# Temperature Forecasting Using Federated Learning on Edge Devices

Aditya Khatavkar[1], Nrupesh Kamble[2], Devraj Shirsath[3], Shriman Ghugare[4], Shubham Shende[5], Prof. N. R. Shetty[6]
[1,2,3,4,5,6]*Department of IT GCE Karad, India*

***Abstract*—It is challenging to build a system that can provide accurate predictions in terms of temperatures using conventional machine learning techniques since they rely on huge computational capabilities. This paper presents an innovative FL-based framework for temperature prediction directly from the edge devices in Android smartphones. Contrary to the use of resource-intensive recurrent neural networks, the suggested method utilizes a small Multi-Layer Perceptron (MLP) in combination with sliding window temporal feature extraction to identify short-range weather patterns. Each node collects past weather data from the Open-Meteo API, pre-processes and normalizes the data locally, and trains the MLP locally as well. Only the trained weights of the model are transferred to a Python-based server that aggregates the models using the Federated Averaging (FedAvg) technique to generate the global model. The system was evaluated on three mobile phones placed in three different locations in the state of Maharashtra, namely Karad, Satara, and Umbraj. The simulation outcomes indicate steady learning behavior, obtaining a MAE value of 0.245°C and MSE value of 0.1035 for a 24-hour forecast horizon.**

***Index Terms*—Edge computing, federated averaging, federated learning, multi-layer perceptron, temperature forecasting.***

## I. INTRODUCTION

Weather predictions are important for an extensive list of applications, such as planning agriculture and disaster preparedness to name but a few. Although there have been huge advancements in Numerical Weather Prediction as well as deep learning, the implementation of high-resolution forecasting is hampered due to the expensive computation required during central model training [15],[16].

This paper seeks to fill these gaps with the development of an end-to-end temperature prediction framework built using federated learning technology which runs on edge devices. This temperature prediction framework does not make use of heavy and bulky recurrent neural network architectures but uses an efficient Multi-Layer Perceptron (MLP) model along with time-series feature engineering techniques.

This work makes four major contributions to literature which can be summarized as follows:

1. End-to-End Edge Federated Learning Pipeline:
An end-to-end federated learning pipeline where the Android devices conduct local learning while the global aggregation happens on a server side written in Python.

2. Time-Series-Aware Feature Engineering:
Sliding window with sine-cosine encoding of the temporal features allows the model to capture the daily cycles of temperature values without using expensive recurrent layers.

3. Edge Optimized MLP Model:
Efficient implementation of a simple MLP model with z-score normalization and adam optimization algorithm suitable for running on consumer smartphones.

4. Edge Validation of the Model:
Deployment and validation of the developed federated learning temperature forecast model across three regions of Maharashtra (Karad, Satara, and Umbraj).

The rest of the paper is organized as follows: Section II provides literature review. Section III gives the system architecture description. Section IV discusses the methodology proposed. Section V describes experimental results. Section VI provides practical considerations, and Section VII draws the conclusion of the paper.

## II. LITERATURE SURVEY

### *A. Foundational Federated Learning and Data Heterogeneity*

Federated Averaging (FedAvg) method, initially published in 2017 by McMahan et al. [2], is widely considered to be one of the first successful attempts at federated learning algorithm design. Unlike traditional algorithms where training data are collected in the central hub, FedAvg permits each client to train a model locally and submit only model updates to the server. The weighted aggregation is applied on the basis of sizes of client's datasets. Due to its effectiveness and inherent privacy-preserving characteristics, FedAvg became a benchmark algorithm for many FL methods, including the one designed in this study.

Multi-Objective and Non-IID Problems: In their work on federated learning of multi-task problems, Skovajsova et al. [3] pointed out difficulties of reconciling conflicting objectives during training in case of significant data heterogeneity across clients' datasets. Their paper revealed an inherent flaw in FedAvg algorithm – its inefficiency in case of non-IID data.

Effect of Data Heterogeneity in Regression Problem: Nachtigall et al. [11] researched effects of data heterogeneity on federated learning regression model, especially related to solar radiation prediction. Their results convincingly support further research in developing robust aggregation techniques.

In order to tackle the problem of class imbalance within datasets between various clients, Alluhaidan et al. [4] conducted a study on the comparison between various aggregation schemes within federated learning. The study involved comparing the FedAvg approach against others like FedProx and custom weights schemes. The results show that the selection of an aggregation scheme plays a significant role in determining the convergence rate and overall model accuracy for federated learning systems with class imbalances. Based on the lessons learned from their study, we incorporate a weighted FedAvg scheme based on sample size within our framework.

### *B. Federated Learning for Weather and Environment*

Weather Prediction Models through Crowdsourcing: De Vita et al. [8] have evaluated the feasibility of implementing a federated learning approach towards decentralized environmental monitoring. Through their investigation, the authors concluded that sensor networks have the potential to collectively train models for weather prediction while preserving the confidentiality of telemetry data. Even though the authors were able to prove that weather modeling through federated learning is feasible, their approach involved only simple regression models that were deployed in simulation environments. Therefore, there was no evaluation of neural models implemented on real mobile devices.

UAV-Based Federated Learning: Singhal et al. [5] proposed an FL-based system for predicting weather in real-time with the help of unmanned aerial vehicles. Although the authors successfully managed to implement a system for real-time weather predictions, which could adapt to changing atmosphere conditions, they needed drones to gather data. On the other hand, Konda et al. [10] introduced a FedWAvg mechanism to counter attacks by malicious users and reduce the effect of deviant client update contributions.

Cellular Network and Edge Constraints: The concept of FedLSTM, for collaborative weather prediction using cellular network, was proposed by Joel et al. [6]. Although the LSTM network works well in sequential data prediction, its computationally heavy nature makes it impractical to run on general smartphones. In order to overcome this constraint, the proposed framework will adopt an MLP model instead of the LSTM for edge processing.

Minimizing Latency: Latency issues exist while conducting federated learning tasks. This problem can be alleviated with the help of the protocol introduced by Su et al. [9], known as DAFL. With this protocol, delay can be reduced by making use of device-to-device communication. Furthermore, similar efforts have been made by R. Chen et al. [17] in order to minimize service delay within mobile federated networks.

### *C. Machine Learning Architectures for Temperature Forecasting*

In addition, Madala et al. [15] investigated hybrid models combining neural network architectures with gradient boosting and decision trees. The findings of this study suggest that simpler models can generate accurate predictions even if they rely on proper temporal feature engineering, which justifies the choice of an MLP in our research. Moreover, Lei et al.

[16] state that ensemble-based models tend to be superior to stand-alone algorithms, which is consistent with the idea behind FedAvg.

Pre-trained/Foundation Models: Finally, some studies employ pre-trained/foundation models for meteorological forecasting purposes. Thus, S. Chen et al. [12] explored the usage of prompt-tuning to customize foundation models based on meteorological data. Furthermore, Jin et al. [14] designed a federated learning algorithm employing transformers augmented with Kolmogorov–Arnold Networks. However, despite achieving high accuracies, such models have a huge computational footprint that disqualifies them from deployment on edge devices.

On-Device Time-Series Processing: Finally, Takanashi and Shinomiya [1] demonstrated that federated learning is capable of processing time series on wearable devices for performance prediction. Although the study relies on physiological data instead of meteorological variables, it proves the efficiency of on-device training in scenarios comparable to temperature forecasting tasks.

*D. Security and Robustness in Decentralized Systems*

Security vulnerability:

Distributed learning platforms pose various security risks. According to Tyagi et al., some vulnerabilities include threats like membership attacks, Byzantine failure, and gradient leaking which could lead to exposure of sensitive data.

Network Security:

In their work, Ojha et al. [13] discussed the use of federated learning platforms in improving network security. The research indicated how collaborative learning can assist in detecting abnormality in distributed networks without the need for central access to sensitive logs.

Privacy and Robustness Improvement:

Bakir et al. presented approaches that can improve privacy and robustness to adversarial participants in decentralized AI platforms.

From the above literature, our work seeks to fill three gaps: the scarcity of on-device federated learning in consumer Android smartphones, the dependence on complex architectures, and difficulties of incorporating machine learning into networking processes.

## III. SYSTEM ARCHITECTURE

There are three primary building blocks in the architecture: the edge-side learning engine, the client communication engine, and the central aggregation server.

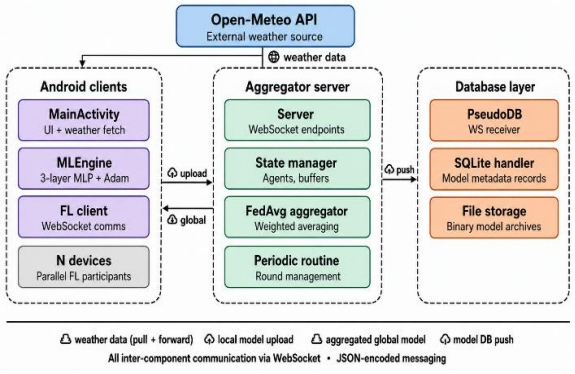


Fig 1. System Architecture

*A. Edge-Side Machine Learning Engine*

The local learning engine runs on the mobile phone and preprocesses the collected data, generates the features, and trains the model locally. Because the data stays within the device, the privacy of location information is maintained [7].

*B. Client Communication Module*

The communication module of the Android client serves as the federated learning agent. On initiation, the module creates an Agent ID, accesses local network settings, and creates separate directories for storing local and global models. The module manages the execution of a highly controlled state machine that consists of the following stages:

1. Registration stage: Client registers at the aggregator using the Agent ID and model state.
2. Polling and synchronization stage: Client polls the server for updates on the global model through lightweight socket messages, thus minimizing the cost of sending regular HTTP requests.
3. Local model uploading stage: After completing local epoch training, the client uploads only the updated weights of the neural network, total training sample size, and local metrics (MAE and MSE).

*C. Central Aggregation Server*

Central aggregator, which has been programmed as a Python server, combines all local models into one global model based on weights updated by client

nodes. It runs the Federated Averaging technique dynamically [2]. Instead of making assumptions about the network topology, the aggregator aggregates the received weight tensors layer by layer. Additionally, the server constantly analyzes tensor sizes, L2 norm, mean value, and standard deviation to identify any possible mathematical abnormalities.

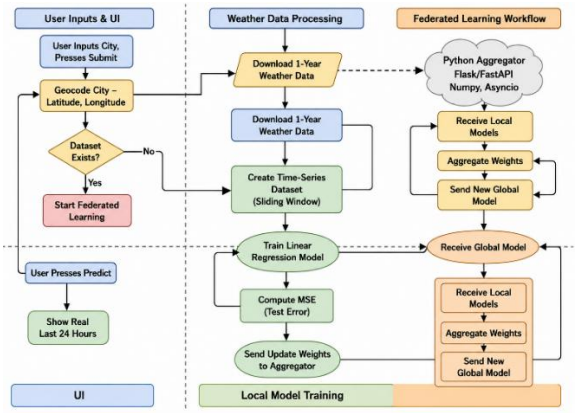


Fig 2. Schematic Workflow of System

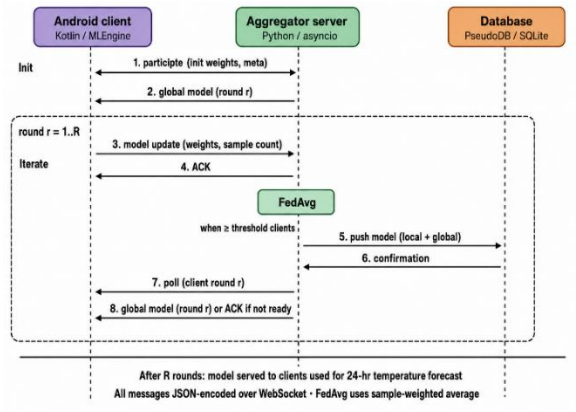


Fig 3. Complete FL communication protocol as a sequence diagram.

## IV. METHODOLOGY

The following segment describes the approach taken towards data representation, mathematical feature engineering, neural network model, and the particular federated learning pipeline employed on the edge devices.

### *A. Data Acquisition and Preprocessing*

Historical weather data is individually gathered by each edge device according to the respective geographical location of each client node. For each client, the local dataset includes 8,784 consecutive hourly weather measurements which correspond to one whole year of weather data. To avoid the issue of over-fitting, the dataset is carefully divided into an 80/20 chronologically structured division of training and testing set consisting of 6,989 and 1,748 entries respectively.

In order to speed up the gradient descent process and avoid domination of larger numeric range features in the objective function, the data is scaled via Z-score normalization technique:

$$z = \frac{x - \mu}{\sigma}$$

where $\mu$ is the mean and $\sigma$ is the standard deviation. In order to completely avoid data leakage, the statistical measures are calculated only on the training set and then used for normalization on the test set.

### *B. Temporal Feature Engineering*

The conversion from time-series to supervised learning is accomplished by applying a sliding window approach. The model makes predictions based on a historical look-back period of 24 hours for predicting the next 24 hours.

Instead of using computationally expensive recurrent layers (LSTM), a cyclic feature generation approach is employed [6]. The input vector for each hour in the 24-hour sliding window contains five features: temperature normalization, humidity normalization, solar radiation normalization, and sine and cosine encoding for the hour of the day. The sine-cosine encoding is essential because it provides a mathematical representation of the cyclical nature of the day's temperature pattern without losing continuity at 23:00 and 00:00 [15].

Further, five high-level context features are added to the flat sliding window vector to account for climate change: sine and cosine of the day of the year, temperature trend in the past 24 hours, temperature trend in the past 6 hours, and average temperature in the last 6 hours.

Cyclic hour encoding used to encode hour of the day so that time remains continuous.

$\sin\left(\frac{2\pi h}{24}\right)$ , $\cos\left(\frac{2\pi h}{24}\right)$ ; h = hour of the day (0–23)

Thus, the total dimension of the input for the neural network is formally calculated as follows:

*InputSize = (WindowSize × InputFeatures) + ExtraFeatures*

With a window size of 24, 5 features per hour, and 5 context features, the final dimensionality of the input vector is set to 125.

*C. Local Model Architecture*

Given the limitations of edge devices regarding RAM and computational power, the prediction is done using a light MLP structure.

The algorithm does not employ heavy computations like convolutions or RNNs, and, thus, is suitable for implementation on edge devices that take part in federated learning.

Input Features

Input features are represented as 125 dimensions based on 24 hours sliding window containing weather data and other features representing time.

Input Representation:

The input layer comprises 125 features which include normalized temperature values, humidity, solar radiation, and time encoded through sine-cosine transformation of hour of day and day of the year.

Hidden Layer 1 (dense1):

In the first complete layer, there are 128 neurons, with the ReLU function being used as the activation function. This layer helps identify non-linear correlations between past meteorological features and future temperature actions without losing computational effectiveness. Forward pass equation for first hidden layer

$h_1 = ReLU(W_1 x + b_1)$

Hidden Layer 2 (dense2):

Second Hidden Layer

Second hidden layer comprises 64 neurons, again employing ReLU activation function. Second hidden layer refines feature extraction and compression before making a final decision on input features. Formula for Forward pass in second hidden layer

$h_2 = ReLU(W_2 h_1 + b_2)$

Output Layer:

The last thick layer comprises 24 neurons, each employing linear activation to match the 24-hour prediction horizon for temperature change. Each neuron makes a prediction for a particular hour. Formula for forward propagation of the output layer is:

$\Delta T = W_3 h_2 + b_3$

Where:

- $x$= input vector (125 features)
- $h_1$= first hidden representation
- $h_2$= second hidden representation
- $\Delta T$= predicted temperature deltas

While training, the network does not attempt to predict the absolute temperature but rather the temperature difference, which is a measure of temperature change compared to the previous hour within the input window. The benefit of predicting the residuals is increased numerical stability in backpropagation and lower prediction variance.

At the time of prediction, the predicted deltas will be summed with the base temperature value, yielding the actual temperature prediction. The predictions are limited by physically possible boundaries (−30 °C to 60 °C).

Delta temperature target

$\Delta T_i = T_{t+i} - T_t$

Where

$T_t$= baseline temperature.

Baseline Temperature

$T_{base} = T_t$

Final temperature prediction

$T_{pred}(t+i) = T_{base} + \Delta T_i$

*D. D. Federated Optimization and Training Pipeline*

Optimization of localized training within the edge device through the use of Adaptive Moment Estimation (Adam) method that modifies the learning rate depending on the mean and variance of gradient moments. Equation for Adam optimization is stated as:

$$\theta_{t+1} = \theta_t - \frac{\alpha m_t}{\sqrt{v_t} + \epsilon}$$

Where;

- $\alpha$ learning rate
- $m_t$ : first moment estimate
- $v_t$ : second moment estimate.

Gradient clipping is applied throughout the backpropagation step to avoid exploding gradients, which is a common problem in mobile-based learning. Based on the experimental setup, the local training process takes place for $E = 3$ epochs for each federated iteration to maintain a good trade-off between computation cost and learning.

After the completion of the local training, the updated weight values $W_i$ and the amount of data $n_i$ are uploaded to the aggregation server. The central server combines the new global model $W_{global}$ using the weighted FedAvg algorithm [2]:

$$W_{global} = \sum_{i=1}^{N} \frac{n_i}{\sum_{j=1}^{N} n_j} W_i$$

where N is the total number of clients contributing their edge devices. In this way, clients with larger or more diverse datasets are assigned adequate weights in relation to the global model. After aggregation, the resultant global model is sent to the client's end and the process begins anew.

## V. EXPERIMENTAL RESULTS

In order to verify the federated learning architecture, the framework was executed in a real-life edge setting, not a simulated one. The following paragraphs present the setup of the hardware used, the metrics, and the results obtained from the experiments conducted.

### *A. Experimental Setup*

The federated network is created via a centralized Python aggregator running on a local network, using three different Android devices as federated learning clients. In order to mimic the geographical dispersion, each client received a separate dataset related to a particular geographical location within Maharashtra State, India.

- Client 1: Karad – Lat 17.68589, Long 73.99333
- Client 2: Satara – Lat 17.28937, Long 74.18183
- Client 3: Umbraj – Lat 17.40277, Long 74.10096

Hardware information for all three edge devices is presented in Table I below. The variation in processors and Random Access Memory (RAM) creates a practical model of a heterogeneous commodity smartphone network.

TABLE I. DEVICE HARDWARE AND COMPUTATIONAL PERFORMANCE

| Device Model | Role | Processor (SoC) | RAM | Avg. Round Time |
|---|---|---|---|---|
| Vivo Y21 | FL Client 1 | MediaTek Helio P35 | 4 GB | ~20.5 sec |
| Redmi Note 10 Pro | FL Client 2 | Snapdragon 732G | 6 GB | ~19.0 sec |
| Vivo Y50 | FL Client 3 | Snapdragon 665 | 8 GB | ~18.0 sec |

### *B. Evaluation Metrics*

Accuracy measures for both local and global models were calculated using two regression measures:

1. MAE: Measures the average magnitude of the absolute errors in the predictions, providing a clear indication of expected temperature deviation in degrees Celsius.
2. MSE: Heavily penalizes larger forecasting errors by squaring the residuals, serving as a strict indicator of model stability.

### *C. Empirical Results and Discussion*

Table II provides an overview of the training setup and the prediction performance for each client based on the aggregation of the global model.

TABLE II FEDERATED TRAINING ACCURACY METRICS ACROSS LOCATIONS

| Location | Train Samples | Test Samples | Epochs / Round | MAE (°C) | MSE |
|---|---|---|---|---|---|
| Karad | 6,989 | 1,748 | 3 | 0.245 | 0.1035 |
| Satara | 6,989 | 1,748 | 3 | 0.245 | 0.1035 |
| Umbraj | 6,989 | 1,748 | 3 | 0.245 | 0.1035 |

From Table II, it is clear that the federated framework provided a very accurate forecast, with an MAE of 0.245°C. Interestingly, all the metrics are identical for all three geographic locations. If this were the case in centralized training, this would suggest a high level of

redundancy in the data used. However, given the nature of this federated system, this is the ideal result since all three clients share the same architecture. Since all three clients use the same dataset size (8,784 rows), identical 80/20 split ratios, identical sliding window feature engineering, and identical global model initialization, they should converge to the same result after the FedAvg aggregation [2].

While maintaining the same degree of predictive accuracy, Table I provides information concerning the effect of hardware variability on computation time. The high-end device, the Vivo Y50, with its 8 GB RAM and Snapdragon 665 CPU, took about 18.0 seconds to finish the local epochs and networking tasks. On the other hand, the low-end phone, the Vivo Y21, had a computation time of 20.5 seconds per cycle. In spite of the differences in the hardware used by the two devices, WebSocket was able to manage the processes without any race conditions or server overload problems.

## VI. CONCLUSION AND FUTURE WORK

This paper presented an entirely functional federated learning platform that was specifically built to carry out accurate temperature prediction at high resolution through an Android smartphone. This was made possible by introducing a more lightweight Multi-Layer Perceptron (MLP) model and adding sliding window sine-cosine temporal features to effectively execute complex time-series predictions in the constrained memory and processing capacity of regular consumer smartphones.

The experimental application across three distinct smartphones (Vivo Y21, Redmi Note 10 Pro, and Vivo Y50) resulted in consistently reliable convergence. The model produced a Mean Absolute Error (MAE) of 0.245°C in a 24-hour forecast period, demonstrating the feasibility of conducting accurate meteorological predictions while avoiding data sharing with a central server. In addition, the architecture was able to overcome any software engineering obstacles associated with implementing the federated learning algorithm, specifically by decoupling gradient descent calculations from the user interface.

No systematic evaluation of the performance of the algorithm in farther-away places such as Pune, Nagpur, and coastal parts like the Konkan region has been undertaken. We noticed that the better the logic of the machine learning, the better the results and better results will be obtained through the use of better models. All WebSocket communication is plain text (ws://). It means all the model weights, metadata and IP addresses of the devices are not encrypted during transfer. There is also a lack of client authentication which implies that anyone who knows the aggregator's IP address and port can connect to the federated learning network and may inject poisonous model updates.

Future versions of this framework are expected to concentrate on improving communication processes. More specifically, introducing device-to-device communication [9] and resource allocation techniques [17] will allow for reducing federated rounds. Besides, since the number of devices in the federated learning process grows, employing differential privacy and secure aggregation methods becomes important to protect the network from possible attacks [7], [10], [18]. In conclusion, this work demonstrates the vast possibilities of using consumer devices as a privacy-first environmental sensor network.

## REFERENCES

[1] K. Takanashi and N. Shinomiya, “Enhancing Running Performance: A Federated Learning Approach with Wearable Devices,” in *2024 IEEE 13th Global Conference on Consumer Electronics (GCCE)*, Tokyo, Japan, 2024, doi: 10.1109/GCCE62371.2024.10760398.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[3] L. Skovajsova, L. Hluchý, and M. Staňo, “A Review of Multi-Objective and Multi-Task Federated Learning Approaches,” in *2025 IEEE 23rd World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Stará Lesná, Slovakia, 2025, doi: 10.1109/SAMI63904.2025.10883172.

[4] T. Alluhaidan and D. Josyula, “Weight Aggregation Methods for Federated Learning in Healthcare - A Comparative Empirical Analysis,” in *2024 IEEE International Conference on*

*Electro Information Technology (eIT)*, 2024, doi: 10.1109/eIT60633.2024.10609860.

[5] P. Singhal and P. Joshi, "Real-Time Incremental Federated Learning for Weather Prediction with UAV Based Data Collection," in *2024 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2024, doi: 10.1109/ICVES61986.2024.10927919.

[6] O. K. Joel, S. Liu, Q. Li, and X. Ge, "FedLSTM Based Cooperative Weather Prediction in RAN," in *2024 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 2024, doi: 10.1109/ICCCWORKSHOPS62562.2024.10693728.

[7] S. Tyagi, I. S. Rajput, and R. Pandey, "Federated Learning: Applications, Security Hazards and Defense Measures," in *2023 International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)*, 2023, doi: 10.1109/DICCT56244.2023.10110075.

[8] C. G. De Vita, G. Mellone, A. Casolaro, M. G. Orsini, J. L. Gonzalez-Compean, and A. Ciaramella, "Federated Learning and Crowdsourced Weather Data: Practice and Experience," in *2024 IEEE 20th International Conference on e-Science (e-Science)*, 2024.

[9] H. Su, P. Prakash, R. Chen, Y. Gong, R. Yu, X. Fu, and M. Pan, "DAFL: Delay Efficient Federated Learning over Mobile Devices via Device-to-Device Transmissions," in *2023 IEEE Global Communications Conference (GLOBECOM)*, 2023, doi: 10.1109/GLOBECOM54140.2023.10437066.

[10] B. R. Konda, V. M. R. Tummala, S. K. R. Ganugapenta, P. K. Siraparapu, A. Hazra, and M. Gurusamy, "FedWAvg: Mitigating Model Contamination in UAV Networks through Federated Weighted Average for Weather Forecasting," in *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, 2024, doi: 10.1109/VTC2024-Fall63153.2024.10757753.

[11] R. Nachtigall, M. L. Haller, A. Wagner, and V. Walter, "Revealing the Effects of Data Heterogeneity in Federated Learning Regression Models for Short-Term Solar Power Forecasting," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3479737.

[12] S. Chen *et al*., "Prompt Federated Learning for Weather Forecasting: Toward Foundation Models with Parameter-Efficient Fine-Tuning," in *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3855–3863, 2023.

[13] A. C. Ojha, D. K. Yadav, and B. Ashwini, "Federated Learning Paradigms in Network Security for Distributed Systems," in *2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG)*, 2023, doi: 10.1109/ICTBIG59752.2023.10456162.

[14] S. Jin, Q. Li, S. Liu, O. K. Joel, and X. Ge, "Hybrid Transformer-KAN Within Federated Learning Framework: A Novel Machine Learning Approach for Improved Short-Term Weather Forecasting," *IEEE Access*, vol. 13, 2025, doi: 10.1109/ACCESS.2025.3591119.

[15] N. C. Madala, S. D. S. P. D. Tallapudi, V. S. Malladi, *et al*., "Improving Weather Forecast Accuracy Using Hybrid Machine Learning Algorithms," in *2024 IEEE 16th International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 348–352, 2024.

[16] T. Lei, Y. Dong, C. Jin, J. Min, and C. Han, "Research on Multi-Model Ensemble Machine Learning Methods for Temperature Forecasting," in *2023 International Conference on Computers, Information Processing and Advanced Education (CIPAE)*, pp. 428–433, 2023.

[17] R. Chen, D. Shi, X. Qin, D. Liu, M. Pan, and S. Cui, "Service Delay Minimization for Federated Learning Over Mobile Devices," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, Apr. 2023.

[18] N. Bakir, R. Louis, V. Pak, and K. Samrouth, "Towards Secure Federated Learning: Enhancing Privacy and Robustness in Decentralized AI Systems," in *2024 International Conference on Computer and Applications (ICCA)*, 2024, doi: 10.1109/ICCA62237.2024.10928052.