UAV-assisted Distributed Learning for Environmental Monitoring in Rural Environments

Vukan Ninkovic,*† Dejan Vukobratovic,* Dragisa Miskovic †
*University of Novi Sad, Novi Sad, Serbia
†The Institute for Artificial Intelligence Research and Development of Serbia, Novi Sad, Serbia

***Abstract*—Distributed learning and inference algorithms have become indispensable for IoT systems, offering benefits such as workload alleviation, data privacy preservation, and reduced latency. This paper introduces an innovative approach that utilizes unmanned aerial vehicles (UAVs) as a coverage extension relay for IoT environmental monitoring in rural areas. Our method integrates a split learning (SL) strategy between edge devices, a UAV and a server to enhance adaptability and performance of inference mechanisms. By employing UAVs as a relay and by incorporating SL, we address connectivity and resource constraints for applications of learning in IoT in remote settings. Our system model accounts for diverse channel conditions to determine the most suitable transmission strategy for optimal system behaviour. Through simulation analysis, the proposed approach demonstrates its robustness and adaptability, even excelling under adverse channel conditions. Integrating UAV relaying and the SL paradigm offers significant flexibility to the server, enabling adaptive strategies that consider various trade-offs beyond simply minimizing overall inference quality.**

***Index Terms*—IoT, UAV, distributed learning, environmental monitoring**

I. Introduction

The pervasive issue of pollution, stemming from various sources such as industrial activities, transportation emissions, and waste disposal, poses significant challenges to the environment, raising concerns about its impacts [1]. Ensuring health and hygiene is vital for both humanity's sustainability and a nation's progress, that is dependent on a clean, hazard-free environment. Therefore, monitoring these aspects is essential to promote a healthy life for citizens, especially in rural and underdeveloped environments. In recent years, the integration of IoT for environmental monitoring in rural areas has emerged, utilizing interconnected devices equipped with diverse sensors to gather real-time data on crucial environmental parameters like air quality, soil moisture, water quality, temperature, and humidity [1], [2]. These devices employ wireless communication technologies such as Wi-Fi, cellular networks, LoRaWAN, or satellite connectivity to transmit data to centralized servers or cloud platforms for storage, analysis, and further processing [2].

Unmanned aerial vehicles (UAVs) have found widespread applications across various industries, governmental bodies, and commercial sectors, performing tasks ranging from telecommunications, rescue operations, to surveillance [3], [4]. Notably, they have gained significant attention for their pivotal role in enabling end-to-end wireless communications, especially in providing connectivity to wireless (IoT) devices in remote or rural areas where traditional cellular coverage is scarce or absent [4]. Specifically, access points integrated onto UAVs are being proposed as a potential solution for anticipated data demand and congestion challenges in future wireless networks [5]. Unlike conventional static infrastructure, UAV networks offer the advantage of flexible deployment, enabling them to te coverage [6].

In this work, we propose a novel approach for distributed learning in IoT environmental monitoring scenarios. Our method integrates the split learning (SL) paradigm with UAV relaying in an IoT network, enhancing data transmission rates and ensuring equitable distribution of computational tasks between edge devices, UAV, and server. Furthermore, our approach enhances system adaptability by empowering the server to determine the optimal transmission strategy based on current channel conditions and specific performance metrics such as latency, throughput, and energy efficiency. Numerical results demonstrate that by utilizing proposed distributed learning approach, IoT system shows great robustness to different channel conditions, simultaneously achieving high performance in terms of estimation accuracy.

II. Background

A. IoT Connectivity in Rural Areas

According to [7], IoT connectivity mostly depends on the level of development of the country. More precisely, in developed countries, rural areas are typically accessible via transportation networks, such as railroad networks, and power is supplied through the electricity grid. However, the challenge lies in mobile operators obtaining a satisfactory return on investment (ROI) for providing backhaul to these areas. Conversely, in developing countries, particularly impoverished areas, the challenge is to bridge the digital divide with developed nations. In rural areas, essential services like healthcare and education depend on connectivity, but inadequate transportation infrastructure isolates villages from major cities, while power generation often relies on local sources. Establishing backhaul in such areas, starting from scratch and facing limited revenue due to poverty, may require state subsidies to emphasize the necessity for cost-effective solutions.

This work has received funding from the Horizon 2020 research and innovation staff exchange grant agreement No 101086387, and from the Science Fund of the Republic of Serbia, grant number 6707, REmote WAter quality monitoRing anD IntelliGence – REWARDING

In the context of providing connectivity in rural areas, especially in developing countries, unmanned aerial vehicles (UAVs) equipped with communication equipment can play a significant role [8]. These drones can serve as flying base stations, establishing temporary or permanent connectivity in areas where traditional infrastructure deployment is impractical or cost-prohibitive. By flying over remote regions, drones can establish wireless links between users and the broader network infrastructure, effectively extending backhaul links to underserved communities [9]. In regions with underdeveloped or non-existent transportation infrastructure, such as in impoverished areas of developing countries, drones offer a versatile and efficient means of providing backhaul links [7]. They can be swiftly deployed and are adaptable to changing conditions, making them valuable in emergency situations or areas with limited access to resources [10].

B. Split Learning

Split learning (SL) [11], [12] is a new distributed learning paradigm, which divides a neural network F (consisting of L layers) into sequential layers across multiple participants, like an edge device and a server. In SL, the edge device shares its training dataset securely with the server, which oversees the training process and handling most computational tasks. This distributed approach accelerates convergence and reduces bandwidth constraints [12].

SL separates model training and inference processes. During training, data remains within individual edge devices to prevent raw information transmission across the network. The neural network can be represented as $F = (f_{\mathrm{E}}, f_{\mathrm{S}})$, where $f_{\mathrm{E}} : \mathbb{R}^N \rightarrow \mathbb{R}^M$ and $f_{\mathrm{S}} : \mathbb{R}^M \rightarrow \mathbb{R}^1$, ($N$ and M are dimensions of raw data and intermediate representation, respectively, with $M < N$). During activation, the edge device sub–network produces an intermediate representation of raw data $\boldsymbol{x}$ as $\boldsymbol{z} = f_{\mathrm{E}}(\boldsymbol{x})$, sent to the server for prediction $\hat{y} = f_{\mathrm{S}}(\boldsymbol{z})$ (f_{S} is sub–network deployed at the server side). This training enables collaborative learning without compromising data privacy, achieved by iteratively exchanging model updates during backward passes between the server and edge device [12]. In the realm of split inference, split learning optimizes efficiency by employing pre–trained models distributed across multiple devices. Initial data processing occurs locally on edge devices, generating intermediate representations $\boldsymbol{z}$, which are then transmitted to a centralized server for aggregation and final inference [12].

In the context of rural IoT network which, except edge device and server, consist of relaying drone (UAV) that provides a backhaul link, neural network F is divided into three main parts, i.e., $F = (f_{\mathrm{E}}, f_{\mathrm{D}}, f_{\mathrm{S}})$, where $f_{\mathrm{D}} : \mathbb{R}^M \rightarrow \mathbb{R}^M$ represents drone sub–network.

C. Recurrent Neural Networks

RNNs, as sequence-based models, have the capability to discern temporal relationships between preceding and current states. Consequently, they represent an ideal solution for processing time series data [13]. Fig. 1 presents a simple depiction

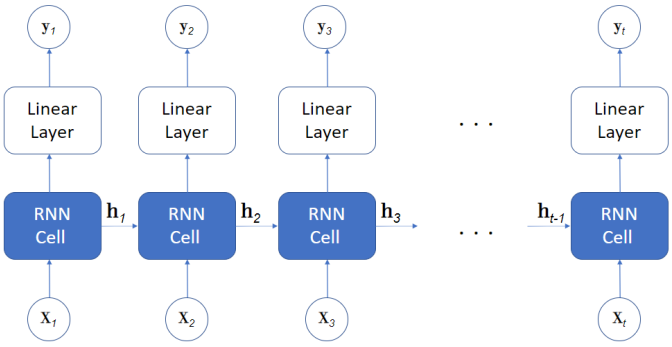

Fig. 1. The structure of Recurrent Neural Network.

of a single-layer RNN. In this illustration, the output from the previous time step, denoted as $t-1$, is incorporated into the input of the current time step, denoted as t, thus enabling the retention of past information. The computation outcome of a single RNN cell can be described by the following function:

$$\boldsymbol{h}_{\mathrm{t}} = \tanh(\boldsymbol{W}_{\mathrm{ih}}\boldsymbol{x}_{\mathrm{t}} + \boldsymbol{b}_{\mathrm{ih}} + \boldsymbol{W}_{\mathrm{hh}}\boldsymbol{h}_{\mathrm{t\text{-}1}} + \boldsymbol{b}_{\mathrm{hh}}), \tag{1}$$

where $\tanh$ denotes the hyperbolic tangent function, $\boldsymbol{h}_t$ and $\boldsymbol{h}_{t-1}$ represent the hidden states at time steps t and $t-1$, respectively, while $\boldsymbol{W}_{ih}$, $\boldsymbol{W}_{hh}$, $\boldsymbol{b}_{ih}$, and $\boldsymbol{b}_{hh}$ are the weights and biases requiring learning, with $\boldsymbol{x}_t$ denoting the input at time t.

Basic RNN cells encounter challenges in learning long-range dependencies primarily due to issues such as vanishing or exploding gradients. To address this limitation, Long Short-Term Memory (LSTM) cells were introduced, as proposed in [14]. These cells incorporate specialized units known as memory blocks within the recurrent hidden layer, thereby augmenting their ability to capture long-term dependencies. Each memory block constitutes a recurrently connected sub–network comprising functional components, namely memory cells and gates. The memory cells retain temporal states of the network, whereas the gates regulate the flow of information from the preceding cell state.

1) Split Learning–Based RNNs: Initially integrating the split learning/inference paradigm into LSTM neural networks faced challenges, prompting researchers to seek alternative methods. Some studies have advocated for using 1D-CNN instead of LSTMs to tackle these issues effectively [15]–[17]. Recent research has introduced efficient approaches to integrate split learning into LSTM networks [18]–[20], embedding the split learning paradigm directly into LSTM architectures to overcome implementation obstacles with innovative strategies.

This paper builds upon the foundational work of [19], which introduced the *LSTMSPLIT* algorithm, splitting the LSTM neural network vertically, and requiring a minimum of two LSTM layers, with the input sequence stored at the edge device. Following the procedure outlined in Section II-B, the intermediate representation $\boldsymbol{z}$ is transmitted from the edge device's LSTM layer to the server's LSTM layer, while update gradients move in the opposite direction (as

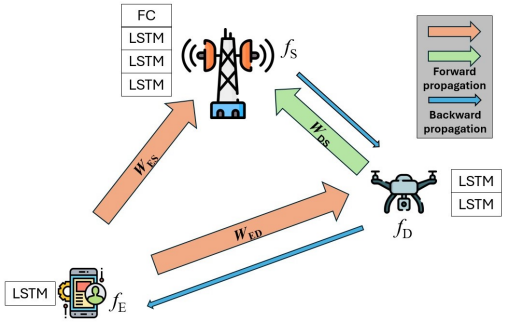

Fig. 2. SL–based fronthaul/backhaul communication with different channel conditions.

depicted on Fig. 2). In the second approach [20], the authors proposed a method where one LSTM layer is distributed across multiple edge devices, partitioned into sub-networks trained individually on each device. This enables the handling of segments within multi-segment training sequences. Communication among edge devices and parameter sharing facilitate inference, aligning with the federated learning paradigm.

III. UAV–ASSISTED RELAYING IN IOT

A. System Model

We examine the conventional IoT system, consisting of an edge device, server, and UAV, where the drone serves as a wireless relay, effectively functioning as a base station with backhaul link [9] (see Fig. 2). Each of these devices possesses unique computational capabilities, i.e, $\mathcal{C}(f_{\text{E}}) < \mathcal{C}(f_{\text{D}}) < \mathcal{C}(f_{\text{S}})$ (where $\mathcal{C}(\cdot)$ is sub–network complexity). At the edge device, we gather raw data denoted by $\boldsymbol{x}$, which may comprise sensor measurements, and these data points are labeled with corresponding labels y. Subsequently, this raw data undergoes pre–processing by the edge device sub–network, resulting in the intermediate representation $\boldsymbol{z} = f_{\text{E}}(\boldsymbol{x})$.

In this study, we establish assumptions concerning various levels of autonomy, primarily concentrated on the server side, which assumes responsibility for coordinating communication and inference processes. Specifically, taking into account channel conditions and essential performance metrics such as error rate, latency, and communication overhead, the server determines whether direct communication with the edge device is warranted or if the intermediate representation should undergo further processing by a drone sub-network. Furthermore, we anticipate significant variations in channel conditions, potentially differing between the edge device and drone (red arrow, $\mathcal{W}_{\text{ED}}$ on Fig. 2) as well as between the edge device and server (red arrow, $\mathcal{W}_{\text{ES}}$ on Fig. 2). Notably, we expect the channel between the drone and server (green arrow, $\mathcal{W}_{\text{DS}}$, on Fig. 2) to maintain consistently good quality throughout the entire network's lifespan.

Regarding the implemented strategy, the intermediate representation $\boldsymbol{z}$ encounters various channel conditions, resulting in its distorted version arriving at the server side. If direct communication occurs between the edge device and server, the intermediate representation at the server input can be defined as $\hat{\boldsymbol{z}} = \mathcal{W}_{\text{ES}}(\boldsymbol{z})$. Conversely, if communication between the edge device and server traverses a drone backhaul link, and the drone sub–network is involved in overall processing and prediction, then $\hat{\boldsymbol{z}} = \mathcal{W}_{\text{DS}}(f_{\text{D}}(\mathcal{W}_{\text{ED}}(\boldsymbol{z})))$. On the server side, local decisions are made regarding desirable performance criteria, particularly latency constraints. For example, the server determines the optimal strategy, deciding whether the entire server sub–network will be included in the prediction process or only its output layer. More precisely, the estimation of y can be defined as $\hat{y}_{full} = f_{\text{S}}(\hat{\boldsymbol{z}})$ or $\hat{y}_{FC} = \hat{f}_{\text{S}}(\hat{\boldsymbol{z}})$ if the server sub-network comprises all layers or only its output layer, respectively.

According to obtained estimation, appropriate loss function, which consists of two different parameters is calculated at server side as:

$$\mathcal{L}(y, \hat{y}_{\text{full}}, \hat{y}_{\text{FC}}) = MSE(y, \hat{y}_{\text{full}}) + MSE(y, \hat{y}_{\text{FC}}), \quad (2)$$

where $MSE(y, \hat{y}) = 1/|\mathcal{D}_{\text{tr}}| \sum_{\mathcal{D}_{\text{tr}}} (y - \hat{y})^2$ is mean–squared error (MSE), $\mathcal{D}$ is training dataset and $|\cdot|$ is its cardinality.

In the backward propagation phase, gradients are determined with respect to the loss function and then conveyed from the server, passing through a drone, and finally directed towards the edge device, essentially reversing the neural network's direction (as illustrated by the blue arrows in Fig. 2). In such a case, all three sub–networks (f_{E}, f_{D}, f_{S}) are jointly optimized. The collaborative optimization typically involves fine-tuning the parameters of all three sub–networks using optimization algorithms like stochastic gradient descent (SGD) or its adaptations, such as Adam [21].

Under the above–mentioned setup, the main goal here is to define the most suitable transmission strategy, regarding the channel conditions and desirable latency, for minimization overall system error, defined as MSE error from Eq. (2), between y and $\hat{y}$ across all test examples.

B. Channel Model

We consider relatively simple wireless communication link between edge device and server ($\mathcal{W}_{\text{ES}}$), edge device and drone ($\mathcal{W}_{\text{ED}}$) and drone and server ($\mathcal{W}_{\text{DS}}$). These links are modeled as conventional erasure channels, with an erasure probability denoted by p. This channel can be represented as a binary vector $\boldsymbol{q} \in \{0, 1\}^M$, where M is the length of the intermediate representation $\boldsymbol{z}$ (as discussed in Section II-B). Individual symbols from $\boldsymbol{z}$ are either erased or they arrive unchanged at the server side. Consequently, $\hat{\boldsymbol{z}} = \boldsymbol{z} \odot \boldsymbol{q}$, where $\odot$ represents element–wise multiplication [22].

IV. PERFORMANCE EVALUATION

A. Training Setup

To evaluate the proposed approach and assess the influence of different channel conditions on overall system performance, as well as the significance of the backhaul, we utilized a dataset perfectly suited to the environmental problem of interest. Specifically, we focus on monitoring pollution in the Danube river near Novi Sad. Our dataset comprises 3,264

instances, with 70% utilized for training and the remaining 30% used for testing purposes. Each instance represents a daily measurement from November 2013 to October 2022, encompassing eight different water quality parameters: temperature, pH value, electrical conductivity, dissolved oxygen, oxygen saturation, ammonium, and nitrite.

Based on the correlation matrix between all measured features, we have decided to predict dissolved oxygen using its last 20 measurements (over the previous 20 days) along with measurements of the other 7 parameters for the current day. More precisely, following data preprocessing and conversion to time series, each instance in the dataset comprises 27 features (including 20 previous dissolved oxygen measurements and 7 other parameters) and one label (representing dissolved oxygen for the current day). Additionally, considering that different parameters are measured on varied scales, we normalize the data to fall within the range of -1 to 1.

Training procedure follows conventional SL, introduced in [12], with slight adjustments to fit the scenario of interest. In more detail, after raw data $\boldsymbol{x}$ is collected, it undergoes pre–processing on the edge device, and an intermediate representation is then sent either directly to the server (via $\mathcal{W}_{\text{ES}}$) or across the backhaul (using a drone, through both $\mathcal{W}_{\text{ED}}$ and $\mathcal{W}_{\text{DS}}$). If the backhaul is utilized, the drone introduces additional processing of the intermediate representation, as depicted in Fig. 2. The neural network F is divided into sub-networks across the edge device (comprising one LSTM layer), the drone (comprising two LSTM layers), and the server side (comprising 3 LSTM layers followed by a fully connected (FC) layer) as depicted in Fig. 2. The number of LSTM hidden units, denoted as H, remains fixed for all conducted experiments, and it is set equal to the length of the intermediate representation M, i.e., $H = M = 10$. The additional fully connected (FC) layer at the server side also consists of 10 neurons. Training is conducted using a learning rate of $\alpha = 0.01$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ on a batch-by-batch basis, with a batch size of 64. We utilize stochastic gradient descent (SGD) with the Adam optimizer, as detailed in Section III-A.

Channel conditions remain fixed during the training phase, following the approach proposed by [23]. However, during testing, we assess the model's performance across a range of erasure probabilities p. It is important to emphasize that we investigate the impact of different channel conditions introduced during the training phase by varying the training erasure probabilities, p_{tr}, in $\mathcal{W}_{\text{ES}}$ and $\mathcal{W}_{\text{ED}}$, while ensuring that the backhaul between the drone and the server $\mathcal{W}_{\text{DS}}$ always maintains good channel conditions with a small erasure probability (Section III-A). The erasure channel conditions are simulated by incorporating additional dropout layers [24] during the training process. These dropout layers replace all three wireless links in Fig. 2, akin to the approach outlined in [22], albeit on a symbol basis. We specify a particular dropout probability to regulate the occurrence of channel erasures within our simulations.

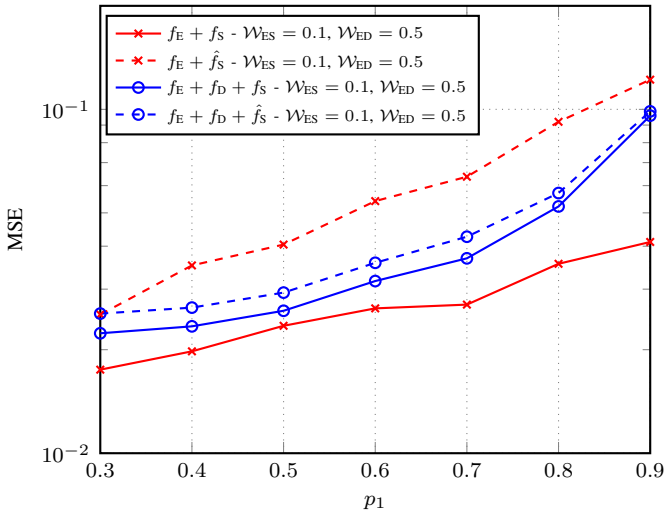

Fig. 3. MSE versus p_1 erasure probabilities: Improved edge-to-server channel conditions (p_{tr} for $\mathcal{W}_{\text{ED}} > \mathcal{W}_{\text{ES}}$) with $\mathcal{W}_{\text{DS}} = 0.05$.

B. Numerical Results

To examine the behavior of the proposed system under diverse conditions, we consistently assume that one of the channels, either $\mathcal{W}_{\text{ES}}$ or $\mathcal{W}_{\text{ED}}$, introduces significant distortions in the intermediate representation.Additionally, during the testing phase, we set the erasure probability for the more distorted channel to p_1, and for the less distorted channel to p_2=$p_1 - 0.3$. Meanwhile, $\mathcal{W}_{\text{DS}}$ remains constant, with its erasure probability set to 0.05 during both the training and testing phases.

In Fig. 3, we compare the MSE performances of the fronthaul and backhaul systems, where the backhaul ($\mathcal{W}_{\text{ED}}$) significantly alters the intermediate representation due to a high erasure probability. More precisely, training erasure probability for $\mathcal{W}_{\text{ED}}$ is set to 0.5, while for $\mathcal{W}_{\text{ES}}$ to 0.1. Consequently, during the testing phase, erasure probability p_1 is associated with $\mathcal{W}_{\text{ED}}$, while the fronthaul link $\mathcal{W}_{\text{ES}}$ is tested with p_2. It becomes evident that channel conditions play a significant role in overall system performance. For instance, the fronthaul system, characterized by full server processing under favorable channel conditions (indicated by the red solid line in Fig. 3), surprisingly demonstrates superior performance despite its lower processing complexity. It is also notable that, within the backhaul system, additional processing does not lead to the recovery of lost symbols. This is evident from the performances achieved by deploying the full server sub–network (solid blue line in Fig. 3), which are similar to those obtained when only the output FC layer is utilized (dashed blue line in Fig. 3).

Examining Fig. 4, we observe that when subjecting our approach to testing conditions resembling real-world scenarios—where the fronthaul link between the edge device and server is highly corrupted, necessitating fallback solutions like the backhaul—we find that additional processing successfully extracts all temporal connections within the time series data. Consequently, the system that integrates all three sub-networks

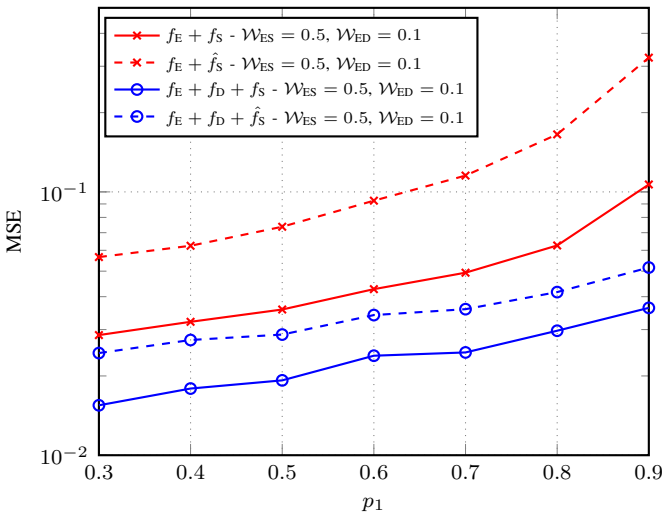

Fig. 4. MSE versus p_1 erasure probabilities: Improved edge-to-drone channel conditions (p_{tr} for $\mathcal{W}_{ED} < \mathcal{W}_{ES}$) with $\mathcal{W}_{DS} = 0.05$.

demonstrates superior performance, as indicated by the blue solid line in Fig. 4.

The results depicted in Figs. 3 and 4 provide compelling evidence that the proposed approach, which integrates backhaul transmission and SL paradigm, offers significant robustness to the server. This robustness enables the server to adapt to unexpected changes in wireless links and define an optimal transmission strategy within various operating conditions. Moreover, it also provides a significant degree of freedom to the server. With this flexibility, the server can incorporate various trade-offs into the implemented strategy, beyond just considering MSE. For instance, it can now factor in parameters such as latency or communication overhead, allowing for a more nuanced and adaptive approach.

V. Conclusion

In this work, we present a novel approach that offers a single, versatile framework for the integration of distributed learning and UAV-assisted relaying in IoT environmental monitoring systems. The proposed architecture demonstrates significant adaptability to varying channel conditions in IoT systems, offering different trade-offs that can be defined by the server, primarily based on desired performance metrics. Furthermore, the introduction of the SL paradigm optimally balances the computational load between each component (edge device, UAV, and server). In future work, our goal is to incorporate additional parameters into the server decision-making process, such as latency and energy efficiency.

References

[1] S. L. Ullo and G. Sinha, "Advances in smart environment monitoring systems using iot and sensors," *Sensors,* vol. 20, no. 11, pp. 3113, May 2020.

[2] G. Mois, S. Folea, and T. Sanislav, "Analysis of three IoT-based wireless sensors for environmental monitoring," *IEEE Trans. Instrum. Meas.,* vol. 66, no. 8, pp. 20562064, Aug. 2017.

[3] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tuts.,* vol. 21, no. 3, pp. 2334–2360, 3rd Quart., 2019.

[4] J. Sabzehali, V. K. Shah, Q. Fan, B. Choudhury, L. Liu, and J. H. Reed, "Optimizing number, placement, and backhaul connectiv ity of multi-UAV networks," *IEEE Internet Things J.,* vol. 9, no. 21, pp. 21548–21560, Nov. 2022.

[5] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: opportunities and challenges," *IEEE Commun. Mag.,* vol. 54, no. 5, pp. 36–42, May 2016.

[6] B. Galkin, J. Kibilda, and L. A. DaSilva, "Backhaul for low-altitude UAVs in urban environments," in *Proc. IEEE Int. Conf. Commun. (ICC),* 2018, pp. 1–6.

[7] E. Yaacoub and M.-S. Alouini, "Efficient fronthaul and backhaul connectivity for IoT traffic in rural areas", *IEEE Internet Things Mag.,* vol. 4, no. 1, pp. 60-66, Mar. 2021.

[8] L. Zhang and N. Ansari, "Optimizing the deployment and throughput of dbss for uplink communications," *IEEE Open J. Veh. Tech.,* vol. 1, pp. 18–28, 2019.

[9] A. Fouda, A. S. Ibrahim, I. Guvenc, and M. Ghosh, "UAV-based in-band integrated access and backhaul for 5G communications," in *Proc. IEEE Conf. Veh. Tech.,* 2018, pp. 1–5.

[10] M. Y. Selim and A. E. Kamal, "Post-disaster 4G/5G network rehabil itation using drones: Solving battery and backhaul issues," in *Proc. IEEE Globecom Workshops (GC Wkshps),* 2018, pp. 1–6.

[11] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *J. Netw. Comput. Appl.,* vol. 116, pp. 1–8, Aug. 2018.

[12] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018, arXiv:1812.00564. [Online]. Avaliable: https://doi.org/10.48550/arXiv.1812.00564

[13] M. Hüsken and P. Stagge, "Recurrent neural networks for time series classification," *Neurocomputing*, vol. 50, pp. 223–235, Jan. 2003.

[14] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, pp. 1735–1780, 1997.

[15] S. Abuadbba, K. Kim, M. Kim, C. Thapa, S. A. Camtepe, Y. Gao, H. Kim, and S. Nepal, "Can we use split learning on 1d cnn models for privacy preserving training?" in *Proc. 15th ACM Asia, ser. ASIA CCS '20.* New York, NY, USA: Association for Computing Machinery, pp. 305–318, 2020.

[16] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtep, H. Kim, and S. Nepal, "End-to-end evaluation of federated learning and split learning for internet of things," in *Proc. IEEE 2020 Int. Symp. on Rel. Distrib. Syst.,* Shanghai, China, Sep. 21-24, pp. 91–100., 2020.

[17] W. Zhang, T. Zhou, Q. Lu, Y. Yuan, A. Tolba, and W. Said, "FedSL: A Communication Efficient Federated Learning With Split Layer Aggregation," *IEEE Internet Things J.*, Early Access

[18] Y. Koda, J. Park, M. Bennis, K. Yamamoto, T. Nishio, M. Morikura, and K. Nakashima, "Communication-efficient multimodal split learning for mmWave received power prediction," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1284–1288, June 2020.

[19] L. Jiang, Y. Wang, W. Zheng, C. Jin, Z. Li, and G. S. Teo, "LSTMSPLIT: effective SPLIT learning based LSTM on sequential time-series data," 2022, arXiv: cs.LG/2203.04305. [Online]. Available: https://doi.org/10.48550/arXiv.2203.04305

[20] A. Abedi and S. S. Khan, "Fedsl: Federated split learning on distributed sequential data in recurrent neural networks," *Multimed. Tools. Appl.*, vol. 83, pp. 28891–28911, Sept. 2023.

[21] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. on Learn. Representation,* May 7–9, pp. 1-41, 2015.

[22] S. Itahara, T. Nishio, Y. Koda, and K. Yamamoto, "Communication-Oriented Model Fine-Tuning for Packet-Loss Resilient Distributed Inference Under Highly Lossy IoT Networkss," *IEEE Access*, vol.10, pp. 14969–14979, 2022.

[23] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.,* vol. 3, no. 4, pp. 563-575, Dec. 2017.

[24] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R.Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, arXiv: cs.NE/arXiv:1207.0580. [Online]. Available: https://arxiv.org/abs/1207.0580