ToEFL: A Novel Approach for Training on Edge for a Smart Agriculture Application

Alakananda Mitra*
Nebraska Water Center,
University of Nebraska-Lincoln
Lincoln, NE, USA
amitra6@unl.edu

Saraju P. Mohanty
Dept. of Computer Science and Engineering,
University of North Texas
Denton, TX, USA
saraju.mohanty@unt.edu

Elias Kougianos
Dept. of Electrical Engineering,
University of North Texas
Denton, TX, USA
elias.kougianos@unt.edu

ABSTRACT

Billions of devices on the Internet of Things (IoT) capture massive amounts of data from everyday events, raising privacy and security concerns. The current technology standard transfers the majority of this data over the internet stores it, processes it, and uses it to train machine learning (ML) and deep learning (DL) models on the cloud server. Then, edge devices make decisions. However, this threatens data privacy and security. Federated learning (FL) can be helpful in this scenario because it transfers the model to the data. However, issues like poor network connectivity, low bandwidth, and network latency demand more research on edge-based models, on-device training, and lightweight and low-overhead communication protocols. In this article, we present ToEFL, an incremental training (T) on (o) edge (E) method for an FL network in a smart agriculture application, addressing data bottlenecks and low computational capability constraints. We selected plant disease detection as the application. The whole idea is to train the model on the edge device to reduce the frequency of server updates and make ToEFL suitable for a remote village where internet service may be poor. The ToEFL system can choose between server and local models based on model performance, with the latter more tailored to a local dataset.

CCS CONCEPTS

• **Computing methodologies → Machine learning**;

KEYWORDS

Smart Agriculture, Internet of Agricultural Things (IoAT), Edge-AI, Federated Learning, Data Privacy.

ACM Reference Format:
Alakananda Mitra, Saraju P. Mohanty, and Elias Kougianos. 2024. ToEFL: A Novel Approach for Training on Edge for a Smart Agriculture Application. In *Proceedings of Great Lakes Symposium on VLSI 2024 (GLSVLSI 2024).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In the last few decades, the digital revolution has resulted in significant development in areas like artificial intelligence (AI), machine learning (ML), deep learning (DL), natural language processing (NLP), cloud computing, edge computing, and the Internet-of-Things (IoT). Billion-IoT devices collect a large amount of data, which demands a lot of computational power and cloud-based solutions to train and infer ML/DL models (Fig. 1a). However, network bandwidth and latency impair end device response, hindering time-sensitive applications [29]. In edge-cloud-continuum solutions (Fig. 1b) the AI-enabled edge handles real-time data collection, processing, and prediction for time-sensitive applications, while the cloud handles heavy-load tasks like training [35]. These two solutions transfer data from the origin to the server. Nevertheless, the need for distributed training, such as federated learning and edge-oriented solutions, arises due to global data privacy and regulation activities (Fig. 1c). Edge-based solutions, such as neural architectures, quantization, pruning, and effective training algorithms, are prioritized for low-powered devices with limited processing power [3]. FL is allowing distributed training to trade-off between data privacy and computational load.

FL is a computing strategy aiming to train an ML or DL model in a decentralized manner instead of training a centralized model on a server [20]. It helps to overcome the data regulation and privacy aspects, as well as unreliable and low-bandwidth internet connections. Clients, such as mobile phones, personal computers, tablets, and application-specific IoT devices, act as decentralized training nodes and actively participate in training with the local data. Once the training is complete these nodes send the local model updates to the server. All the updates are then aggregated to generate a global model in the server.

The rest of the article is organized as follows: the research problem, the proposed solution, and its significance are discussed in Section. 2. We briefly describe recent work on various aspects of FL and the motivation of our study in Section 3. Section 4 presents the methodology of our work focusing on the model architecture, training protocol, the data pipeline, and the implementation in detail. The results are discussed in Section 5. Finally, the conclusions, limitations, and future work have been discussed in Section 6.

2 CONTRIBUTION OF THE CURRENT PAPER

2.1 Problem Addressed

In [12] and [13], the authors explored the feasibility of deployment of CNNs in TinyML devices and extended that idea to deploy in

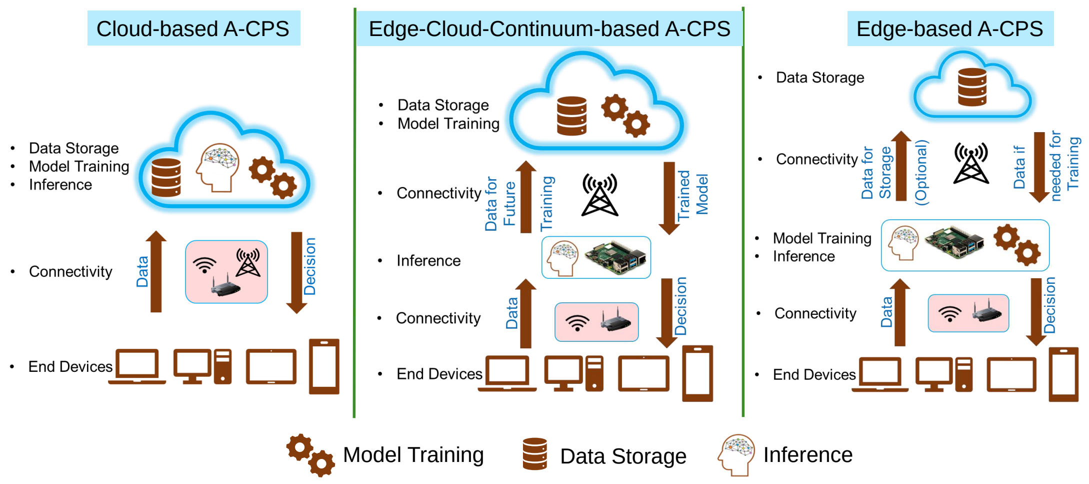


Figure 1: Types of Cyber-Physical Systems for Agriculture

a decentralized federated learning (FL) environment. This paper also aims to prioritize data privacy. Here, we studied whether complete training on an embedded board is viable compared to partial training in [13].

2.2 Solution Proposed

To address this we proposed an efficient training method with a data pipeline to avoid the memory bottleneck for embedded systems. For embedded systems, the majority of the research is on smaller size neural architecture or data pipeline, validated with handwritten digits dataset, MNIST (grayscale images of size 28×28) [11] and they don't reflect the real-world complexity. Hence, each model/method should be validated using a real-life domain dataset.

2.3 Significance of the Solution

Our contribution can be summarized as:

- We present an incremental algorithm for training nodes of an FL network to overcome the computational constraints.
- The unsupervised training method lets us add any unknown class.
- A model selector block has been proposed to emphasize the local database effect on local models rather than the global model.
- Two different datasets have been used to validate the training protocol.
- The training protocol for a single node has also been verified in a Raspberry Pi-4B single-board computer.

3 RELATED WORK

In this section, we briefly discuss current FL research on training methodologies and FL networks.

In [5], authors explored the effects of the number of clients sampled per round (cohort size) on FL. Strategies like *structured updates* and *sketched updates* were used in [20] to reduce the uploading cost from clients to the server and how the method efficiency is affected by the number of communications between the clients and server, has also been studied. The authors used the Federated Averaging algorithm [23] to reduce the number of communications. The convolutional model from [31] was trained with CIFAR-10 data and also an LSTM *next-word* prediction model with data from Reddit [2].

Network communication is a major limitation in FL as the local model needs to send the updates to the global model. In [25], the authors presented a method using rate-distortion theory to reduce the average communication cost. A recurrent neural network (RNN) model was trained on-device to predict the next word in mobile virtual keyboard [16]. The work shows how to train language models on client devices. To improve the performance for non-IID data, the authors in [21] proposed a model contrastive learning approach where the similarity between the model representations is used to rectify the local training for the clients. An open-source framework for training ML models in the FL setting was presented in [28]. It facilitates the training of ML models in an FL production environment. Table. 1 describes some more studies, their methods, and findings.

From the above discussion, it is clear that most efforts to improve the FL paradigm focused on improving client-server connectivity or cloud training methodologies. However, localized training is necessary considering the heterogeneity of user data [18]. Additionally, weak network connections in remote communities can hinder recurring client model updates to the server. Using edge-based model training instead of fine-tuning can overcome this limitation.

In Agriculture, vast IoT sensors collect massive amounts of data and are an ideal application field for FL. Researchers are trying to solve various agricultural issues like plant disease [12, 17], disease and pest detection [8], crop classification [19], intrusion-detection for IoT devices [22] and so on. However, the majority of the research is focused on a specific agro issue and does not address the training aspects of FL in an agriculture setting.

4 PROPOSED METHOD

The discussion in Section 3 revealed that the edge-based training of AI models is one of the ways to efficiently run an FL-based

Table 1: Works on Various Aspects of Federated Learning

Papers	Methods	Findings
[18]	Model Agnostic Meta-Learning	FL is personalization focused. Used-based fine-tuning of the model presents a more accurate model than when optimizing is done for the global model.
[4]	A generative model	Manual dataset sanity is not possible in FL. The model debugged common data issues where manual data inspection is impossible.
[33]	Norm thresholding and weak differential privacy	As FL is a decentralized process, adversarial attacks pose a serious problem in FL. The attacks could be resisted with the mentioned strategies without affecting the performance.

application. We propose ToEFL, a novel method for Training (T) on Edge (E) for an FL network of a smart agricultural [26] application.

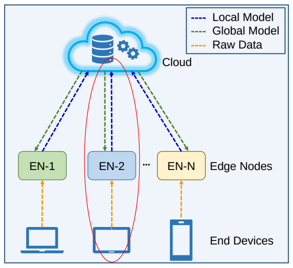
4.1 Overview

Here, we propose an incremental training protocol (Algorithm.1). It allows the resource-constrained edge nodes of an FL system to handle data processing, model training, model selection, and inference. When images are available to a node, unsupervised learning allows adding new and unknown classes. A high-level overview of an FL network using ToEFL is provided in Fig. 2a. It has three levels: end device, edge, and cloud server. (Fig. 2b) shows all modules in an edge node. Figure 2c outlines the key steps: data collection, processing, edge model training, server updates, local model aggregation, global model transmission, edge model selection, and inference. Fig. 3 depicts the workflow of training the local model in an edge node.

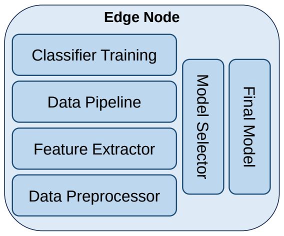
4.2 Network and Training Protocol

The disease classification network has two parts- *feature extractor* and *classifier*. The *feature extractor* extracts the features from the images and the classifier classifies the images based on the feature vectors. However, loading all the extracted feature vectors in a data frame is too large to fit into the memory of an edge device. Hence we followed the training protocol mentioned below:

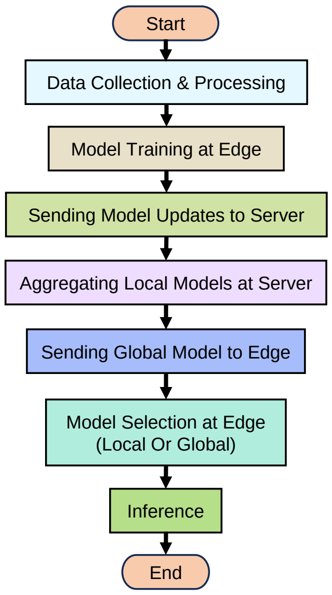
- We used *transfer learning* for faster training and higher accuracy. MobileNetV2 [30] pre-trained on ImageNet [9] dataset has been used as the *feature extractor* and feature vectors were obtained from a pre-specified layer (we omitted the last five layers of MobileNetV2).
- The feature vectors from the input images are extracted batch-wise (in our experiment $batchsize = 32$) to save in a .csv file.
- Accessing all the feature vectors at once by loading them to memory can stall the training. Hence, we train the classifier with the feature vectors of images one by one using Algorithm.1.



(a)



(b)


Start
Data Collection & Processing
Model Training at Edge
Sending Model Updates to Server
Aggregating Local Models at Server
Sending Global Model to Edge
Model Selection at Edge (Local Or Global)
Inference
End

(c)

Figure 2: Details of the Proposed Method, ToEFL. (a) shows the higher-level overview of an FL network. (b) depicts the modules in an edge node. (c) shows the seven main steps of ToEFL of any branch (e.g., the red-circled branch).

- The classifier is encapsulated in a pipeline. Standardization of the feature vectors is first performed in the pipeline to get a standardized distribution with a 0 mean and unit variance.
- We experimented with the *logistic regression* as the final classifier layer to map standardized feature vectors to class labels. As this case is a classical multi-class (M) classification problem, we used *OneVsRest* with *linear regression* to accommodate multiple classes. So, we train the multi-class classification problem as separate M binary class problems where each classifier f_m is trained to determine whether the feature vector belongs to a class m or not where $m \in \{1, 2, ..., M\}$. For a test example c, all M classifiers are run for c, and the highest score is selected. As the optimizer *SGD* (with *learning rate* 0.01) and loss function *log loss* have been used.
- The classifier has also been trained in an unsupervised way so that future unknown classes can be classified.

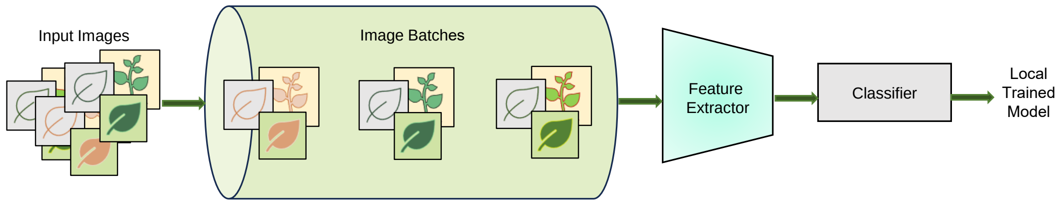


Figure 3: Workflow of Training in an Edge Node.

Algorithm 1 How does ToEFL work?

```
1:  Input: Image $\mathcal{I}$
2:  Output: Prediction $\mathcal{P}$
3:  Random weights $w_0$ are initialized to the global model in the Server;
4:  $w_0$ is sent to all the edge nodes $N$ of the FL network;
5:  for $n \in N$ do    /* Training for all the nodes. */
6:      Set a list for the metric and initialize its value to a empty list;
7:      $counter = 0$;
8:      Set a batch size;
9:      while $image \neq None$ do    /* When images are available to a node. */
10:         Make image batches $\mathcal{B}$.
11:         for $\beta \in \mathcal{B}$ do
12:             Extract feature vectors $\mathcal{F}$ from $\beta$ using the feature extractor module;
13:             Flatten $\mathcal{F}$ for $\beta$;
14:             Save $\mathcal{F}$;
15:             for $f \in \mathcal{F}$ do
16:                 Send $f$ to the classifier pipeline;
17:                 Predict $\mathcal{P}$ for $f$;
18:                 Update the weights;
19:             end for
20:             Delete $\mathcal{F}$
21:         end for
22:         Save metric;
23:         $counter+ = 1$;
24:     end while
25:     if $metric_{counter} \geqslant metric_{counter-1}$ then
26:         Send the local model $\mathcal{M}_l$ to the server;
27:         Server aggregates the model as per [24];
28:         Server sends the global model $\mathcal{M}_g$ to $n$;
29:     end if
30:     if $metric_{local} \gg metric_{global}$ then
31:         Keep the $\mathcal{M}_l$;
32:     else
33:         Keep the $\mathcal{M}_g$;
34:     end if
35:     Use the trained model $\mathcal{M}$ to predict the unknown image $image_{unknown}$;
36: end for
```

4.3 Data Collection and Processing

To validate the proposed method, two datasets were used. Both datasets are available publicly- the cotton disease dataset [7] and the corn or maize leaf disease dataset [6]. The cotton disease dataset has four classes with 2310 images and the corn dataset also has the same number of classes but with 4124 images. The images were resized to 224×224 and normalized as per the mean and standard deviation of the ImageNet dataset [10] as the *feature extractor* is pre-trained on the ImageNet. Since real-life data come into the stream, it eliminates the need for data storage facilities. Various image augmentation strategies have been performed including vertical and horizontal flip, transpose, random rotation, blurring, and grid distortion.

4.4 Implementation

To validate the performance of ToEFL, we performed the classification task for the datasets [6, 7]. A Raspberry Pi 4 Model B board was used as the edge platform. The data was streamed from the laptop to the Pi using the WiFi. To implement the pipeline and data streaming, the River [27] library was used. The codes are written in Python. NumPy, Pandas, and Keras are other libraries and ML framework used in this work.

5 PERFORMANCE EVALUATION

Table. 2 describes the performance metrics for both datasets. Among those metrics, accuracy tells the total correct predictions by the model. Here, accuracy between $\sim (72-74)\%$ has been obtained for both datasets. However, there is a gap between cotton disease dataset [7] training and testing accuracy. As [7] has healthy and diseased cotton plants, the shape and spread of the plants are different in different images which caused the accuracy difference. However, the training and testing datasets for the corn dataset, consisting of leaf images, show similar types of accuracy. Fig. 4a shows the *confusion matrix (cm)* for no image augmentation scenario and Fig. 4b shows that of with data augmentation. As the name suggests, those metrics describe where the model was confused to predict. It gives us an idea of how to improve the training. From the *cm*, we calculated some other metrics. *Precision* tells how many instances of the positive class were predicted correctly among the predicted positive class and *recall* describes how many relevant instances were correct. So, precision tells more about the quality of the prediction whereas recall emphasizes more about sensitivity or quantity. One-vs-all strategy is followed for the cases of multiple classes like here. In this study, we were able to reach the same range for *precision*, *recall*, and *F1-score* as *accuracy*.

Reasons for lower performance metrics:

- In real life, the data comes continuously, and data storage in resource-constrained edge devices is not advisable. Hence, in real life, data will be coming in different conditions. However, we didn't have many images with various conditions. We need more images with all types of conditions to have better accuracy.
- Image augmentation has helped to get better accuracy for the cotton dataset but degrades the model for the corn dataset. This explains that a model with a higher capacity is needed.
- Image augmentation techniques used here might not be suitable for the used cases.

This study explains how the absence of data variations and meaningful data augmentation can affect the training of any ML model. As training on edge when the data come on the fly is a relatively new area of research [32], an exact match for comparison is not available. Table. 3 compares ToEFL with some works that employed *Raspberry Pi* for training or testing. The table shows no other studies did the training from scratch.

6 CONCLUSION

Federated learning enables IoT devices to train in a shared distributed way. Data is no longer saved in the cloud but is used and stored at the local nodes. However, training in resource-constrained edge devices demands more research on training protocols at the edge, suitable models for TinyML devices, quantization, and pruning techniques. This article presents an incremental training protocol for training an ML model at the edge. As the upload speed is less than the download speed [1], the frequency of local updates to the server has been reduced by extracting the features in batches. The resource constraint has been addressed by training the classifier with individual feature vectors. As ToEFL has been trained in an unsupervised way so that unknown classes can be added to the system. The node training has only been implemented here.

Data streaming strategies have been used to replicate the continuous data flow. On-the-fly data training makes training more challenging because it takes longer to train as the training is performed with the feature vectors one by one. Principal Component Analysis (PCA) can address the high sparsity of the feature vectors. Training with more data can also increase the accuracy.

Future work will include:

- The implementation of the entire FL system with model overwrites facilities- an overwrite facility is needed because there might be different crop fields where ToEFL is used. In such cases, the local model predominates over the global model.
- Training with field data will complete the study.
- A detailed study needs to be performed for the best data augmentation strategies.
- More experiments on the model are needed which can predict complex images.

ACKNOWLEDGEMENT

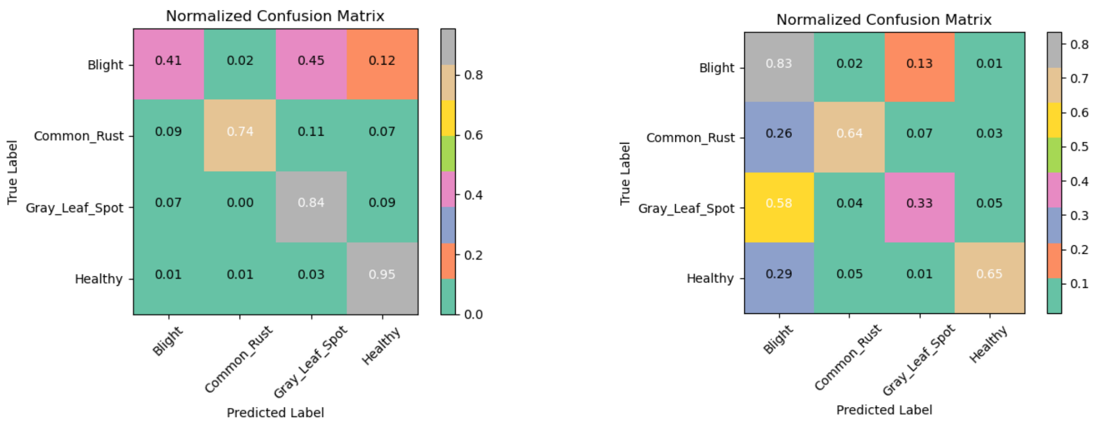
The authors are thankful to Catherine Dockendorf for the initial research on Federated Learning-based approaches in Smart Agriculture Applications.

REFERENCES

[1] 2017. Federated Learning: Collaborative Machine Learning without Centralized Training Data. https://blog.research.google/2017/04/federated-learning-collaborative.html. Accessed on April 06, 2024.
[2] Rami Al-Rfou, Marc Pickett, Javier Snaider, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2016. Conversational contextual cues: The case of personalization and history for response ranking. *arXiv preprint arXiv:1606.00372* (2016).
[3] Alberto Ancilotto, Francesco Paissan, and Elisabetta Farella. 2023. XiNet: Efficient Neural Networks for tinyML. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 16968–16977.
[4] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. 2019. Generative Models for Effective ML on Private, Decentralized Datasets. *CoRR* abs/1911.06679 (2019). arXiv:1911.06679 http://arxiv.org/abs/1911.06679
[5] Zachary Charles, Zachary Garrett, Zhouyuan Huo, Sergei Shmulyian, and Virginia Smith. 2021. On large-cohort training for federated learning. *Advances in neural information processing systems* 34 (2021), 20461–20475.
[6] Corn or Maize Leaf Disease Dataset [n. d.]. https://www.kaggle.com/datasets/smaranjitghose/corn-or-maize-leaf-disease-dataset. Accessed on February 28, 2024.
[7] Cotton Disease Dataset [n. d.]. https://www.kaggle.com/datasets/janmejaybhoi/cotton-disease-dataset/data. Accessed on February 25, 2024.
[8] Fangming Deng, Wei Mao, Ziqi Zeng, Han Zeng, and Baoquan Wei. 2022. Multiple diseases and pests detection based on federated learning and improved faster R-CNN. *IEEE Transactions on Instrumentation and Measurement* 71 (2022), 1–11.
[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition.* Ieee, 248–255.
[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 248–255. https://doi.org/10.1109/CVPR.2009.5206848
[11] Li Deng. 2012. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.
[12] Catherine Dockendorf, Alakananda Mitra, Saraju P Mohanty, and Elias Kougianos. 2023. "Lite-Agro: Exploring Light-Duty Computing Platforms for IoAT-Edge AI in Plant Disease Identification". In *Proc of the 6th IFIP International Internet of Things Conference.* Springer, 371–380. https://doi.org/10.1007/978-3-031-45882-8_25.
[13] Catherine Dockendorf, Saraju P Mohanty, Alakananda Mitra, and Elias Kougianos. 2023. Lite-Agro 2.0: Integrating Federated and TinyML in Pear Disease Classification IoAT-Edge AI. In *Proc. of the IEEE International Symposium on Smart Electronic Systems (iSES).* IEEE, 429–432. https://doi.org/10.1109/iSES58672.2023.00096.
[14] Fotios Drakopoulos, Deepak Baby, and Sarah Verhulst. 2019. Real-time audio processing on a Raspberry Pi using deep neural networks. In *23rd International Congress on Acoustics (ICA 2019).* Deutsche Gesellschaft für Akustik, 2827–2834.
[15] Oliver Dürr, Yves Pauchard, Diego Browarnik, Rebekka Axthelm, and Martin Loeser. 2015. Deep Learning on a Raspberry Pi for Real Time Face Recognition.. In *Eurographics (Posters).* 11–12.
[16] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).
[17] Godwin Idoje, Tasos Dagiuklas, and Muddesar Iqbal. 2023. Federated Learning: Crop classification in a smart farm decentralised network. *Smart Agricultural Technology* 5 (2023), 100277.
[18] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. 2019. Improving Federated Learning Personalization via Model Agnostic Meta Learning. *CoRR* abs/1909.12488 (2019). arXiv:1909.12488 http://arxiv.org/abs/1909.12488
[19] Fawad Salam Khan, Sikandar Khan, Mohd Norzali Haji Mohd, Athar Waseem, Muhammad Numan Ali Khan, Sajid Ali, and Rizwan Ahmed. 2022. Federated learning-based UAVs for the diagnosis of Plant Diseases. In *2022 International Conference on Engineering and Emerging Technologies (ICEET).* IEEE, 1–6.
[20] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).
[21] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 10713–10722.
[22] Dapeng Man, Fanyi Zeng, Wu Yang, Miao Yu, Jiguang Lv, and Yijing Wang. 2021. Intelligent intrusion detection based on federated learning for edge-assisted internet of things. *Security and Communication Networks* 2021 (2021), 1–11.
[23] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics.* PMLR, 1273–1282.
[24] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR*

Table 2: Performance Metrics for ToEFL for Cotton Disease and Corn Leaf Disease Datasets

Dataset	**No. of Training Data**	**Data Aug.**	**Precision**		**Recall**		**F1-Score**		**Accuracy (%)**	
			Macro Avg.	**Micro Avg.**	**Macro Avg.**	**Micro Avg.**	**Macro Avg.**	**Micro Avg.**	**Training**	**Testing**
Corn Dataset	3617	No	0.72	0.72	0.74	0.72	0.69	0.72	75.23	71.72
	18080	Yes	0.67	0.67	0.61	0.67	0.62	0.67	69.48	66.66
Cotton Disease Dataset	2204	No	0.72	0.72	0.74	0.72	0.69	0.72	79.85	72.71
	22041	Yes	0.73	0.73	0.71	0.73	0.74	0.70	81.46	73.91



(a) **(b)**

Figure 4: Confusion Matrices (CM) for Corn Dataset. Fig. shows the CM with no image augmentation and Fig. shows the CM with image augmentation.

Table 3: Comparative Analysis with Existing Studies

Paper	**Hardware Board**	**Initial Training at Edge**	**Retraining at Edge**
[34]	Raspberry Pi 4B	No	Yes
[15]	Raspberry Pi _B	No	No
[14]	Raspberry Pi 3B+	No	No
ToEFL	**Raspberry Pi 4B**	**Yes**	**Yes**

abs/1602.05629 (2016). arXiv:1602.05629 http://arxiv.org/abs/1602.05629

[25] Nicole Mitchell, Johannes Ballé, Zachary Charles, and Jakub Konečný. 2022. Optimizing the Communication-Accuracy Trade-off in Federated Learning with Rate-Distortion Theory. *CoRR* abs/2201.02664 (2022). arXiv:2201.02664 https://arxiv.org/abs/2201.02664

[26] Alakananda Mitra, Saraju P Mohanty, and Elias Kougianos. 2023. Smart Agriculture–Demystified. In *Proc of the 6th IFIP International Internet of Things Conference*. Springer, 405–411. https://doi.org/10.1007/978-3-031-45878-1_28.

[27] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, et al. 2021. River: machine learning for streaming data in Python. (2021).

[28] G Anthony Reina, Alexey Gruzdev, Patrick Foley, Olga Perepelkina, Mansi Sharma, Igor Davidyuk, Ilya Trushkin, Maksim Radionov, Aleksandr Mokrov, Dmitry Agapov, et al. 2021. OpenFL: An open-source framework for Federated Learning. *arXiv preprint arXiv:2105.06413* (2021).

[29] Guoping Rong, Yangchen Xu, Xinxin Tong, and Haojun Fan. 2021. An edge-cloud collaborative computing platform for building AIoT applications efficiently. *Journal of Cloud Computing* 10 (2021), 1–14.

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[31] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).

[32] Bharath Sudharsan, John G Breslin, and Muhammad Intizar Ali. 2020. Edge2train: A framework to train machine learning models (svms) on resource-constrained iot edge devices. In *Proceedings of the 10th International Conference on the Internet of Things*. 1–8.

[33] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. 2019. Can You Really Backdoor Federated Learning? *CoRR* abs/1911.07963 (2019). arXiv:1911.07963 http://arxiv.org/abs/1911.07963

[34] Fushuai Wang, Renren Zheng, Penghui Li, Hanni Song, Dongming Du, and Jingchao Sun. 2021. Face recognition on Raspberry Pi based on MobileNetV2. In *2021 International Symposium on Artificial Intelligence and its Application on Media (ISAIAM)*. 116–120. https://doi.org/10.1109/ISAIAM53259.2021.00031

[35] Mazin Yousif. 2022. "Intelligence in the Continuum". *"IEEE Computer Society"* 2, 3 (2022), 38.