UAV-ASSISTED TINYML TRANSFER LEARNING FOR EDGE SOIL-MOISTURE FORECASTING IN URBAN AND PERI-URBAN SMART FARMING

Wenping Wang
Rolf Brühl

Distributed irrigation intelligence is increasingly relevant to smart-city systems because urban and peri-urban food production now depends on reliable sensing, low-power communications, and resource-efficient water management. This paper presents a UAV-assisted tiny machine learning (TinyML) framework for edge soil-moisture forecasting using transfer learning on low-power internet of things nodes. The system combines bespoke ESP32-based sensing hardware, unmanned aerial vehicle (UAV)-enabled over-the-air model delivery, and lightweight deep learning inference for local decision support. The reported implementation uses two datasets: a five-site public dataset collected at 10-minute intervals over three years, and a three-month field dataset collected in Amman, Jordan, using a custom capacitive soil-moisture platform. The forecasting pipeline applies hourly aggregation, interpolation of sparse missing values, min-max scaling, and seasonal-trend decomposition using Loess before training compact deep neural network (DNN) and long short-term memory (LSTM) models. The transfer-learning experiment shows that only 441 trainable parameters are updated out of 5,293 total DNN parameters; without transfer learning the model needs more than 300 epochs to converge, whereas transfer learning reduces convergence to fewer than 25 epochs on average and achieves an R^2 *of 92.9%. In the reported edge deployment case, a compressed DNN with architecture* $40 \times 20 \times 10 \times 1$ *occupies 7,920 bytes, produces inference in 97.80 ms, attains an average* R^2 *of 97.13% with an MSE of 0.0036, and can be transferred by over-the-air update in under 10 s. An 8-unit LSTM reaches 99.8% average* R^2 *with an MSE of* 7.9228×10^{-5}*, while larger LSTM configurations in the full performance sweep deliver validation* R^2 *values above 99.9%. Framed for smart-city and urban development scholarship, the study demonstrates that edge-native irrigation intelligence can reduce communication burden, improve resilience in connectivity-constrained environments, and support data-driven water stewardship in distributed urban agriculture.*

Index Terms — TinyML, transfer learning, UAV, smart farming, smart cities, urban agriculture, soil-moisture forecasting, edge intelligence

© The author(s) 2024. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY 4.0) license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.66033/judsc2024-108

INTRODUCTION

Urban and peri-urban agriculture is increasingly being treated as a component of smart-city infrastructure rather than as a purely rural production activity. In dense and rapidly expanding metropolitan regions, local food production, community green spaces, and peri-urban cultivation all depend on efficient water use, low-latency sensing, and resilient digital coordination. For this reason, edge-native forecasting tools that can operate with intermittent connectivity are directly relevant to the broader agenda of urban development and smart cities.

The operational difficulty is that low-cost internet of things (IoT) nodes used in irrigation and environmental monitoring are typically deployed with tight power, memory, and communication constraints. Permanent cloud connectivity is not always feasible, and long-range uplinks can erode battery life and increase communication overhead. A promising alternative is to combine low-power sensing with unmanned aerial vehicles (UAVs) that intermittently provide data ferrying, local model delivery, and short-range communication windows. When this communication pattern is coupled with tiny machine learning (TinyML), forecasting can be performed directly on the edge node instead of relying on continuous cloud inference.

This paper presents a complete manuscript built around that architecture. The study uses compact deep neural network (DNN) and long short-term memory (LSTM) models to predict future soil-moisture conditions on an ESP32-class edge device, while using UAV-assisted transfer learning (TL) to adapt trained models across sensing locations. The core technical contribution is the integration of four elements in a single operational pipeline: (i) custom field sensing, (ii) lightweight time-series forecasting, (iii) transfer learning for rapid adaptation to a new site, and (iv) over-the-air (OTA) model delivery through a UAV.

The work is well aligned with the scope of *Journal of Urban Development and Smart Cities* because it addresses three concerns central to smart-city systems research: resource-efficient water management, resilient edge intelligence for infrastructure in weak-connectivity zones, and the use of mobile platforms to extend digital services where fixed network availability is limited.

The manuscript is organised as follows. Section describes the sensing platform, UAV-assisted system architecture, and datasets. Section presents the forecasting formulation, preprocessing pipeline, model structures, and evaluation metrics. Section reports the deployment and forecasting results using the values explicitly documented in the source study. Section interprets these findings in the context of smart-city water stewardship and urban edge infrastructure. Section concludes the paper.

SYSTEM ARCHITECTURE AND DATA

Operational architecture

The framework divides sensing, learning, and model delivery into distinct but coordinated stages. Ground sensors collect soil and ambient environmental readings, store recent values locally, and run lightweight forecasting models on-device. A UAV acts as a mobile communication and delivery layer: it aggregates sensor data when needed, transfers updated models by OTA update, and reduces the requirement for continuous wide-area connectivity.

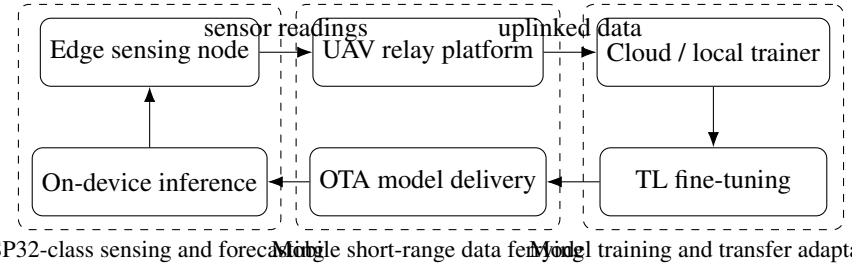

Figure 1: Bi-directional UAV-assisted edge learning workflow for soil-moisture forecasting. The system couples local sensing and inference with intermittent mobile delivery of updated TinyML models.

Edge hardware and communications

Each sensor node is custom-made and carries three measurements: (i) a capacitive soil-moisture sensor that measures volumetric water content, (ii) an air relative humidity sensor, and (iii) an air temperature sensor. The sensing node is built around an ESP32 module. The source study reports 512 bytes of EEPROM, which is sufficient for only 128 floating-point values, corresponding to roughly 21 hours of soil-humidity readings at a 10-minute sampling interval. The ESP32's 4 MB of flash therefore becomes the practical storage location for both sensor logs and the deployed machine-learning model.

For communications, the study uses WiFi and the MQTT protocol to exchange data between the edge device and the UAV-hosted broker. The UAV platform also maintains an onboard SQL database to store collected readings before transfer-learning adaptation and re-publication of an updated model.

UAV trajectory and mission profile

The UAV used in the reported implementation is a hexacopter with a Pixhawk-PX4 32-bit flight controller executing an autonomous waypoint mission. In the illustrated deployment, 14 sensors are distributed across a 90 m $\times$ 150 m area. The UAV hovers above each sensor at a height of 8–12 m to maintain a reliable WiFi connection. The hover duration is 90 s per sensor, with 10 s travel time between waypoints. For 14 sensors, the total mission time is:

$$14 \times 90 + 14 \times 10 = 1400 \text{ s} = 23.33 \text{ min}. \tag{1}$$

Given the reported 25-minute battery life, the UAV can complete this full mission within one charge cycle.

Datasets

The framework uses two datasets. The first is a public dataset from Street and Wookey, containing soil relative humidity, air temperature, and air humidity readings from five distinct sites over three years, sampled every 10 minutes. Measurements were taken using ECH2O EC-5 volumetric moisture sensors at a depth of 5 cm. The second dataset is a field deployment collected in Amman, Jordan, using the custom ESP32-based sensor platform. It contains three months of consecutive measurements at 10-minute intervals.

Table 1: Reported dataset and deployment facts used in the forecasting framework.

Item	Reported characteristic
Public source dataset	Five sites; three years; 10-minute sampling; soil relative humidity, air temperature, and air humidity
Field dataset	Amman, Jordan; three months; 10-minute sampling; custom capacitive soil-moisture platform
Aggregation	Six consecutive 10-minute readings averaged into one hourly sample
Sensor stack	Soil volumetric water content, air relative humidity, and air temperature
Illustrated UAV mission	14 sensors over 90 m $\times$ 150 m; 8–12 m hover height; 90 s hover per node; 10 s inter-waypoint travel
Flight endurance	25-minute battery life; 23.33-minute total mission in the illustrated scenario

FORECASTING METHODOLOGY

Time-series formulation

The data are first converted from 10-minute sampling to an hourly time series by averaging each non-overlapping block of six samples. If the resulting hourly series is denoted by

$$\mathbf{y} = \{y_1, y_2, \ldots, y_{N_s}\}, \tag{2}$$

then the forecasting problem is to predict a future value using a fixed look-back window. For a look-back length L_b and a prediction horizon S_a, the model uses

$$\{y_1, y_2, \ldots, y_{L_b}\} \tag{3}$$

as input and learns a regression mapping for the future target

$$y_{L_b+S_a}. \tag{4}$$

In the reported experiments, short-horizon forecasting is emphasised, including one-hour-ahead and two-hour-ahead prediction.

Data cleaning and pre-processing

The pre-processing pipeline includes four steps:

1. interpolation of sparse missing values;
2. aggregation of six 10-minute samples into one hourly value;
3. min-max scaling;
4. seasonal-trend decomposition using Loess (STL).

The min-max transformation is defined as

$$\hat{y} = \frac{y - \min(y)}{\max(y) - \min(y)}, \tag{5}$$

and the inverse transformation is

$$y = y'\big(\max(y) - \min(y)\big) + \min(y), \tag{6}$$

where y' denotes the predicted value in scaled form.

STL decomposition is then used to separate the time series into trend, seasonal, and remainder components:

$$Y(t) = LT(t) + ST(t) + R(t). \tag{7}$$

The source study also reports a trend-strength measure:

$$S_T = \max\left\{0, 1 - \frac{\mathrm{Var}(R(t))}{\mathrm{Var}(LT(t) + R(t))}\right\}. \tag{8}$$

For the illustrated 21-day decomposition example, the reported trend strength is 92.78%, indicating that the trend component strongly drives the time series.

DNN forecasting model

The DNN is a fully connected network with rectified linear unit (ReLU) activations in the hidden layers and a linear output layer. Given an input vector $\mathbf{x} \in \mathbb{R}^N$ formed from previous soil-moisture readings, the hidden states are

$$\mathbf{h}_1 = \mathrm{ReLU}(W_1\mathbf{x} + \mathbf{b}_1), \tag{9}$$
$$\mathbf{h}_2 = \mathrm{ReLU}(W_2\mathbf{h}_1 + \mathbf{b}_2), \tag{10}$$
$$\mathbf{h}_3 = \mathrm{ReLU}(W_3\mathbf{h}_2 + \mathbf{b}_3), \tag{11}$$

and the prediction is

$$\hat{y} = W_4\mathbf{h}_3 + b_4. \tag{12}$$

The paper evaluates multiple DNN footprints, with two main hidden-layer families: $80 \times 40 \times 20 \times 1$ and $40 \times 20 \times 10 \times 1$.

LSTM forecasting model

For time-series forecasting, the study also evaluates compact single-layer LSTM models. The recurrent update follows the standard gating mechanism:

$$\mathbf{f}_t = \sigma\big(W_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f\big), \tag{13}$$
$$\mathbf{i}_t = \sigma\big(W_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\big), \tag{14}$$
$$\mathbf{o}_t = \sigma\big(W_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\big), \tag{15}$$
$$\tilde{\mathbf{C}}_t = \tanh\big(W_C[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C\big), \tag{16}$$
$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t, \tag{17}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t). \tag{18}$$

The reported LSTM performance sweep tests structures of 64×1, 32×1, and 16×1. During TL, the study appends an additional dense layer after the LSTM layer to capture differences in the new dataset.

Transfer-learning protocol

Transfer learning is used to adapt a pre-trained model from one domain to a related target domain with limited new data. In the reported DNN transfer case, the pre-trained model is reused and only a subset of weights is tuned on the target deployment. The study states that just 441 parameters are trainable during TL, out of 5,293 total DNN parameters.

Table 2: Reported transfer-learning outcomes for the DNN TL experiment.

Metric	**Reported value**
Total DNN parameters	5,293
Trainable parameters during TL	441
Baseline convergence without TL	More than 300 epochs for R^2 convergence
Convergence with TL	Fewer than 25 epochs on average
Reported TL accuracy	$R^2 = 92.9\%$
TL configuration	DNN with $L = 80$, $M = 40$, $N = 20$, plus one extra dense hidden layer with 20 nodes
Forecast horizon	Two hours ahead

Evaluation metrics

The study evaluates model quality using mean squared error (MSE) and the coefficient of determination R^2:

$$\mathrm{MSE}(y,\hat{y}) = \frac{1}{N_s}\sum_{i=1}^{N_s}(y_i - \hat{y}_i)^2, \tag{19}$$

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{N_s}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N_s}(y_i - \bar{y})^2}. \tag{20}$$

For embedded deployment, the paper also reports RAM footprint, flash footprint, and inference time on the ESP32 MCU.

Algorithm 1 UAV-assisted TinyML forecasting workflow

Require: Source dataset $\mathscr{D}_s$, target dataset $\mathscr{D}_t$, look-back L_b, forecast horizon S_a

1: Aggregate six 10-minute readings into one hourly sample
2: Interpolate sparse missing values
3: Apply min-max scaling and STL decomposition
4: Construct supervised input-output pairs using L_b past values and target $y_{L_b+S_a}$
5: Train a compact DNN or LSTM model on $\mathscr{D}_s$
6: Fine-tune a subset of model parameters on $\mathscr{D}_t$ using transfer learning
7: Package the compressed model for OTA delivery
8: UAV visits the target node, exchanges data, and publishes the updated model
9: Edge node stores the model and performs local forecasting for incoming hourly windows

RESULTS AND EMBEDDED PERFORMANCE

Reported deployment benchmark

A concrete edge deployment benchmark is reported for the DNN and the compact LSTM in the one-hour-ahead forecasting setup shown in the source study's main demonstration figure. The DNN uses a $40 \times 20 \times 10 \times 1$ fully connected structure and yields a model size of 7,920 bytes, an average inference time of 97.80 ms (10.3 samples per second), an average R^2 of 97.13%, and an MSE of 0.0036. The compressed model can be mounted in the MCU file system, and the OTA transfer time is reported as less than 10 s on average.

For the corresponding compact LSTM benchmark, the reported model uses one LSTM layer with 8 units and one dense output layer. The model size is 32,816 bytes. The paper reports an average R^2 of 99.8% and an MSE of 7.9228×10^{-5}, showing the substantial accuracy advantage of the recurrent model at the cost of a larger footprint.

Table 3: Key reported deployment outcomes for the compact benchmark models.

Model	Flash (bytes)	Inference metric	MSE	R^2
DNN ($40 \times 20 \times 10 \times 1$)	7,920	97.80 ms (10.3 samples/s)	0.0036	97.13%
LSTM (8 units)	32,816	Reported as higher-throughput edge inference	7.9228×10^{-5}	99.8%

DNN footprint-performance sweep

Table 4 reproduces the reported DNN performance envelope across two hidden-layer families and multiple look-back lengths. Two patterns are clear. First, decreasing the look-back window reduces flash usage and inference time. Second, the smaller $40 \times 20 \times 10 \times 1$ architecture retains strong performance while delivering a much lower flash footprint and substantially faster inference than the larger $80 \times 40 \times 20 \times 1$ design.

Table 4: Reported results for forecasting with DNNs on the ESP32 MCU.

Input length	DNN structure	RAM (bytes)	Flash (bytes)	Inference (ms)	MSE	Val. MSE	Val. R^2 (%)
20	$80 \times 40 \times 20 \times 1$	30,100	27,192	611.15	0.00380	0.00537	95.01
18	$80 \times 40 \times 20 \times 1$	30,084	26,240	594.98	0.00347	0.00508	95.28
16	$80 \times 40 \times 20 \times 1$	30,084	25,320	569.85	0.00425	0.00598	94.15
14	$80 \times 40 \times 20 \times 1$	30,068	24,432	571.66	0.00353	0.00500	96.07
12	$80 \times 40 \times 20 \times 1$	30,068	23,576	552.59	0.00395	0.00559	95.20
10	$80 \times 40 \times 20 \times 1$	30,052	22,752	516.20	0.00358	0.00518	95.92
8	$80 \times 40 \times 20 \times 1$	30,052	21,960	505.57	0.00351	0.00491	96.32
6	$80 \times 40 \times 20 \times 1$	30,036	21,200	495.92	0.00323	0.00453	96.84
4	$80 \times 40 \times 20 \times 1$	30,036	20,472	486.66	0.00357	0.00463	96.85
2	$80 \times 40 \times 20 \times 1$	30,020	19,776	453.16	0.00378	0.00459	97.04
20	$40 \times 20 \times 10 \times 1$	30,100	11,672	160.29	0.00363	0.00516	95.31
18	$40 \times 20 \times 10 \times 1$	30,100	11,040	150.32	0.00382	0.00543	95.40
16	$40 \times 20 \times 10 \times 1$	30,084	10,440	132.07	0.00334	0.00455	96.10
14	$40 \times 20 \times 10 \times 1$	30,084	9,872	124.78	0.00307	0.00427	96.84
12	$40 \times 20 \times 10 \times 1$	30,068	9,336	116.51	0.00347	0.00454	96.70
10	$40 \times 20 \times 10 \times 1$	30,068	8,832	110.00	0.00290	0.00416	96.96
8	$40 \times 20 \times 10 \times 1$	30,052	8,360	102.85	0.00257	0.00370	97.44
6	$40 \times 20 \times 10 \times 1$	30,052	7,920	97.80	0.00251	0.00350	97.14
4	$40 \times 20 \times 10 \times 1$	30,036	7,548	92.78	0.00246	0.00333	98.17
2	$40 \times 20 \times 10 \times 1$	30,036	7,172	87.44	0.00340	0.00426	97.35

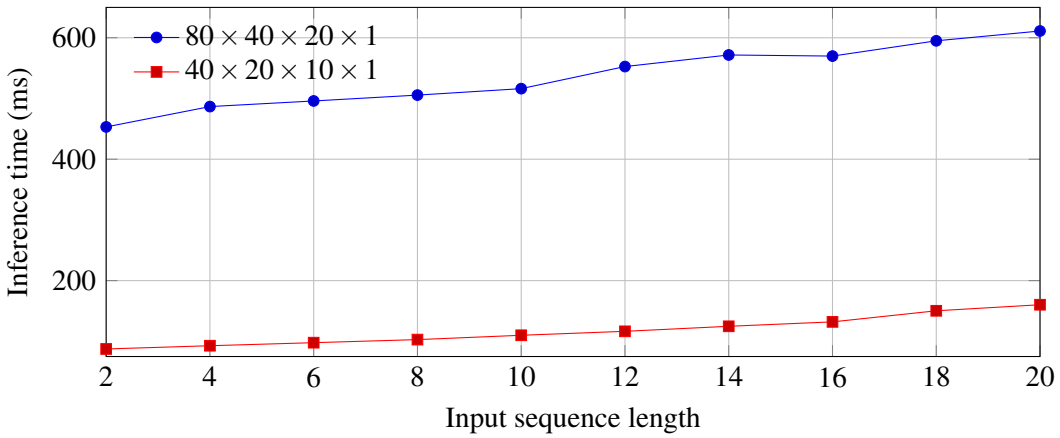

Figure 2: Reported DNN inference time as a function of look-back length. The smaller hidden-layer family achieves a large reduction in latency while preserving strong validation accuracy.

LSTM footprint-performance sweep

Table 5 presents the reported LSTM performance sweep. All tested configurations achieve validation R^2 values above 99.9%, while shorter input sequences and smaller unit counts reduce both flash usage and latency. The smallest reported footprint is obtained with a 9-step input and a 16×1 LSTM, requiring 38,576 bytes of flash and 5.5 ms per inference.

Table 5: Reported results for forecasting with LSTMs on the ESP32 MCU.

Input length	**LSTM structure**	**RAM (bytes)**	**Flash (bytes)**	**Inference (ms)**	**MSE**	**Val. MSE**	**Val. R^2 (%)**
15	64×1	32,768	81,440	34.5	0.000054	0.000046	99.96
12	64×1	32,768	68,016	27.0	0.000096	0.000096	99.94
9	64×1	32,768	50,400	17.5	0.000067	0.000064	99.95
15	32×1	32,768	68,320	17.0	0.000076	0.000076	99.93
12	32×1	32,768	54,896	13.5	0.000047	0.000040	99.95
9	32×1	32,768	42,064	10.0	0.000076	0.000073	99.93
15	16×1	32,768	64,832	9.0	0.000037	0.000037	99.96
12	16×1	32,768	51,408	7.5	0.000085	0.000085	99.91
9	16×1	32,768	38,576	5.5	0.000051	0.000051	99.95

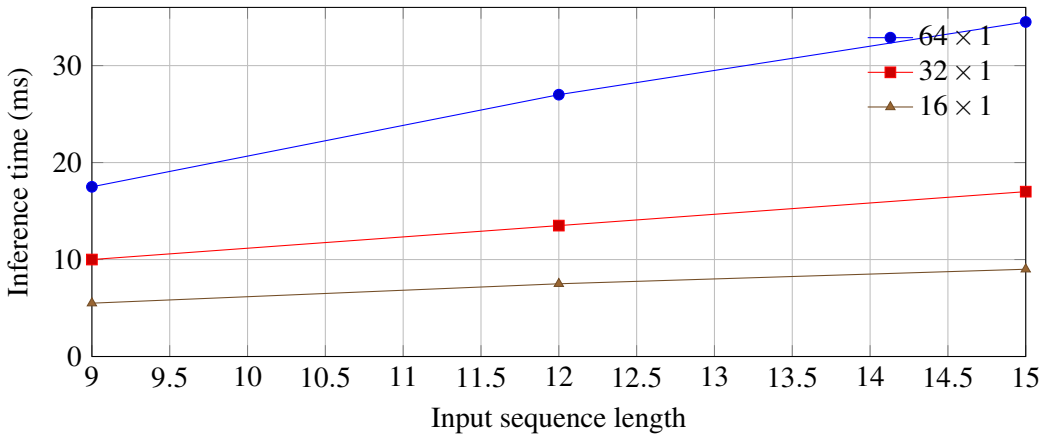

Figure 3: Reported LSTM inference time across tested sequence lengths and unit counts. Lower-complexity recurrent models remain highly accurate while reducing flash and latency costs.

Transfer-learning behaviour

The TL experiment is important because it establishes the practical value of UAV-assisted model adaptation rather than only demonstrating static deployment. The reported DNN TL case shows that the majority of model parameters can remain fixed while a small trainable subset is adapted to the new site. This sharply reduces the training burden: the model requires more than 300 epochs for convergence when trained in the usual way, but fewer than 25 epochs on average when transfer learning is applied. The resulting R^2 of 92.9% is presented as satisfactory for two-hour-ahead site adaptation.

Together with the compact footprint and short OTA transfer time, these results show that the system is not merely a laboratory inference demonstration. It is a functioning deployment strategy in which the UAV serves as a practical bridge between fixed sensing nodes and higher-level training resources.

DISCUSSION

Three conclusions are particularly relevant for smart-city scholarship.

First, the study shows that localised environmental intelligence can be embedded into low-power infrastructure without requiring continuous cloud dependence. In smart-city terms, this matters because edge autonomy increases resilience in zones where broadband coverage is weak, intermittent, or too costly to maintain for dense sensor fleets.

Second, the paper quantifies the trade-off between prediction accuracy and embedded resource use. The DNN sweep demonstrates that substantial reductions in flash size and latency can be achieved by selecting smaller hidden-layer structures and shorter look-back windows, while the LSTM sweep shows that recurrent models can deliver very high predictive accuracy with manageable computational cost. This type of engineering trade-off is directly relevant to urban infrastructure design, where scalability and maintenance burden often determine whether a sensing system is viable.

Third, the transfer-learning results are operationally important. Urban and peri-urban agricultural environments are heterogeneous: soil composition, microclimate, irrigation practice, and local weather exposure vary substantially across sites. The fact that only 441 of 5,293 DNN parameters are trainable during the reported

TL phase indicates that adaptation can be made lightweight enough to fit intermittent mobile service windows. In practice, that means a single UAV mission can both collect data and deliver improved inference capability to distributed nodes with limited downtime.

From the perspective of *Journal of Urban Development and Smart Cities*, the framework is best understood as a distributed urban services architecture. The direct application is smart irrigation, but the underlying design pattern is broader: a low-power edge device performs local inference, a mobile platform extends the communication envelope, and training remains centralised or semi-centralised. This same pattern can be reused for urban green-infrastructure monitoring, microclimate observation, flood-sensitive landscaping, and other spatially distributed civic sensing tasks.

The study also identifies several future directions that remain compelling. The source paper explicitly notes that more advanced architectures such as CNNs, CNN-LSTM, and bidirectional LSTM should be implemented through the TinyML framework. It also highlights the need to address memory and computational constraints more aggressively and to examine federated learning, where training is performed at the edge before model parameters are sent back upstream. These directions are consistent with the next stage of urban edge intelligence: more adaptive, more distributed, and less dependent on stable wide-area connectivity.

CONCLUSION

This paper presents a complete UAV-assisted TinyML framework for edge soil-moisture forecasting using transfer learning on low-power IoT nodes. The reported system combines bespoke ESP32-based sensing hardware, short-range UAV communications, compact DNN and LSTM models, and OTA delivery of updated models. The empirical results show that the framework is practically deployable: a compact DNN can run in 97.80 ms with a 7,920-byte footprint and OTA transfer under 10 s, while LSTM configurations achieve validation R^2 values above 99.9% across the main performance sweep. In the transfer-learning case, adaptation is limited to 441 trainable parameters out of 5,293 total and reduces convergence from more than 300 epochs to fewer than 25, with an R^2 of 92.9%.

The broader significance is that the framework provides an edge-native model for water-aware smart-city infrastructure. By reducing communication burden, preserving local autonomy, and enabling intermittent model delivery through mobile platforms, the approach supports resilient and scalable digital services for urban and peri-urban agriculture. It therefore contributes not only to smart farming, but also to the broader design of distributed, resource-efficient, and connectivity-aware systems for urban development.

REFERENCES

[1] Pan, S. J., and Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[2] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[3] Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. STL: A seasonal-trend decomposition procedure based on Loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[4] Warden, P., and Situnayake, D. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, Sebastopol, CA, 2019.

[5] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[6] Liakos, K. G., Busato, P., Moshou, D., Pearson, S., and Bochtis, D. Machine learning in agriculture: A review. *Sensors*, 18(8):2674, 2018.

[7] Tsouros, D. C., Bibi, S., and Sarigiannidis, P. G. A review on UAV-based applications for precision agriculture. *Information*, 10(11):349, 2019.

[8] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, Cambridge, MA, 2016.

Wenping Wang, ESCP Business School, Heubnerweg 8-10, 14059, Berlin, Germany

Rolf Brühl, ESCP Business School, Heubnerweg 8-10, 14059, Berlin, Germany; bruehl@escp.eu

Manuscript Published; 02 October 2024.