dataset that never appears in real problems. We established different thresholds for the sizes of the data uploaded and downloaded. Flows shorter than 5 seconds were marked as Irrelevant, flows between 5 and 100 seconds as Short, and flows longer than 100 seconds as Long. For both download and upload data sizes, we used 2MB as the splitting point. Short flows with low upload and download values that represent HTTP traffic appeared to be the most common class.

From this point, we focus on an algorithm that predicts the category of flow.

3.4. Service names resolution

We resolved the DNS names from the gathered addresses to better understand the traffic patterns in terms of external services contacted. We did it once a day due to changes in the domain name system for dynamic addresses. Unfortunately, still about 40% of the addresses were unresolved and therefore not categorized. Nonetheless, we were able to identify critical (most important) services for users. The services were located both inside and outside MEC architecture, and of course included those related to MEC multimedia infrastructure.

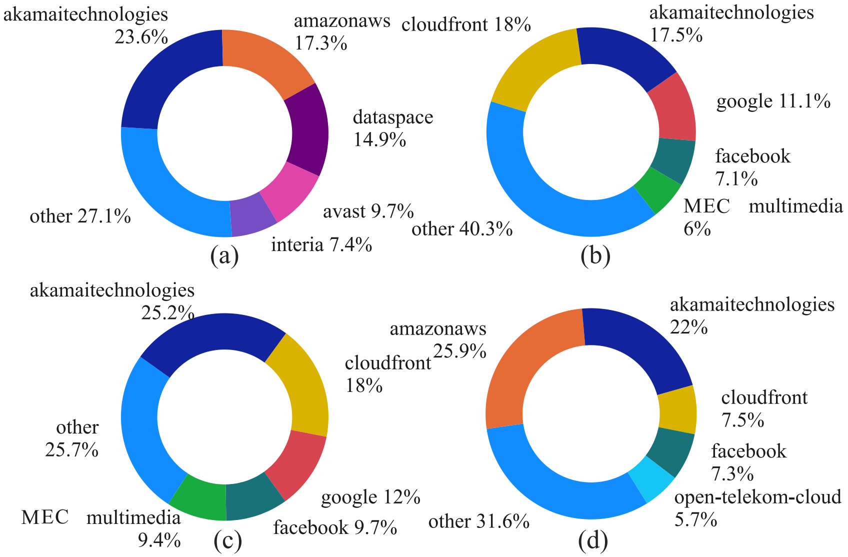

Figure 4. Top 6 services in long flows within MEC based on segmentation. (a) Long flows with high upload and high download; (b) Long flows with high upload and low download; (c) Long flows with low upload and high download; (d) Long flows with low upload and low download

We analyzed them further as they were essential in our use case. Network traffic hitting the servers of MEC multimedia systems differs from the usual one. Almost every flow is long, and over 50% of them have a high upload and download ratio, which could indicate intense activity of teachers and students who were presenting and therefore sending multimedia in both directions. Another considerable part (around 40%) consists of long flows with a high download ratio that could indicate either non-interactive (e.g., file downloads) or less interactive sessions (e.g., with a single, unidirectional video stream).

Within long flows, we can observe that global CDN services play a significant role in the segment (see Figure 4a). For this particular segment, we can see Avast (an antivirus system) in addition to popular local services. It is interesting to see it within this segment - it could indicate that antivirus software updates impact end-device network performance.

We recognized similar shares for CDN services within other segments (see Figure 4c and Figure 4d) except for the segment with high upload and low download rates (see Figure 4b).

There were two major flow segments for MEC multimedia systems servers (see Figure 4), and here we can point out that only for those flow segments, traffic to MEC multimedia systems servers is a significant part of the whole traffic.

3.5. Mirroring MEC in an emulated environment

We prepared an emulated environment that mirrors the existing MEC infrastructure to evaluate our FL approach. We utilized the Containernet[4] platform to build our network environment [31]. It allowed us to specify the existing networking constraints among the nodes. Separate Docker containers emulated respective edge processing nodes of FL-MEC. We set specific resource constraints[5] reflecting the capabilities of the processing unit (we used Raspberry Pi 4 as our reference hardware). In addition, we had full access to the Linux OS on the nodes as well as to the real network stack, so the emulated environment closely mirrored system aspects of the real environment. Before executing FL evaluation, we had been gathering data from five real locations for more than one year.

4. FL-MEC learning model

In our research, we wanted to perform classification using a machine learning model. We did not want to provide a complete machine learning model for real-life applications, but to evaluate the opportunities and limitations of federated learning usage. For the dataset, we used simple threshold-based segmentation of flows described in Section 3.3 to check the feasibility of FL in a distributed environment. We wanted to

[4] Containernet https://github.com/containernet/containernet

[5] Docker: Runtime options with Memory, CPUs, and GPUs https://docs.docker.com/config/containers/resource_constraints/

base our implementation on the Decentralized Federated Averaging due to promising results reported in an original work [39]. The algorithm uses a new decentralization approach, consisting of multiple training steps (epochs) between each communication round.

4.1. Federated averaging

In this section, we present our implementation concept of federated learning based on Decentralized Federated Averaging [39]. In iteration 0, we initialize the models on each node and train the models on local data. Then we send parameters to other nodes and perform averaging. The next iteration trains the models on new local data and performs averaging. Training is performed incrementally.

Algorithm 1 Federated learning on a single node

```
Parameters $E, N, I \in \mathbb{Z}^+$
Initialization: Initialize model
for $i = \{1, 2, \ldots, D\}$ do
    Train model on local data for $E$ epochs
    Send parameters to other $N$ nodes
    Receive parameters from $N$ other nodes
    Perform averaging
end for
```

Algorithm 1 presents the described steps for a single node. It is also important that the training is continuous, due to changes in network flows characteristics. It is worth mentioning that we do not want to use a centralized server for averaging; rather, each node acts both as a client and as a server. During the evaluation, we also modified the algorithm by using weighted averaging based on data amount and by decreasing communication intensity.

4.2. Neural network model

The classification algorithm selection was restricted because it had to be applicable to online training and the provide opportunity for averaging. Neural networks were the first choice as they meet these two requirements. Considering the limited resources, we ended up with a moderate-size neural network (see Figure 5).

We also decided to skip advanced reprocessing or dimension reduction methods like PCA [15] or UMAP [24] because we wanted to evaluate federated averaging, and advanced preprocessing techniques could impact the reliability of our results. The only preprocessing technique we applied was scaling the data using a standard scaler tool provided by the Python Sklearn library. In our case, we wanted to use online learning, which allows to perform incremental training [14], beneficial in FL approach [9].

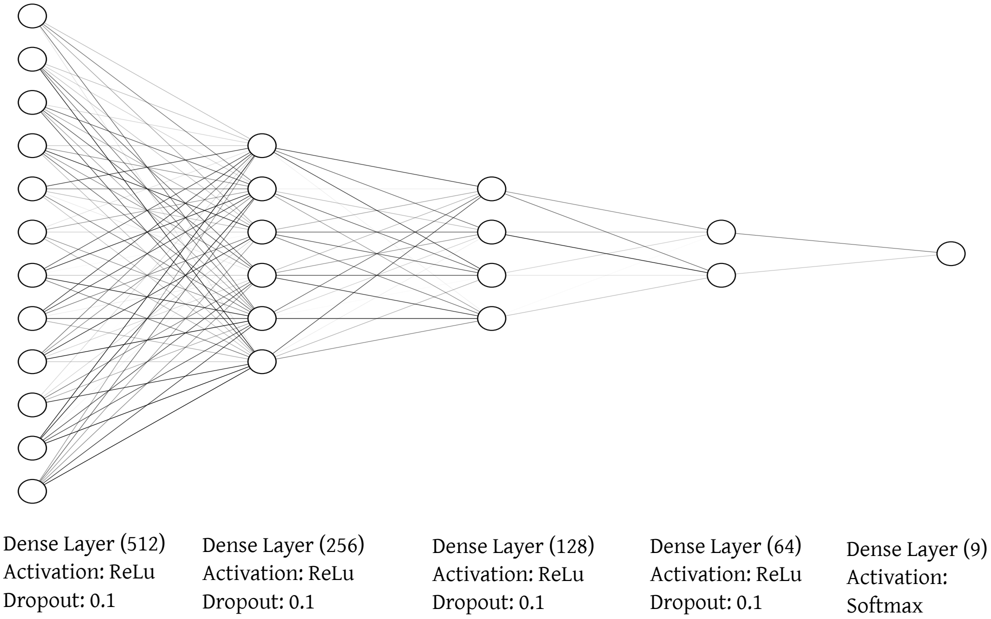

Figure 5. Architecture of a neural network used in flows classification

4.3. Model evaluation

For evaluation, we used accuracy and logarithmic loss metrics. We utilized built-in TensorFlow categorical accuracy, which is just the sum of total correct predictions divided by the sum of test entries. We decided to use logarithmic loss due to its wide application among many works related to machine learning and because of information on prediction certainty.

We distributed our data unevenly to represent the real-world environment, in which some sites are more active, i.e., they participate in more flows than others, and therefore some sensing nodes could access more training data samples than others. Unsurprisingly, the accuracy of local models depended on the volumes of the training data (see Figure 6).

We proposed three versions of federated averaging:

- *simple* – simple averaging of all neural network's layer;
- *weighted* – weighted averaging of all neural network's layers based on training data volume;
- *two-step* – same as above, with restricted synchronization intensity to one communication round per second – to check if communication costs could be reduced.

After the first iteration of all federated models, accuracy drops drastically compared to the values depicted in Figure 6 due to weight averaging (see Figure 7). In this initial stage, models could search for local minima, and the averaging of weights could cause those weights to become meaningless. However, from this stage, all models seem to learn in the correct directions, and averaging weights seems justified and reasonable.

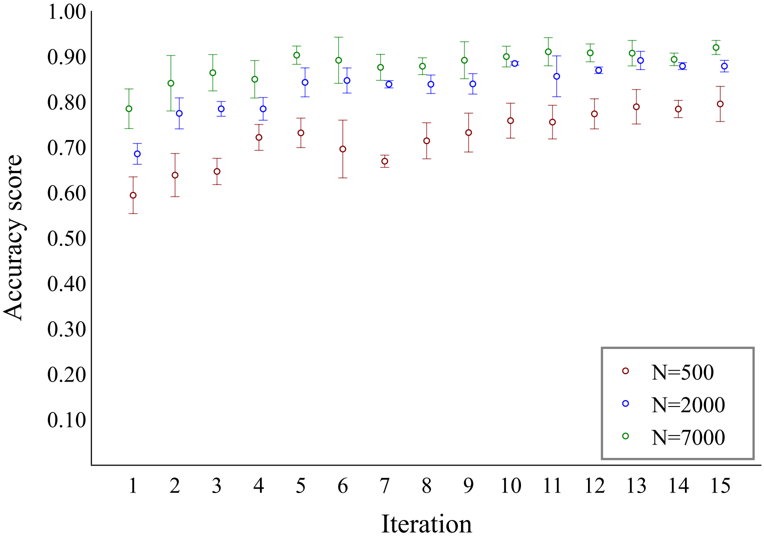

Figure 6. Learning curves for local models (N represents batch size)

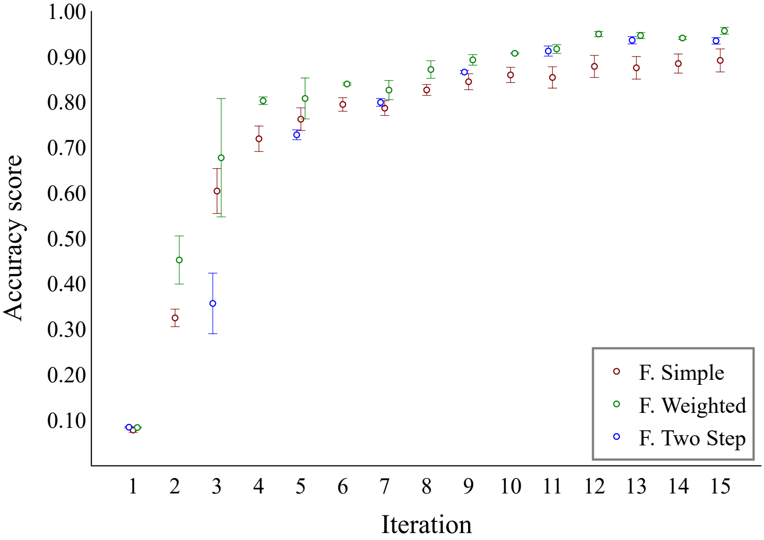

Figure 7. Learning curves for federated models

Compared to local models, the simple federated model was not very successful, but the weighted model outperformed them significantly (see Table 3). In addition, it was even better than the local model N=7000, indicating that federated learning could work similarly to regularization.

We also proposed a weighted model restricted with regard to the communication – only one round was allowed every second. Our goal was to check whether communication could be limited without a significant performance drop, and it turned out that it was feasible.

Table 3
Final accuracy scores and logarithmic losses for models
with different training data batch sizes and for federated model

	Accuracy	Logarithmic Loss
Local model N=500	0.803	0.840
Local model N=2000	0.878	0.389
Local model N=7000	0.924	0.175
Federated model	0.867	0.321
Weighted federated model	0.957	0.117
Two step federated model	0.934	0.149

We presented accuracy scores and logarithmic losses for all models used in our experiment (see Table 3). Analysis of logarithmic losses for models is very similar to the analysis of accuracy we presented above, so we decided to skip that part.

5. Runtime analysis

In our experiment, we also measured the times for both local and federated training methods and compared them. We wanted to check if the durations of iterations (we included the time of loading the dataset from the file to the iteration duration) are the same, and how long averaging would take (we have waiting time for other nodes' weights).

When we compare the results obtained for the federated model with the local models, we can observe that the total duration measured for the federated models is significantly longer than for the local ones (see Table 4).

Table 4
Iterations duration for local models and federated model on representative nodes

	Local model	FL – training	FL – averaging
node 1 N=500	8.4 s	8.3 s	9 min 27 s
node 2 N=2000	47.3 s	48.2 s	8 min 42 s
node 3 N=7000	71.2 s	70.6 s	7 min 27 s

The main reason is the costly averaging process, which has a negative impact, especially for smaller local models. Our experiment was conducted with about 100 emulated nodes with emulated connection parameters modeled on the MEC environment. The averaging overhead is caused by a long waiting time for weights from other nodes, especially larger ones. To sum up, federated learning takes significantly longer than local training. However, based on the scale and available resources in the EC, we believe that it is still acceptable.

The amount of data exchanged during federated learning depends on model size. In our case, the neural network weights size was 867 KB, and during each iteration of Algorithm 1, one node sent its own weights file to the other 100 nodes. Multiplying the weights file size by the number of nodes and the number of iterations (15), we calculated that the total data exchanged in the network during federated learning is about 130 GB (see Table 5). It is quite a considerable network load, and sometimes it causes problems with communication, but appropriate software modifications address it. For example, in a real-world scenario, the training data can be exchanged in periods of low user activity (in MEC, that would be late afternoons and nights).

Table 5

Summary table of transferred data volumes

Weights file size	Data sent per node per iteration	Data sent per node (total)	Data exchanged (total)
0.867 MB	86.7 MB	1.3 GB	130 GB

6. Future work and security considerations

Our research provided interesting insights regarding designing and implementing EC-based FL applications and systems. First, developing a robust acquisition system is challenging. Thorough stress testing of EC capabilities within production environment settings is essential. Issues such as communication interruptions and faulty devices (both measuring appliances and network equipment) were predominant sources of problems encountered in our setup. One interesting approach to further improve the system's fault tolerance and availability is to use a network protocol designed for data exchange under such volatile conditions as the *OMG Data Distribution Service (DDS)* [19, 30].

Second, FL significantly enhances privacy because there is no central server that stores raw data; measuring nodes only share model updates, such as gradients or weight adjustments. Our approach employed a peer-to-peer system where edge nodes directly communicate model updates with each other. However, this introduces additional challenges regarding protecting against data integrity and security attacks. For example, poisoning attacks on data and models have been studied and analyzed [38, 40]. We can partially address the issue by utilizing blockchain technology in our FL process. In our case, it will provide a decentralized and tamper-resistant tracking layer for model updates. In addition, it can enhance data integrity and trust among nodes participating in the FL process. However, integrating blockchain into EC systems also presents challenges. The primary concern involves resource constraints, as devices typically have limited computing power, storage, and energy resources. Blockchain (particularly in proof-of-work configurations) requires additional computational and energy resources, which may be infeasible for constrained EC devices. Furthermore, blockchain implementations can introduce latency and scalability issues due to the time-consuming consensus mechanisms required to validate transactions across nodes in the FL process. We can use lightweight consensus algorithms or deploy hybrid blockchain solutions explicitly tailored for resource-constrained environments to address this issue [17, 22, 41].

Third, data preprocessing is critical for effectively handling data streams on EC devices. Due to limited memory capacity, all aggregation and filtering processes should occur directly on the EC site. One interesting approach to further improve FL-MEC runtime efficiency and resource usage is to leverage event-driven serverless computing and offload resource-intensive computations to other nodes [20]. Additionally, data compression techniques and adaptive sampling methods can be employed to minimize the data volume required for processing (and for transmission).

Finally, correctly selecting an appropriate machine learning algorithm for FL presents another challenge. Only a limited subset of algorithms can efficiently utilize federated learning methodologies. While neural networks are typically the preferred choice due to their flexibility, other algorithms, which support online training, might also be interesting [43]. Moreover, the flexibility of neural networks is an obvious advantage, but it could also be challenging to pick up an appropriate architecture for a task, especially for more advanced problems. Additionally, lightweight releases of popular machine learning frameworks (TensorFlow Lite, PyTorch Mobile [23] or Flower [6]) can be particularly advantageous for EC devices because they are tailored for devices with limited resource capabilities.

7. Conclusions

Our work evaluated a few federated weight averaging techniques, demonstrating their feasibility and significant benefits in a simulated environment. Using a standard neural network architecture for a classification task as a consistent testbed, we showed that federated approaches offer substantial improvements over traditional, isolated training models.

The results highlight the core advantages of federated learning. Our Weighted Federated Model not only outperformed the local models trained on small (N=500) and average (N=2000) datasets, but it also achieved a final accuracy of 0.957, which is significantly higher than the local model trained on the largest dataset (N=7000, accuracy 0.924). This compellingly demonstrates that federated learning serves two critical functions. Firstly, it allows clients with limited data to benefit from the collective knowledge of the network, achieving performance far beyond what would be possible with their local data alone.

Secondly, the superior performance of the federated model suggests a strong regularization effect. By averaging models trained on diverse data distributions, the global model becomes more robust and less prone to overfitting on the specifics of any single client's dataset, leading to better generalization.

Finally, the verification of our approach in a simulated environment with realistic constraints proved that even simple communication protocols based on IP and HTTP are effective for establishing federated networks. In summary, our findings confirm that federated learning is not just a viable approach, but a powerful one that can lead to models that are more accurate and robust than even well-trained isolated models.

Acknowledgements

The research presented in this paper was partially supported by the National Centre for Research and Development (NCBiR), Poland, project no. POIR.01.01.01-00-1515/20 and by the Polish Ministry of Science and Education funds assigned to the AGH University of Krakow.

References

[1] Abreha H.G., Hayajneh M., Serhani M.A.: Federated learning in edge computing: a systematic survey, *Sensors*, vol. 22(2), 450, 2022. doi: 10.3390/s22020450.

[2] Ahmed M., Naser Mahmood A., Hu J.: A survey of network anomaly detection techniques, *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016. doi: 10.1016/j.jnca.2015.11.016.

[3] Anaissi A., Suleiman B., Alyassine W.: Personalised federated learning framework for damage detection in structural health monitoring, *Journal of Civil Structural Health Monitoring*, vol. 13(2), pp. 295–308, 2023.

[4] Asad M., Shaukat S., Hu D., Wang Z., Javanmardi E., Nakazato J., Tsukada M.: Limitations and future aspects of communication costs in federated learning: A survey, *Sensors*, vol. 23(17), 7358, 2023. doi: 10.3390/s23177358.

[5] Azab A., Khasawneh M., Alrabaee S., Choo K.K.R., Sarsour M.: Network traffic classification: Techniques, datasets, and challenges, *Digital Communications and Networks*, vol. 10(3), pp. 676–692, 2024. doi: 10.1016/j.dcan.2022.09.009.

[6] Beutel D.J., Topal T., Mathur A., Qiu X., Fernandez-Marques J., Gao Y., Sani L., *et al.*: Flower: A friendly federated learning research framework, *arXiv preprint arXiv:200714390*, 2020. doi: 10.48550/arXiv.2007.14390.

[7] Bonawitz K., Eichner H., Grieskamp W., Huba D., Ingerman A., Ivanov V., Kiddon C., *et al.*: Towards Federated Learning at Scale: System Design. In: A. Talwalkar, V. Smith, M. Zaharia (eds.), *Proceedings of the 2nd SysML Conference, Palo Alto, CA, USA*, vol. 1, pp. 374–388, 2019. https://mlsys.org/Conferences/2019/doc/2019/193.pdf.

[8] Chen M., Shlezinger N., Poor H.V., Eldar Y.C., Cui S.: Communication-efficient federated learning, *Proceedings of the National Academy of Sciences*, vol. 118(17), e2024789118, 2021. doi: 10.1073/pnas.2024789118.

[9] Dong J., Wang L., Fang Z., Sun G., Xu S., Wang X., Zhu Q.: Federated Class-Incremental Learning. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10154–10163, 2022. doi: 10.1109/CVPR52688.2022.00992.

[10] Fragkou E., Katsaros D.: A joint survey in decentralized federated learning and TinyML: A brief introduction to swarm learning, *Future Internet*, vol. 16(11), 413, 2024. doi: 10.3390/fi16110413.

[11] Gabrielli E., Pica G., Tolomei G.: A survey on decentralized federated learning, *arXiv preprint arXiv:230804604*, 2023. doi: 10.48550/arXiv.2308.04604.

[12] Gandia R.L., Zieliński S., Konieczny M.: Model-based approach to automated provisioning of collaborative educational services. In: *International Conference on Computational Science*, pp. 640–653, Springer, 2021. doi: 10.1007/978-3-030-77980-1_48.

[13] Hernandez-Ramos J.L., Karopoulos G., Chatzoglou E., Kouliaridis V., Marmol E., Gonzalez-Vidal A., Kambourakis G.: Intrusion Detection based on Federated Learning: a systematic review, *ACM Computing Surveys*, vol. 57(12), pp. 1–65, 2025. doi: 10.1145/3731596.

[14] Hoi S., Sahoo D., Lu J., Zhao P.: Online Learning: A Comprehensive Survey, *Neurocomputing*, vol. 459, 2018. doi: 10.1016/j.neucom.2021.04.112.

[15] Hotelling H.: Analysis of a complex of statistical variables into principal components, *Journal of Educational Psychology*, vol. 24(6), p. 417, 1933. doi: 10.1037/h0070888.

[16] Imteaj A., Thakker U., Wang S., Li J., Amini M.H.: A Survey on Federated Learning for Resource-Constrained IoT Devices, *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 2021. doi: 10.1109/JIOT.2021.3095077.

[17] Kaur M., Gupta S.: Performance Evaluation of A Lightweight Consensus Protocol for Blockchain IoT Networks, *Computer Science*, vol. 26(1), 2025. doi: 10.7494/csci.2025.26.1.5483.

[18] Khraisat A., Alazab A., Singh S., Jan T., Jr. Gomez A.: Survey on federated learning for intrusion detection system: Concept, architectures, aggregation strategies, challenges, and future directions, *ACM Computing Surveys*, vol. 57(1), pp. 1–38, 2024. doi: 10.1145/3687124.

[19] Konieczny M.: Enriching WSN environment with context information, *Computer Science*, vol. 13(4), pp. 101–114, 2012. doi: 10.7494/csci.2012.13.4.101.

[20] Konieczny M., Musiał S., Zieliński S.: Hybrid Serverless Processing in Environmental Monitoring – Building Digital Twin. In: *2024 IEEE 65th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*, pp. 1–6, 2024. doi: 10.1109/ITMS64072.2024.10741932.

[21] Lian X., Zhang C., Zhang H., Hsieh C.J., Zhang W., Liu J.: Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In: *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5336–5346, Curran Associates Inc., Red Hook, NY, USA, 2017.

[22] Liu Y., Peng J., Kang J., Iliyasu A.M., Niyato D., Abd El-Latif A.A.: A secure federated learning framework for 5G networks, *IEEE Wireless Communications*, vol. 27(4), pp. 24–31, 2020. doi: 10.1109/mwc.01.1900525.

[23] Luo C., He X., Zhan J., Wang L., Gao W., Dai J.: Comparison and Benchmarking of AI Models and Frameworks on Mobile Devices, 2020. https://arxiv.org/abs/2005.05085.

[24] McInnes L., Healy J., Saul N., Groùberger L.: UMAP: Uniform Manifold Approximation and Projection, *Journal of Open Source Software*, vol. 3(29), p. 861, 2018. doi: 10.21105/joss.00861.

[25] McMahan B., Moore E., Ramage D., Hampson S., Arcas B.A.y.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: A. Singh, J. Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282, PMLR, 2017. https://proceedings.mlr.press/v54/mcmahan17a.html.

[26] Namdev N., Agrawal S., Silkari S.: Recent Advancement in Machine Learning Based Internet Traffic Classification, *Procedia Computer Science*, vol. 60, pp. 784–791, 2015. doi: 10.1016/j.procs.2015.08.238.

[27] Nanor E., Cobbinah M.B., Qinli Y., Junming S., Kobiah C.: FedSULP: A communication-efficient federated learning framework with selective updating and loss penalization, *Information Sciences*, vol. 651, p. 119725, 2023. doi: 10.1016/j.ins.2023.119725.

[28] Nariman G.S., Hamarashid H.K.: Communication overhead reduction in federated learning: a review, *International Journal of Data Science and Analytics*, vol. 19(2), pp. 185–216, 2025.

[29] Papernot N., McDaniel P., Sinha A., Wellman M.P.: SoK: Security and Privacy in Machine Learning. In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 399–414, 2018. doi: 10.1109/EuroSP.2018.00035.

[30] Pardo-Castellote G.: Omg data-distribution service: Architectural overview. In: *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pp. 200–206, IEEE, 2003.

[31] Peuster M., Karl H., van Rossem S.: MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments. In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 148–153, 2016. doi: 10.1109/NFV-SDN.2016.7919490.

[32] Rafique W., Qi L., Yaqoob I., Imran M., Rasool R., Dou W., Rafiq W.: Complementing IoT Services Through Software Defined Networking and Edge Computing: A Comprehensive Survey, *IEEE Communications Surveys & Tutorials*, vol. PP, pp. 1–1, 2020. doi: 10.1109/COMST.2020.2997475.

[33] Ren Y., Cao Y., Ye C., Cheng X.: Two-layer accumulated quantized compression for communication-efficient federated learning: TLAQC, *Scientific Reports*, vol. 13(1), p. 11658, 2023. doi: 10.1038/s41598-023-38916-x.

[34] Rezaei S., Liu X.: Deep Learning for Encrypted Traffic Classification: An Overview, *IEEE Communications Magazine*, vol. 57(5), pp. 76–81, 2019. doi: 10.1109/mcom.2019.1800819.

[35] Ring M., Wunderlich S., Scheuring D., Landes D., Hotho A.: A survey of network-based intrusion detection data sets, *Computers & security*, vol. 86, pp. 147–167, 2019. doi: 10.1016/j.cose.2019.06.005.

[36] Said S., Ihab R., Hesham O., Tabl I.A., Maged N., Youssef S., Elagamy M.: AIOT-Arch: Furthering Artificial Intelligence in Big Data IoT Applications, *IOP Conference Series: Materials Science and Engineering*, vol. 1051(1), p. 012008, 2021. doi: 10.1088/1757-899X/1051/1/012008.

[37] Samie F., Bauer L., Henkel J.: Hierarchical Classification for Constrained IoT Devices: A Case Study on Human Activity Recognition, *IEEE Internet of Things Journal*, vol. PP, 2020. doi: 10.1109/JIOT.2020.2989053.

[38] Sharma A., Marchang N.: A review on client-server attacks and defenses in federated learning, *Computers & Security*, p. 103801, 2024. doi: 10.1016/j.cose.2024.103801.

[39] Sun T., Li D., Wang B.: Decentralized Federated Averaging, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45(4), pp. 4289–4301, 2023. doi: 10.1109/TPAMI.2022.3196503.

[40] Xia G., Chen J., Yu C., Ma J.: Poisoning attacks in federated learning: A survey, *IEEE Access*, vol. 11, pp. 10708–10722, 2023. doi: 10.1109/access.2023.3238823.

[41] Xu G., Kong D.L., Zhangs K., Xu S., Cao Y., Mao Y., Duan J., Kang J., Chen X.B.: A model value transfer incentive mechanism for federated learning with smart contracts in AIoT, *IEEE Internet of Things Journal*, vol. 12(3), pp. 2530–2544, 2024. doi: 10.1109/jiot.2024.3468443.

[42] Zieliński K., Czekierda Ł., Malawski F., Straś R., Zieliński S.: Recognizing value of educational collaboration between high schools and universities facilitated by modern ICT, *Journal of Computer Assisted Learning*, vol. 33(6), pp. 633–648, 2017. doi: 10.1111/jcal.12207.

[43] Zygmunt M., Konieczny M., Zielinski S.: Accuracy of statistical machine learning methods in identifying client behavior patterns at network edge. In: *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pp. 575–579, 2019. doi: 10.1109/TSP.2019.8768885.

Affiliations

Patryk Paszko
Patryk Paszko ScData, Krakow, Poland, ktppaszko@gmail.com

Marek Konieczny
AGH University of Krakow, Faculty of Computer Science, Krakow, Poland, marekko@agh.edu.pl

Sławomir Zieliński
AGH University of Krakow, Faculty of Computer Science, Krakow, Poland, slawek@agh.edu.pl

Bartosz Kwolek
AGH University of Krakow, Faculty of Computer Science, Krakow, Poland, kvoloo@gmail.com

Received: 30.04.2025
Revised: 27.10.2025
Accepted: 1.12.2025