A Multi-Faceted Approach to Maritime Security: Federated Learning, Computer Vision, and IoT in Edge Computing

Mateus R. Cruz, Samuel B. Mafra, Eduardo H. Teixeira and Felipe A. P. Figueiredo

***Abstract*— Brazil faces several challenges related to maritime monitoring, including illegal fishing, drug trafficking, oil spills, and other environmental hazards. The approach aims to automate the detection and analysis of maritime activity, enabling faster and more accurate decision-making for improved maritime security. The computer vision algorithms used in the approach are designed for object detection and classification, and the federated learning approach is used to train the models while preserving data privacy. The results obtained by the final computer vision model and algorithm are presented, and the challenges faced in developing robust CV models and improving classification accuracy are discussed. The proposed approach offers a promising direction for improving maritime security and inspires further research in this area.**

***Keywords*— Internet of Things, Federated Learning, Computer Vision, Maritime Surveillance.**

I. Introduction

Monitoring the maritime environment is a critical task for ensuring maritime safety, security, and environmental protection. Brazil, as a country with a long coastline and extensive maritime borders, faces several challenges like illegal fishing, drug trafficking, oil spills, and other environmental hazards related to maritime monitoring [1]. However, recent advances in Computer Vision (CV) and artificial intelligence (AI) have opened up new possibilities for improving the efficiency and effectiveness of maritime monitoring systems. These technologies can help to automate the detection and analysis of maritime activity, allowing for faster and more accurate decision-making [2].

At the same time, these monitoring systems generate and handle large volumes of critical data. At the same time, it is also highly sensitive and needs to be protected against unauthorized access, manipulation, and theft. Federated Learning (FL), a Machine Learning (ML) technique that allows for decentralized model training without sharing raw data, can help address some of these challenges. In this approach, data remains on local devices or servers, and only model updates are shared between devices, allowing collaborative model training without exposing sensitive data to potential threats [3]. In such dense and remote scenarios, it is often not feasible to transfer all the data to a central server for model training due to limited bandwidth and energy constraints. FL, which enables distributed devices to collaboratively learn a global model without sharing their data, is therefore an attractive approach.

Furthermore, the Federated Average (FedAvg) algorithm can be customized to accommodate the specific constraints and requirements of IoT devices. By utilizing the FedAvg algorithm [4], IoT applications can achieve better accuracy, energy efficiency, and scalability while maintaining data privacy and security. This combination allied with the correct architecture allows the implementation of edge CV capabilities in low-power devices in IoT scenarios. The implemented architecture needs to be powerful and the same time lightweight to support IoT devices characteristics. SqueezeNet is a lightweight neural network architecture that has been designed to run efficiently on low-power and resource-constrained devices [5]. This makes it an ideal choice for implementing CV capabilities in IoT devices where resources such as processing power, memory, and battery life are limited.

This article presents an IoT application that runs on the edge and is suitable for scenarios where data is sensitive. Also, results presented by the final CV model and algorithm are shown, and conclusions about the application are discussed.

II. Related Works

Researchers have developed classification algorithms to identify different types of ships based on RGB images. Where diverse algorithms including a bag of features, support vector machines (SVM), and convolutional neural networks (CNN) are used [6]. On the other hand, VGG19 has presented a better accuracy than capture by the cited technique, reaching a 95.8% rate of accuracy [7]. Furthermore, image processing via CV models can face several problems regarding CV illumination effects, potential occlusion, orientation, scale, and variety of objects [8]. Not only that, the privacy and security of the

Mateus Cruz, Instituto Nacional de Telecomunicações (Inatel), Santa Rita do Sapucaí (MG), e-mail: mateusrc@dtel.inatel.br; Samuel Baraldi Mafra, Departamento de Engenharia de Telecomunicações, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: samuelbmafra@inatel.br; Eduardo Henrique Teixeira,Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: eduardoteixeira@dtel.inatel.br; Eduardo Henrique Teixeira,Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: eduardoteixeira@dtel.inatel.br; Felipe Augusto Pereira de Figueiredo, Departamento de Engenharia de Telecomunicações, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, e-mail: felipe.figueiredo@inatel.br;
This work is partially supported RNP, with resources from MCTIC, Grant No. 01245.020548/2021-07, under the Brazil 6G project of the Radiocommunication Reference Center (Centro de Referência em Radiocomunicações - CRR) of the National Institute of Telecommunications (Instituto Nacional de Telecomunicações - Inatel), Brazil, Huawei, under the project Advanced Academic Education in Telecommunications Networks and Systems, contract No PPA6001BRA23032110257684, Brazil, the National Council for Scientific and Technological Development-CNPq (403827/2021-3), FAPESP (2021/06946-0),and by Minas Gerais State Agency for Research and Development (FAPEMIG) via Grant No. TEC - APQ-03283-17

captured data must be taken into consideration for commercial and private uses [9]. Currently, much of the work found in the literature takes into consideration only technical aspects and performance metrics of the models [10].

Related to the FL, several researchers already explored the application of this training method in the monitoring system. Also, researchers already propose FL as a solution to the challenges of traditional ML approaches in maritime environments [11]. In fact, FL has become a key solution for building ML models that rely on distributed datasets, especially in emerging networks like IoT systems. However, the heterogeneous nature of devices and models in complex IoT networks has made it challenging for FL to perform well, requiring the development of new algorithms, models, and protocols [12]. In addition, IoT in Maritime Transportation Systems (MTSs) is creating new revenue opportunities and improving efficiency while reducing costs. By enabling real-time tracking of shipments, pre-emptive maintenance, route optimization, reduced fuel consumption, and improved safety, the IoT is transforming MTSs into a data-driven and digitalized industry. The solution to challenges faced by existing wireless federated learning (WFL) schemes in unstable wireless channel conditions is already proposed. The proposed solution is a software-defined empowered efficient WFL architecture with embedding Low-Density Parity-Check. (LDPC) communication coding, which improves the anti-interference ability and GPU-CPU acceleration ability during wireless transmission [13].

III. Development, Training, and Implementation

The proposed application relies on heterogeneous devices both in terms of software and hardware. Firstly, two Raspberry Pi 4B boards along with an Nvidia Jetson Nano board were used as End Nodes of the network, while an Ideapad Gaming 3I was used as an aggregator server in the application.

First of all, the two Raspberry Pi 4B boards do not have a Graphics Processing Unit (GPU), and must do all visual processing of the images directly in their processor. The two boards have a Broadcom (Cortex-A72 quad-core of up to 1.5 GHz), while the Nvidia Jetson Nano board has a quad-core core Arm A57 processor of up to 1.43 GHz and the Notebook has an Intel Core i5-10300H processor of up to 4.50 GHz. However, the Nvidia Jetson Nano and Ideapad Gaming 3I devices have graphics processing support that enables more efficient and faster image processing.

Regarding software, the Linux Operating System (OS) was used on all the devices, changing only the distros installed on each one. The Raspberry Pi boards have Raspberry PI O.S., Idealpad Gaming 3i with Arch Linux, and the Nvidia Jetson Nano board with Ubuntu 20.04. The main framework used to build the FL was Flower (FL Framework), which is an open-source hold-out framework for building federated training systems. The framework provides a set of abstractions and Application Programming Interface (APIs) for building systems and it is designed to be flexible and extensible, allowing users to customize various aspects of the training process, such as the communication protocol, optimization algorithm, and model architecture. Figure 1 illustrates the application.

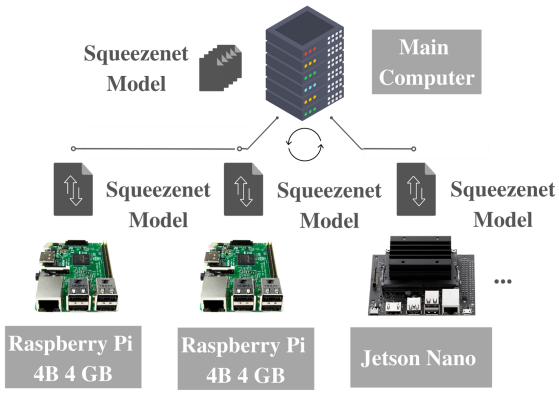

Fig. 1. Application simplified topology used in the FL approach.

The Federated Learning (FL) approach enables the implementation of artificial intelligence techniques in heterogeneous devices, as in the scenario presented here. The Raspberry Pi 4B 4 GB (referred to as Raspberry Pi 4 I) has relatively simple hardware, lacks GPU processing capabilities, but offers 4 GB of RAM for online image processing. These limitations constrain some characteristics of the model and the training process, including slow training speed, reduced batch sizes, and limited model complexity. In contrast, the Raspberry Pi 4B 8GB (referred to as Raspberry Pi 4 II) provides more RAM for image processing during model training, thereby increasing the capabilities of the device.

However, it is important to allocate the devices appropriately in real-world scenarios, considering the specific processing requirements of each location. For instance, the Raspberry Pi 4 I could be implemented in coastal, beach, pier, or harbor environments with a relatively low number of ships. The Raspberry Pi 4 II is suitable for intermediate-level environments where the number and movement of ships is not insignificant, but not overly complex. On the other hand, the Jetson Nano should be deployed in locations with a large number of images and high computational power requirements.

An extremely important aspect of a good implementation of an FL is the correct choice of aggregation strategy. The proposed application makes use of Federated Averaging (FedAvg). FedAvg is a simple and effective algorithm for training ML models in an environment. It has been successfully used in a variety of applications, including image classification [14], natural language processing [15], and much more. The FedAvg algorithm used in the application works as follows:

1) **Initialization**: The server initializes a global model, which is a set of weights for the CV model that will be trained.
2) **Client Selection**: The server randomly selects a subset of clients to participate in the training process.
3) **Local Training**: Each selected client downloads the current global model from the server, trains the model on its local data, and computes an update to the model's weights.

4) **Weight Aggregation**: Each client sends its model update back to the server, which aggregates the updates to compute a new global model.
5) **Repeat**: The server repeats the process of client selection, local training, and weight aggregation for a fixed number of rounds or until convergence.

The Equation III presented below is the one used for weighting and aggregating the weights sent by clients. In the equation, ω_{t+1} is the updated global model at the end of round $t+1$ in FedAvg. The sum over k represents the aggregation of the updates from all K clients in the random subset S_t selected by the server in round t.

$$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^k.$$

These are two scripts in Python, one for the aggregator server and one for the other clients. Starting with the server script, where the server aggregates update from the clients to create a global model. The server initializes the global model at ω_0 and iterates over rounds, selecting a subset of clients S_t in each round. The clients in S_t update their local model using the current global model ω_t and return the updated model to the server. The server then averages the updates from all clients to produce the new global model ω_{t+1}.

The client code implements the local update step for each selected client thought the Algorigtms 2. The client partitions its data into batches and performs several local epochs of gradient descent using batch size B and learning rate η. The updated model w is then returned to the server for aggregation. Overall, FedAvg enables distributed training of ML models while keeping the data local to each client, which can help address privacy concerns and reduce communication costs.

Algorithm 1 Server

initialize ω_0
for each round t = 1,2, ... **do**
  $N \leftarrow n$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ in parallel **do**
    $\omega_{t+1}^k \leftarrow \text{ClientUpdate}(k, \omega_t)$
  end for
  $\omega_{t+1}^k \leftarrow \omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_{t+1}^k$
end for

Algorithm 2 Client

$\beta \leftarrow$ (split P_k into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \beta$ **do**
    $\omega \leftarrow \omega - \eta \bigtriangledown (w; b)$
  end for
end for
return w to server

The communication protocol used for the joint learning of the devices was gRPC. gRPC is a high-performance, open-source framework developed by Google that is used to build fast and efficient remote procedure call (RPC) systems. In a system that uses gRPC, the server acts as the gRPC service provider, while the clients act as the gRPC clients. The server exposes a set of gRPC methods that the clients can call, such as "get_model" or "update_weights". The clients can also expose their own gRPC methods, such as "get_data" or "upload_gradients".

Finally, the architecture used for ship classification was SqueezeNet, a deep neural network architecture for image classification that was introduced in 2016. The key innovation of SqueezeNet is its ability to achieve state-of-the-art accuracy on image classification tasks while using a much smaller number of parameters compared to other deep neural networks. The architecture of the neural network consists of a series of convolutional layers followed by pooling layers and fully connected layers. The convolutional layers use a combination of 1x1 and 3x3 filters to extract features from the input image, while the pooling layers reduce the spatial dimensions of the feature maps. The most distinctive feature of SqueezeNet is its use of "fire" modules, which are composed of a squeeze layer and an expanded layer. The squeeze layer consists of 1x1 convolutions that reduce the number of input channels, while the expanding layer consists of a combination of 1x1 and 3x3 convolutions that increase the number of output channels. Figure 2 illustrates the SqueezeNet architecture used in the application:

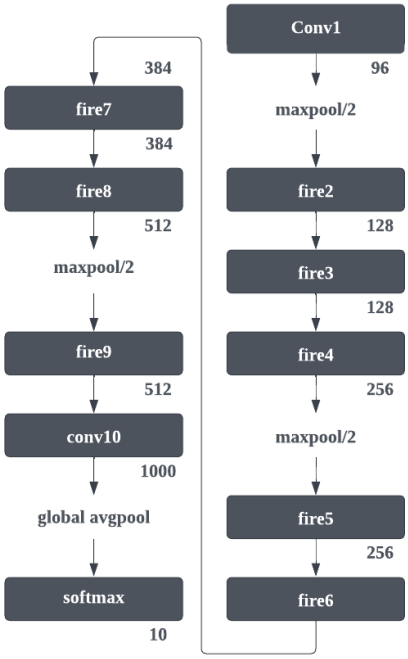

Fig. 2. Squeezenet simplified architecture illustration.

A dataset present in the Kaggle Data Science community was used to create the final dataset presented for the model during its training and validation. This is because several data augmentation techniques were added to extend the original image base, among them:

- **Brightness**: Between -50% (a.a) and +50% (a.b);
- **Noise**: Up to 10% of pixels (b).

- **Crop**: 0% Minimum Zoom, 20% Maximum Zoom (c);
- **Blur**: Up to 3px (d);

The Figure 3 visually presents an image with the data augmentation as example. The dataset had an initial size of 6596 images, while the final dataset had a total of 17648 images. This difference shows an increase of 167.56%, greatly increasing the variety of images presented to the model during training.

Fig. 3. Data augmentation dataset samples.

First of all, it should be taken into consideration that the final dataset must be divided equally for the three devices. For, the training of the model via FL takes place individually on each available board in the system. Therefore, the amount of images available on each board must be equal. Another important aspect to be highlighted regarding the training is the equal division of the dataset into identical training and validation dataset. Not only that, the choice of images to be transferred to each device should be random. For this, a Python algorithm was used to create the individual dataset for each device.

As for splitting the dataset, 80% of the dataset was used for training and 20% for testing the individual devices. Also, the same parameters for learning rate and epochs were used. The optimizer selected for training the models was Adam. Adam is a popular optimization algorithm for training neural networks, which stands for Adaptive Moment Estimation. It is known for its efficiency and robustness in deep learning and has become one of the most widely used optimization algorithms. Epochs and rounds are two other key concepts used to describe the training process of a model across multiple devices or clients. An epoch refers to a single pass of the training data through the model. During each epoch, the model makes predictions on the training data, calculates the loss, and updates the model parameters to improve its accuracy only locally. But, a round can be thought of as a collaborative learning iteration between multiple client devices and a central server. During a round, each client device trains its local ML model on its own data, and the central server aggregates the updated model parameters from all client devices to improve the global model. As for the configuration used in training the models, this is shown in Table I.

TABLE I
THE PARAMETERS USED DURING THE TRAINING.

Parameters	Value
Epochs	2
Rounds	5
Learning Rate	0.001

Three metrics were used to validate the performance of the final model, being (I) Loss, (II) Error Rate, and (III) Accuracy. The Loss metric measures the difference between the predicted output and the actual output of the model, while the Error Rate metric measures the proportion of incorrectly classified samples in the dataset. The accuracy, on the other hand, refers to the proportion of correctly classified samples in the dataset. Figure 4 illustrates the Error Rate (f.a), Accuracy (f.b), and Loss Function per epoch presented by the model during the model training for validation (f.c) and training (f.d) set. The curves present in the graphs show the behavior of the three devices throughout the training individually. However, it should be taken into account that every two epochs FL occurs and the aggregator server receives the individual weights of each model, aggregates them and returns a global model for all of them.

From the curves presented it is possible to extract that the model performed during task learning, achieving a reasonably low error rate of 3.25% and at the same time a low loss of 10.81%. The accuracy presents a satisfactory result, reaching 98% during the training. However, the model seems to have reached a plateau zone in the neighborhood of 10%, failing to reach lower values in the validation dataset. Meanwhile, the loss on the training dataset continues to decrease, presenting a possible over-fitting of the model.

IV. Conclusions

This paper proposes a multi-faceted approach to maritime security that uses FL, CV, and IoT in edge computing. The approach can automate the detection and analysis of maritime activity, improving decision-making for better maritime security. The approach has advantages such as preserving data privacy, reducing manual intervention, and processing data closer to the source. However, challenges such as developing robust CV models, improving classification accuracy, and addressing over-fitting issues need to be addressed in future work. The proposed approach offers a promising direction for improving maritime security and inspires further research in this area.

References

[1] C. Bueger, “What is maritime security?,” Marine Policy, vol. 53. Elsevier BV, pp. 159–164, Mar. 2015. doi: 10.1016/j.marpol.2014.12.005.

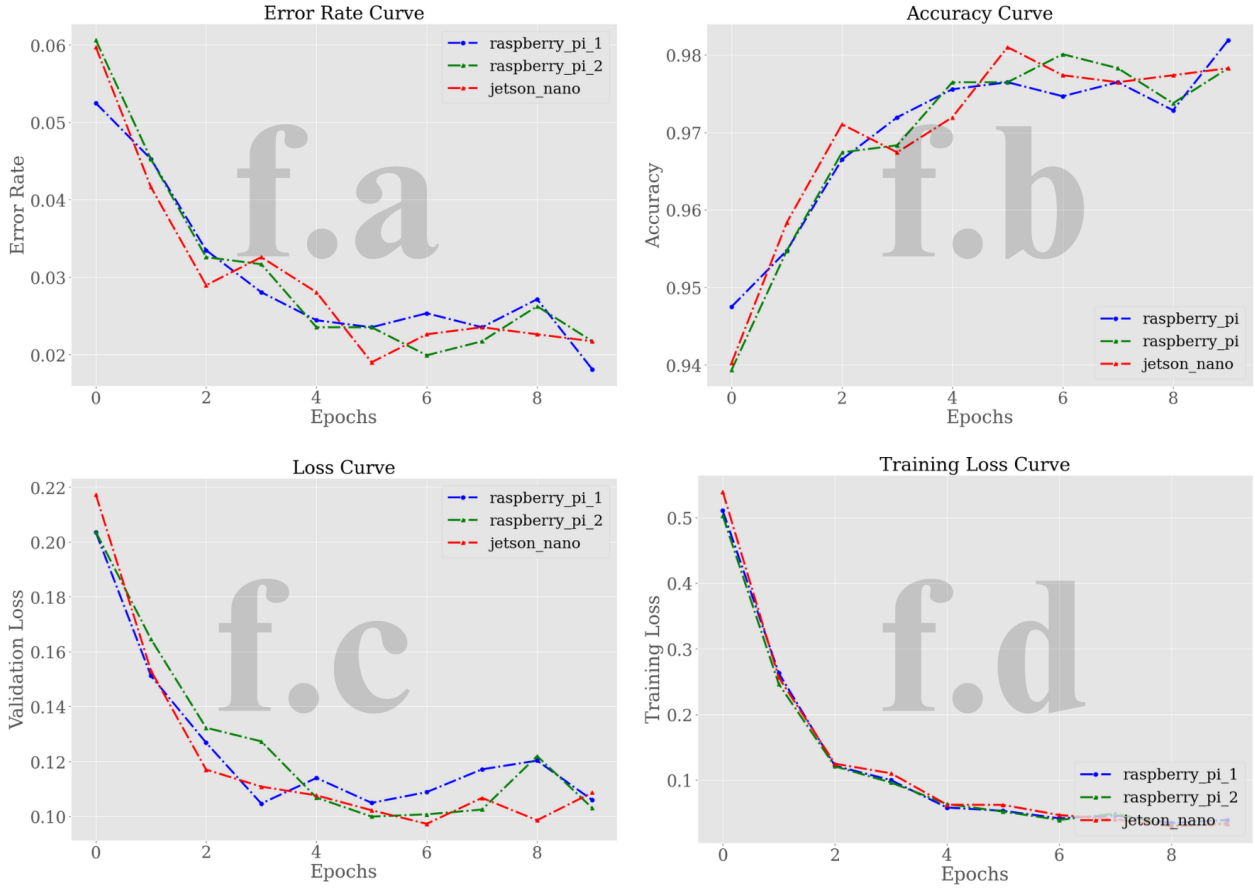

Fig. 4. Curves for Error Rate (Graphic f.a), Accuracy (Graphic f.b), and Loss (Graphic f.c for Validation set and Graphic f.d for Training set) presented by all devices during during the FL training.

[2] Q. Li and L. Yang, "The Key Technologies of Marine Multiobjective Ship Monitoring and Tracking Based on Computer Vision," Mobile Information Systems, vol. 2022. Hindawi Limited, pp. 1–7, May 17, 2022. doi: 10.1155/2022/9582701.

[3] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, "A survey of federated learning for edge computing: Research problems and solutions," High-Confidence Computing, vol. 1, no. 1. Elsevier BV, p. 100008, Jun. 2021. doi: 10.1016/j.hcc.2021.100008.

[4] T. Sun, D. Li, and B. Wang, "Decentralized Federated Averaging," IEEE Transactions on Pattern Analysis and Machine Intelligence. Institute of Electrical and Electronics Engineers (IEEE), pp. 1–12, 2022. doi: 10.1109/tpami.2022.3196503.

[5] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size." arXiv, 2016. doi: 10.48550/ARXIV.1602.07360.

[6] Q. Shi, W. Li, R. Tao, X. Sun, and L. Gao, "Ship Classification Based on Multifeature Ensemble with Convolutional Neural Network," Remote Sensing, vol. 11, no. 4. MDPI AG, p. 419, Feb. 18, 2019. doi: 10.3390/rs11040419.

[7] Z. Hui, C. Na and L. ZhenYu, "Combining a Deep Convolutional Neural Network with Transfer Learning for Ship Classification," 2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA), Xiangtan, China, 2019, pp. 16-19, doi: 10.1109/ICICTA49267.2019.00011.

[8] D. K. Prasad, C. K. Prasath, D. Rajan, L. Rachmawati, E. Rajabaly, and C. Quek, "Challenges in video based object detection in maritime scenario using computer vision." arXiv, 2016. doi: 10.48550/ARXIV.1608.01079.

[9] B. Nagy et al., "Privacy-preserving Federated Learning and its application to natural language processing," Knowledge-Based Systems, vol. 268. Elsevier BV, p. 110475, May 2023. doi: 10.1016/j.knosys.2023.110475.

[10] D. K. Prasad, H. Dong, D. Rajan, and C. Quek, "Are Object Detection Assessment Criteria Ready for Maritime Computer Vision?," IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 12. Institute of Electrical and Electronics Engineers (IEEE), pp. 5295–5304, Dec. 2020. doi: 10.1109/tits.2019.2954464.

[11] A. Giannopoulos, P. Gkonis, P. Bithas, N. Nomikos, G. Ntroulias, and P. Trakadas, "Federated Learning for Maritime Environments:Use Cases, Experimental Results, and Open Issues." Institute of Electrical and Electronics Engineers (IEEE), Feb. 23, 2023. doi: 10.36227/techrxiv.22133549.v1.

[12] J.-P. A. Yaacoub, H. N. Noura, and O. Salman, "Security of federated learning with IoT systems: Issues, limitations, challenges, and solutions," Internet of Things and Cyber-Physical Systems, vol. 3. Elsevier BV, pp. 155–179, 2023. doi: 10.1016/j.iotcps.2023.04.001.

[13] Z. Li, Y. Hong, A. K. Bashir, Y. D. Al-Otaibi, and J. Wu, "Software-Defined GPU-CPU Empowered Efficient Wireless Federated Learning With Embedding Communication Coding for Beyond 5G," IEEE Open Journal of the Communications Society, vol. 4. Institute of Electrical and Electronics Engineers (IEEE), pp. 990–1000, 2023. doi: 10.1109/ojcoms.2023.3266444.

[14] A. Danilenka, M. Ganzha, M. Paprzycki, and J. Mańdziuk, "Using adversarial images to improve outcomes of federated learning for non-IID data." arXiv, 2022. doi: 10.48550/ARXIV.2206.08124.

[15] A. Das, M. Degeling, X. Wang, J. Wang, N. Sadeh, and M. Satyanarayanan, "Assisting Users in a World Full of Cameras: A Privacy-Aware Infrastructure for Computer Vision Applications," 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE, Jul. 2017. doi: 10.1109/cvprw.2017.181.