Poster Abstract: Embedded ML Pipeline for Precision Agriculture

Dhruv Sheth
Edge Impulse, San Jose, California, USA
dhruvsheth.linkit@gmail.com

Bharath Sudharsan
Data Science Institute, NUI Galway, Ireland
b.sudharsan1@nuigalway.ie

John G. Breslin
Data Science Institute, NUI Galway, Ireland
john.breslin@nuigalway.ie

Muhammad Intizar Ali
School of Electronic Engineering, DCU, Ireland
ali.intizar@dcu.ie

ABSTRACT

Invariable of the agriculture type (precision, smart, or digital), the monitoring process of factors that increase the crop yield and growth is mostly non-ML, manually structured approaches with practical pain points. In this scenario, to reduce monitoring costs and maintenance efforts, there is a requirement for low-cost semi-autonomous distributed systems that can remotely collect plant data and perform standalone ML-based analytics without depending on cloud servers or the internet.

In this work, we provide an embedded ML pipeline, which users can use/follow for end-to-end solution design and implementation for any of their use-cases. To demonstrate the pipeline, we use it to collect image data, train a CNN-based regression algorithm, perform hardware-specific tuning, generate optimized code, and deploy binaries on Sony Spresense setup. The initial testing shows that even the resource-constrained MCU-based Spresense, in real-time (992 ms), high performance (96.2% accuracy, 1.86 cm^2 RMSE), could analyze a plant in a semi-autonomous environment to predict the leaf area and plant growth.

KEYWORDS

IoT Devices, Precision Agriculture, TinyML, Optimization.

1 INTRODUCTION

Advancements in ML and IoT hardware [1] are improving the ability of scientists to profile plants for understanding their growth under dynamic and unpredictable conditions. As a result, farmers can make rapid progress in plant selection and trait analysis, thus accelerating crop yield [2]. The commercial ML-based profiling and analytics solutions have the least customizability, and the upfront investment limits the farmers from using them during plants breeding and growing stages. Also, when surveying the open-source solutions and studies, most of the reliable ML algorithms require intensive computation and expensive hardware for standalone execution. In agricultural research, Leaf Area Index (LAI) is a potential index due to its direct connection with the crop's health condition and growth state. As summarized, since ML-based solutions are not yet widely accessible, plant scientists and farmers follow the manual, invasive method for accurate LAI calculation that relies on plant deconstruction.

In this paper, we present our embedded ML pipeline, using which we show the users how to design a cost-effective end-to-end (from data collection till hardware deployment) approach for non-invasive (leaving the crops intact) semi-autonomous accurate plant LAI estimation and growth prediction. There is a myriad of applications of this work - e.g., using our pipeline-designed solution to remotely monitor, predict, and alert any decrease in LAI of plants over time. As the designed solution is also optimized using the pipeline, it can fit within the memory of low-cost IoT hardware thus can be deployed in multiple points across the farm. This work also aims to contribute to the scope of the United Nations (UNs) second sustainable development goals to *increase productivity and production by implementing resilient agricultural practice.*

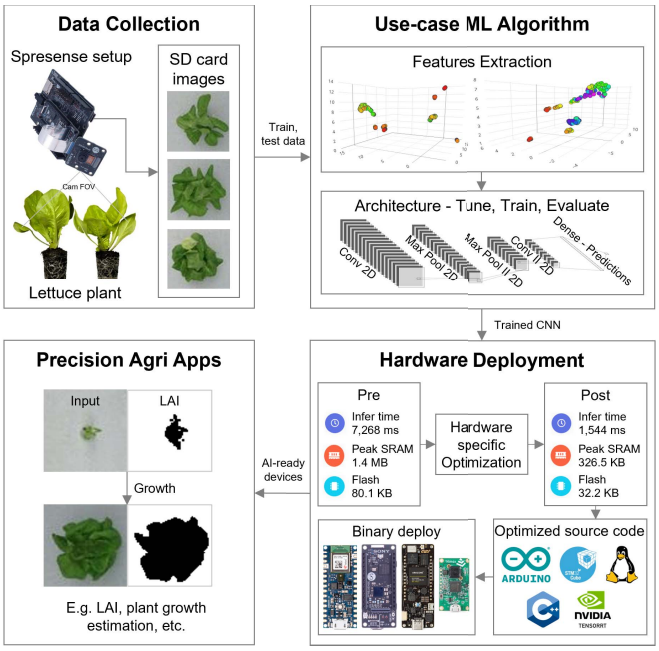

Figure 1: Components, design flow of embedded ML pipeline.

2 EMBEDDED ML PIPELINE DESIGN

Figure 1 presents our embedded ML pipeline which researchers and engineers can follow to design end-to-end ML-based solutions for agriculture use-cases and execute it on resource-constrained IoT hardware (MCUs, small CPUs, AIoT boards) for deployment in labs, greenhouses, farms, gardens, etc. Each pipeline component is briefly described in the upcoming subsections.

2.1 Data Collection

Numerous datasets exist to train ML algorithms for agriculture use-cases. Still, data needs to be captured for uncommon plants whose dataset is not available - researchers cannot have built datasets for millions of existing plants. We present best practices for collecting plant data: (i) Images can contain excess background pixels, which needs cropping followed by uniform adjustment (we set to 96x96 pixel resolution); (ii) It is crucial to capture depth (from camera to plant) along with RGB data to record the vertical growth - camera need to be set in same height throughout (we set to 78 cm from ground); (iii) Data augmentation can be used to enlarge the training

978-1-6654-9624-7/22/$31.00 ©2022 IEEE
DOI 10.1109/IPSN54338.2022.00064

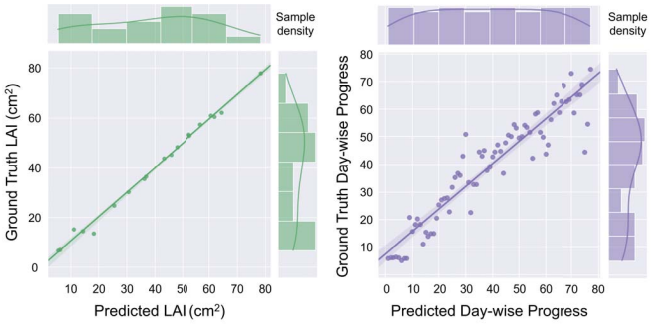

Figure 2: Ground truth vs predictions for the trained CNNs.

dataset - we rotated images by 90°, 180°, 270°, also flipped horizontally and vertically; (iv) To adapt the ML algorithm to dynamic illuminations, images need to be converted to the HSV color space, then brightness of the images need to be adjusted by altering the V channel - we adjusted the brightness to 0.8, 0.9, 1.1, 1.2 times that of the original images to simulate the change in daylight; (v) Add translational blur parameter (we added 0.01) to enable accurate predictions even under motion; (vi) Camera angles need to be placed vertically perpendicular with respect to the ground plane for each plant to ensure standardized images and deviation in LAI increase and Leaf growth is in constant incrementation. In this work, we collected data using a 5.11 MP Sony camera. Other camera-based boards (such as ESP32-CAM, Sipeed MAIX Bit, M5 StickV AI Camera, OpenMV Cam H7) can be used with image pre-processing to match the input requirements of the ML algorithm.

2.2 Use-case ML Algorithm

Training and Tuning. Figure 1, presents architecture of the designed CNN. Here, each convolution layer contains a 3x3 kernel to extract features. The Max Pooling layers have 2x2 kernels, stride of 2. This CNN was trained for two use-cases. For the first, LAI (decimal numbers with cm^2 unit) were provided as labels. $LAI_{perplant} = \sum_1^n 2.65 * 10^{-2} cm^2$ with n as max pixel area under Otsu's Segmentation. For the second, the day-wise labels (unitless integral values) corresponding to growth stages were provided. Models were trained for both the use-cases with a dropout of rate 0.25, batch size 16, constant learning rate 0.002. The network was trained till 225 epochs, beyond which loss function elevated.

Evaluation. The resultant models when fed with pre-processed (see below), 3 Channel, 96x96 images of greenhouse lettuce, the first can estimate the LAI, and the second can predict the day-wise progression. The evaluation of both models is given in Figure 2 as relational plots. The average RMSE for both models were low as ≈ 1.86 cm^2, or in other words, the predicted data is on an average ≈ 0.23 cm^2 less than Ground Truth. Based on the hardware target, other network architectures, tuning, hyperparameter optimization for the same tasks can be explored via AutoML using tools such as Ray Tune, Optuna, EON Tuner, Hyperopt, Scikit-Optimize.

Image Processing. The captured plant sample for inference using thus trained models need pre-processing before feature extraction. We initially use Otsu's adaptive thresholding mechanism to segment plants from the background. For small leaves, the Otsu threshold segments the image leaving some noise at the periphery of the confined region, impacting the prediction accuracy. Hence, for such images with plant area less than the average area, a Floodfill algorithm is used to binarize the noise or holes in the image.

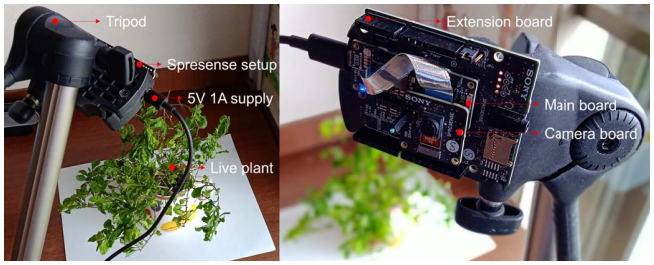

Figure 3: Real-world testing: Inference on Sony Spresense.

2.3 Hardware Deployment & Real-world Testing

In the *hardware-specific optimization* step, significant resource conservation can be achieved while maintaining consistent accuracy and loss scores. For our CNN, the RAM usage decreased ≈ 74 % (1.4 MB to 362.5 KB), Flash usage decreased ≈ 59% (80.1 KB to 32.2 KB), and ≈ 4.7x inference speedup (7268 ms to 1544 ms). Our CNN was optimized using `tfmot` [3]. In the next *optimized source code and binary deployment* steps, the tuned model is built and flashed on the setup shown in Figure 3. We report that the ARM Cortex-M4F MCU on Spresense, consistent for 100s of collected samples, processed the captured 5.11 M pixels plant images and provided close to real-time inference results in less than 1 ms. For deployment on more constrained MCUs [4] or on bare-metal, end-to-end optimization can be performed using `microTVM` [5].

3 CONCLUSION & FUTURE WORK

We believe the transparent embedded ML pipeline design presented in this work to open future avenues for a broad spectrum of semi-autonomous precision agriculture solutions. With slight alterations, this pipeline is also suitable for designing ML solutions using IoT environment sensors that measure CO_2, soil moisture & volumetric water content, light intensity, atmosphere temperature & humidity, etc. Future work plans to: (i) Present the design flow and each pipeline component in detail, particularly the CNN AutoML tuning and optimized source code generation; (ii) Evaluate the pipeline by using it to design and deploy novel solutions that relieve practical pain points in state-of-the-art crop monitoring approaches.

ACKNOWLEDGEMENT

This publication has emanated from research supported in part by research grants from Science Foundation Ireland (SFI) under Grant Number 16/RC/3918 (Confirm), 12/RC/2289_P2 (Insight) and 16/RC/3835 (VistaMilk), with all grants co-funded by the European Regional Development Fund.

REFERENCES

[1] B. Sudharsan, P. Patel, J. G. Breslin, and M. I. Ali. Sram optimized porting and execution of machine learning classifiers on mcu-based iot devices: demo abstract. In *ACM/IEEE 12th ICCPS*, 2021.
[2] L. Zhang et al. Growth monitoring of greenhouse lettuce based on a convolutional neural network. In *Oxford Academic, Horticulture Research*, 2020.
[3] Tfmot python api: https://github.com/tensorflow/model-optimization/, 2021.
[4] B. Sudharsan, J. G. Breslin, and M. I. Ali. Ml-mcu: a framework to train ml classifiers on mcu-based iot edge devices. In *IEEE IoT Journal*, 2021.
[5] Apache tvm: https://github.com/apache/tvm/, 2021.