Scalable and High-Performance TinyML Solutions for Plant Disease Detection

Final Report of Team: Tiny_MONO

Mohanasundaram SV*, Purushothaman R**, Ramasamy Srinivasagan***

* Independent Researcher, Bangalore, India, qdigitx@gmail.com
**Student, BS-Data Science & Applications, IIT Madras, India, 22f1001344@ds.study.iitm.ac.in
***Faculty, Computer Engg., CCSIT, King Faisal University, Saudi Arabia, rsamy@kfu.edu.sa

Abstract

The agricultural sector faces several challenges in its efforts to increase production. One of the biggest challenges are diseases and insect pests that destroy crops and lead to a decline in production. This work primarily focuses to bridge the gap between traditional disease diagnosis on agricultural crops. It adopts modern technological advancements in Deep learning by developing a comprehensive Tiny ML application in disease detection. The application will give farmers the knowledge and tools they need to make data-driven decisions such as disease management. Traditional farming practices often lack access to advanced technologies and data-driven insights, making it difficult for farmers to effectively optimize their operations. This work aids precision agriculture and presents techniques for early detection of plant diseases. Specifically, we present two classes of techniques: first, the application of image processing techniques related to ML algorithms and second, the application of deep learning in disease detection. As the techniques necessitate powerful computing equipment that requires a constant power supply and a high bandwidth for their implementation, we propose a new solution more accessible to developing countries, which is based on Tiny Machine Learning (TinyML), the emerging technology of embedded Machine Learning. Although TinyML runs machine learning on low-cost and low-power devices, it can infer from the models obtained by the machine learning algorithms. The idea is to get the model after training and then convert it into a tf-lite model with a size that can be inserted into devices with limited resources.

Index Terms—Tiny Machine Learning, Tensorflow Lite, Machine Learning.

I. Introduction

A noticeable pattern becomes apparent when analysing the state of plant disease detection systems. High-tech diagnostic instruments are more common in rich countries and significantly fewer in less developed ones. There are plenty of factors why diagnosing plant diseases in rural farming regions might be challenging. Here are a few such problems:

1. Limited Expertise
2. Costly sensing techniques
3. Crop Diversity
4. Lack of technological awareness

To mitigate these issues, it is necessary to employ Tiny ML solutions with low-cost and low-power affordable sensors which is scalable and high performance oriented.

II. Methodology

This study adopts a dual-pronged approach to plant disease detection, employing PyTorch for model training and subsequent conversion to the ONNX format. A comparative study is done comparing PyTorch models that have been converted from ONNX and models and that have only been trained in Edge Impulse. Comprehensive criteria assessed by the EON Turner tool, such as latency, RAM, and ROM, serve as a reference for choosing the best model. This approach guarantees a thorough assessment of model performance, addressing the necessity of effectiveness and handling resources when utilising accurate plant disease detection techniques.

Also, a Flutter application that integrates a PyTorch model has been carefully created to help farmers. This application is quite unique and offers farmers an easy-to-use interface to utilise cutting-edge plant disease detection technology. The application has cleverly integrated the PyTorch model, which was previously trained and converted using ONNX. By ensuring that farmers can effortlessly access and utilise state-of-the-art technology through the user-friendly Flutter application interface, this synergistic approach represents a major advancement in improving agricultural practices. The major reason for the approach is that it can be integrated in to mobile application (Flutter based) and easily scalable and accessible by farmers. The same model can be targeted to TinyML –Microcontroller ecosystem supported by Edge Impulse and used for drone and manual scouting.

III. EMBEDDED APPLICATION

The framework used here are as follows:

3.1 Tensorflow Lite Micro

The Tensorflow Lite Micro (TFLM) package is compact enough to fit into a microcontroller's memory and offers a set of tools for effectively deploying machine learning models. For example, it offers an efficient approach for quantizing the model's weights by calculating an effective dynamic range from a representative dataset without sacrificing the model's accuracy. Moreover, converting a Keras or Tensorflow model to a Tensorflow lite micro model is really simple. To save memory usage, the models that are used are quantized.

3.2 Pytorch

For plant disease recognition, PyTorch, a popular deep learning framework, may easily interface with Edge Impulse using the Open Neural Network Exchange (ONNX) standard. PyTorch offers a dynamic and flexible computational graph, which is ideal for building and perfecting intricate neural network configurations. The platform is made to work with ONNX models, thus switching from PyTorch to Edge Impulse will go smoothly. Through the use of ONNX, we can integrate Edge Impulse's deployment capabilities with PyTorch's model training strengths to build a reliable and effective system for plant disease detection on edge devices.

3.3 Flutter

Farmers can access cutting-edge agricultural tools thanks to Flutter, a flexible user interface toolkit, which easily integrates pre-trained models in mobile apps. With the help of plugins and model integration, Flutter makes it possible for smartphones to diagnose diseases in real time. This facilitates scalability, lowers costs, and provides farmers with timely information. Farmers may access the interface more easily, which promotes accessibility and effective crop management decision-making.

IV. DATASET

We used 15 different classes of plant leaves and background images available with PlantVillage data-set for model development. Some of the plant leaves images are shown in Fig. 1. The model in question was trained utilizing various training epochs, batch sizes, and dropouts. In comparison to widely-used transfer learning methods, the proposed model attains superior performance by employing the validation data. The data-set contains 61,486 images. We used six different augmentation techniques for increasing the dataset size. The techniques are image flipping, Gamma correction, noise injection, PCA color augmentation, rotation, and Scaling. The train and test data used for the model development are shown in Table I.

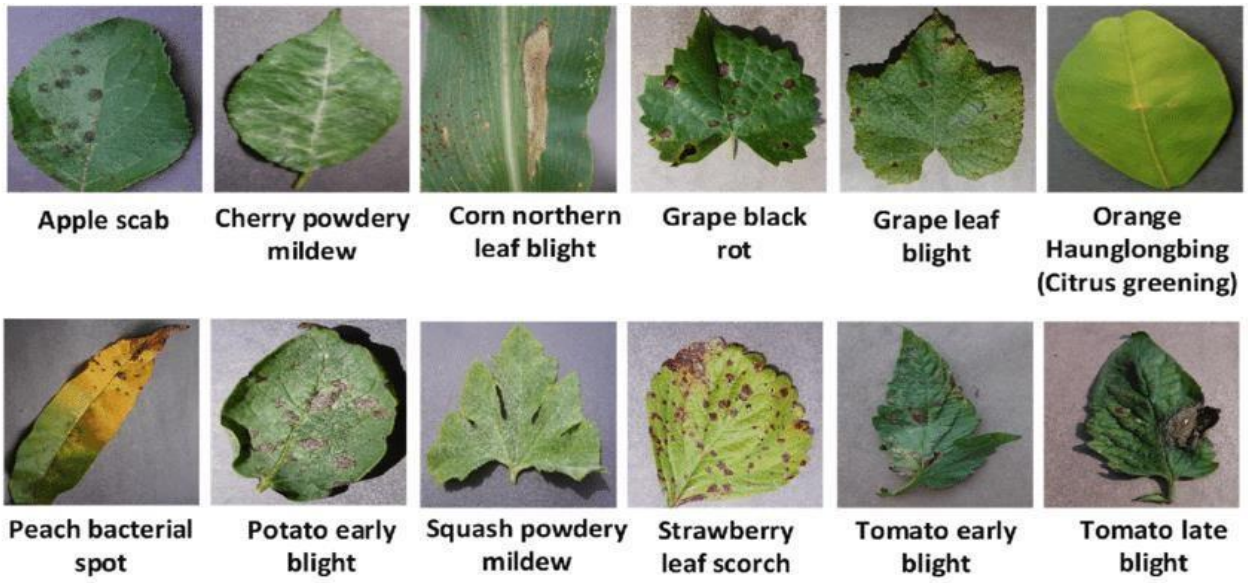

Fig 1: Dataset

TABLE I: Train and Test dataset flow type breakdown

S.No.	Classes	Training Set	Cleaned Training Set	Test Set	Cleaned Test Set
0	Apple	3717	3577	1289	1190
1	Blueberry	1238	1228	355	354
2	Cherry	80	65	60	10
3	Corn	4484	4480	1861	1861
4	Grape	3711	3700	1022	1022
5	Orange	3464	3302	944	944
6	Peach	3651	3411	1048	1048
7	Pepper	5000	3819	1436	1436
8	Potato	3000	2839	937	922
9	Raspberry	927	927	836	626
10	Soybean	100	66	28	28
11	Squash	3714	2086	1067	1067
12	Strawberry	949	935	272	272
13	Tomato	4200	3402	5200	5174
14	Random Leaf Images - Noise	1410	910	197	187

V. ML MODEL

Custom Machine Learning Models

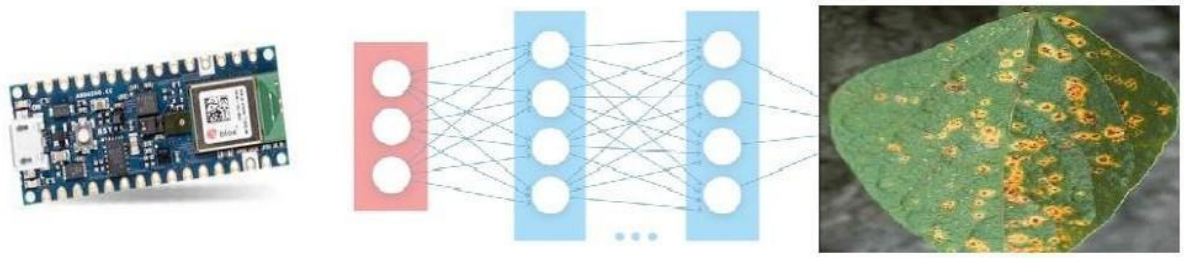

Fig 2: Model Architecture

Mobilenet V2 architecture was chosen due to its simplicity and easily deployment with resource constrained devices. The image size used is 224x224. Seven hidden layers are used to get better training accuracy. Arduino Nano 33 BLE sense, a microcontroller-based edge device works with TFLM (Tensorflow lite for microcontrollers), was used as the target device supported by ARM Cortex M4, running at 64 MHz speed for better computational power. The Custom model which was trained, the model parameters shown below during training and testing and got the performance on Train Accuracy: 96.7**,** Test Accuracy: 98.9**,** Validation Accuracy: 98.7.

```
----------------------------------------------------------------
        Layer (type)               Output Shape          Param #
================================================================
            Conv2d-1          [-1, 32, 224, 224]             896
              ReLU-2          [-1, 32, 224, 224]               0
       BatchNorm2d-3          [-1, 32, 224, 224]              64
            Conv2d-4          [-1, 32, 224, 224]           9,248
              ReLU-5          [-1, 32, 224, 224]               0
       BatchNorm2d-6          [-1, 32, 224, 224]              64
         MaxPool2d-7          [-1, 32, 112, 112]               0
            Conv2d-8          [-1, 64, 112, 112]          18,496
              ReLU-9          [-1, 64, 112, 112]               0
      BatchNorm2d-10          [-1, 64, 112, 112]             128
           Conv2d-11          [-1, 64, 112, 112]          36,928
             ReLU-12          [-1, 64, 112, 112]               0
      BatchNorm2d-13          [-1, 64, 112, 112]             128
        MaxPool2d-14            [-1, 64, 56, 56]               0
           Conv2d-15           [-1, 128, 56, 56]          73,856
             ReLU-16           [-1, 128, 56, 56]               0
      BatchNorm2d-17           [-1, 128, 56, 56]             256
           Conv2d-18           [-1, 128, 56, 56]         147,584
             ReLU-19           [-1, 128, 56, 56]               0
      BatchNorm2d-20           [-1, 128, 56, 56]             256
        MaxPool2d-21           [-1, 128, 28, 28]               0
           Conv2d-22           [-1, 256, 28, 28]         295,168
             ReLU-23           [-1, 256, 28, 28]               0
      BatchNorm2d-24           [-1, 256, 28, 28]             512
           Conv2d-25           [-1, 256, 28, 28]         590,080
             ReLU-26           [-1, 256, 28, 28]               0
      BatchNorm2d-27           [-1, 256, 28, 28]             512
        MaxPool2d-28           [-1, 256, 14, 14]               0
          Dropout-29                 [-1, 50176]               0
           Linear-30                  [-1, 1024]      51,381,248
             ReLU-31                  [-1, 1024]               0
          Dropout-32                  [-1, 1024]               0
           Linear-33                    [-1, 39]          39,975
================================================================
Total params: 52,595,399
Trainable params: 52,595,399
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 143.96
Params size (MB): 200.64
Estimated Total Size (MB): 345.17
```

TABLE II: Performance Metrics Description

Metric	Equation	Explanation
Accuracy	(T P + T N)/(T P + F P + T N + F N)	Percentage of correctly classified instances
Precision	(T P/(T P + F P))	Percentage of positive class predictions, that are indeed positive
Recall	(T P/(T P + F N))	Measures how well the model can correctly identify instances of positive class
F1-score	(2 ∗ Precision ∗ Recall)/(P recision + Recall)	A single score the balance Precision and Recall together
False Positive Rate	(F P/(F P + T N))	Probability of a false alarm

Table II shows a description of all the metrics used to evaluate the performance of an ML model. When dealing with imbalanced data, the accuracy measurement cannot reflect the true performance of an ML technique. A simple classifier can predict all samples to be in the majority class, thus achieving a high accuracy model, but limited to classify instances belonging to minority classes leading to a restricted usage of overall model. Metrics such as precision, recall and F1-score were used to better capture the ability of the classifier.

Results and Discussion:

The developed model is trained using Edge Studio and the Plant village datasets. The developed model is deployed in to a smart phone using the flutter interface tool kit. The snapshot of the UI for plant disease detection is shown in Fig. 3, for peach leaves. Similarly, the accuracy, loss, F1 score, ram requirements, inference time for the different crop leaves viz Grape, Apple, Corn, and Potato were shown in Fig. 4 to 7 respectively. Different diseases were correctly classified according to the feature extracted by convolution layers of the model.

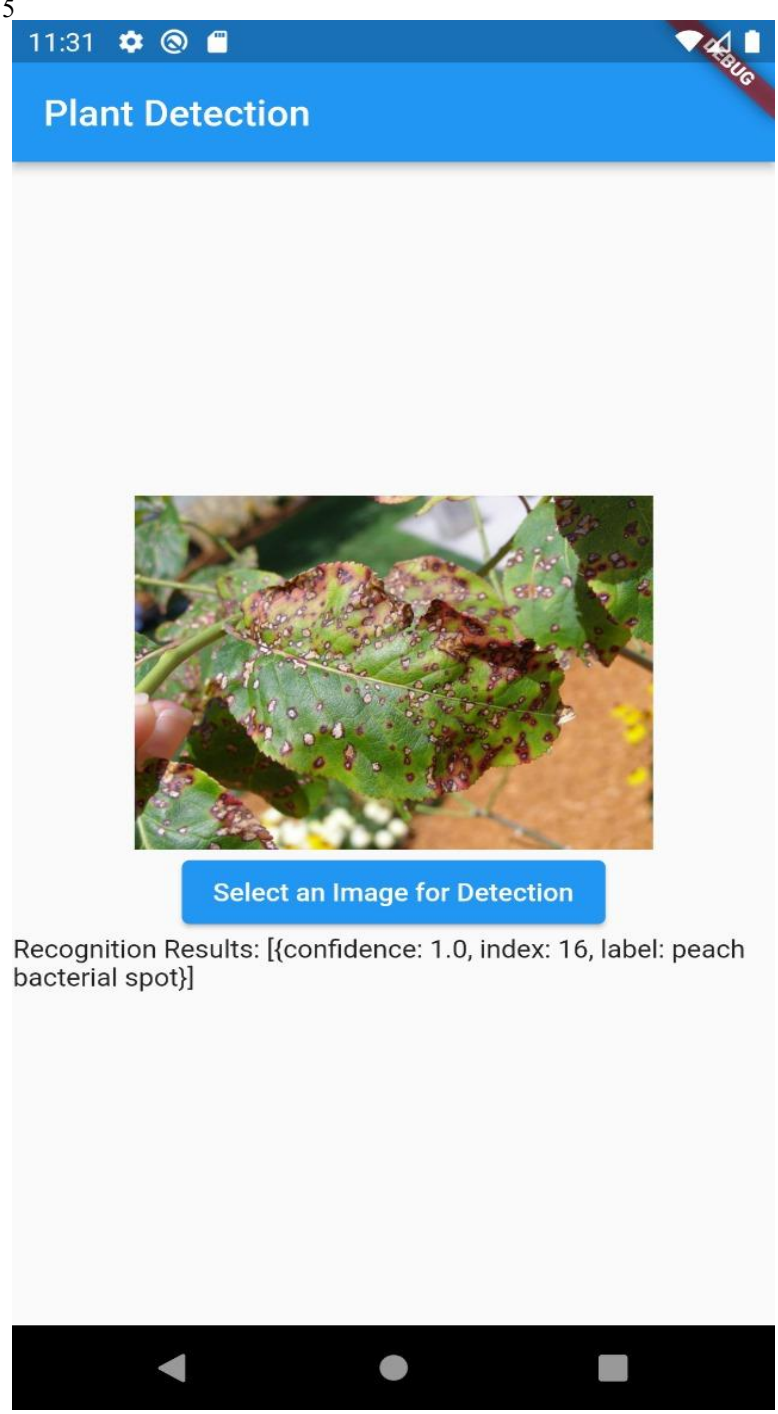

Fig 3. Snapshot of the model deployed in mobile application for peach plant leaves

Fig 4. Snapshot of performance measures for Grapes

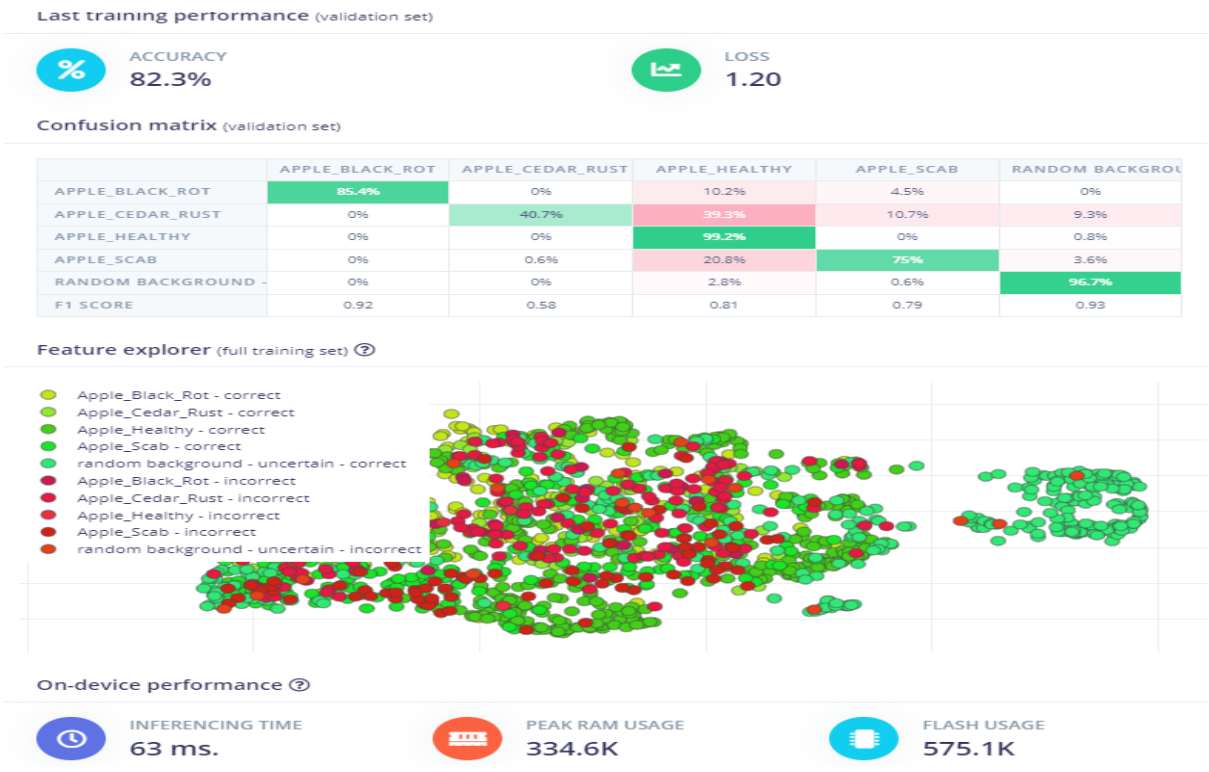

Fig 5. Snapshot of performance measures for Apple

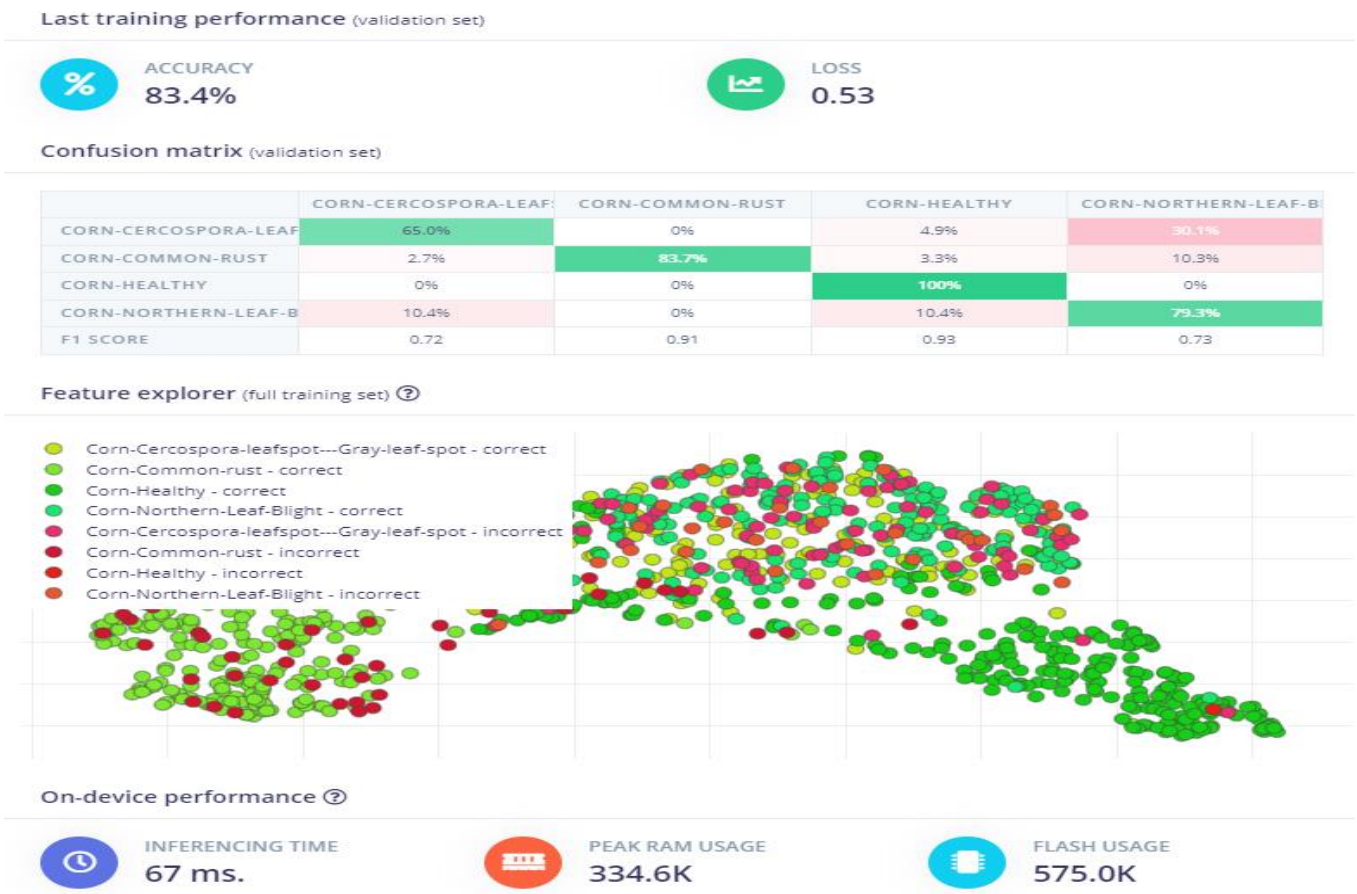

Fig 6. Snapshot of performance measures for Corn

Fig 7. Snapshot of performance measures for potato

The OV7675 camera of the TinyML development kit was used for testing the results shown in Fig 4 to Fig 7. The model's deployment in Arduino Nano 33 BLE sense MCU represented a successful application with encouraging results. Still, the major goal is to improve the model to make it work in resource constrained MCUs. To help in that aspect optimization was carried out using EON tuner and Table III presents the optimal performance achieved when models were trained with the EON Turner.

Model Benchmarks

Crop	rgb-mobilenetv2-24b		grayscale-mobilenetv2-1ac		grayscale-conv2d-e98	
	DSP	NN	DSP	NN	DSP	NN
Corn - 96% Accuracy	Latency: 3ms | RAM: 4Kb	Latency: 160ms | RAM: 200Kb	Latency: 11ms | RAM: 31Kb	Latency: 50ms | RAM: 200Kb	Latency: 80ms | RAM: 36Kb	Latency: 221ms | RAM: 280Kb
Apple - 93% Accuracy	Latency: 69ms | RAM: 146Kb	Latency: 196ms | RAM: 300Kb	Latency: 60ms | RAM: 132Kb	Latency: 114ms | RAM: 200Kb	Latency: 190ms | RAM: 240Kb	Latency: 160ms | RAM: 530Kb
Potato - 94% Accuracy	Latency: 44ms | RAM: 40Kb	Latency: 194ms | RAM: 200Kb	Latency: 170ms | RAM: 200Kb	Latency: 460ms | RAM: 520Kb	Latency: 160ms | RAM: 200Kb	Latency: 160ms | RAM: 200Kb
Grape - 96% Accuracy	Latency: 143ms | RAM: 484Kb	Latency: 141ms | RAM: 813Kb	Latency: 211ms | RAM: 363Kb	Latency: 290ms | RAM: 860Kb	Latency: 126ms | RAM: 210Kb	Latency: 260ms | RAM: 340Kb

The intended result is to provide farmers with an easy-to-use application that enables quick and easy disease diagnostics on their smartphones using trained models. By expanding access to leading-edge agricultural technology, the move aims to allow all farmers, regardless of their knowledge of agriculture, to take advantage of the model's potential for efficient crop disease diagnosis and management.

VI. Future Work & Conclusions

In a nutshell, this study used Edge Impulse to train models for agricultural disease detection, and significant classification of diseases was tested for four essential crops: potato, corn, apple and grape. A Flutter-based application has been seamlessly integrated with the custom model that was generated with the ability to identify and diagnose crop illnesses. This mobile application has the potential to assist the farmers in early pest detection and management. Additionally, the use of two different models, one designed for mobile devices and the other specifically for Tiny microcontrollers, brings the utility to handheld devices, such as drones, enabling effective scouting and manual evaluation by farmers.