A Benchmark of Deep Learning Models for Multi-leaf Diseases for Edge Devices

Pham Tuan Anh* and Hoang Trong Minh Duc**

*Posts and Telecommunications Institute of Technology, Hanoi, Viet Nam
**SET, Hanoi University of Science & Technology, Hanoi, Viet Nam, IEEE student member
Email: anhpt.b17vt021@stu.ptit.edu.vn, DUC.HTM182430@sis.hust.edu.vn

***Abstract*—Every season, leaf diseases are one of the main reasons affecting the production of many crops, which cause enormous damage to farmers. To minimize the loss, deep learning techniques are utilized to detect leaf infection and wildly outperformed the traditional method of manual detection. However, deploying such models is a challenge since devices in the field normally have limited resources and low computational power while large datasets have to be used. Therefore, in this paper, we benchmark the most popular deep learning models for multi-leaf disease detection to gauge which model is the most suitable for real deployment. Using a real-world large-scale dataset from PlantVillage and a Raspberry Pi 3, we found that MobileNet V3 provides a reliable accuracy of 96.58%, small Inference/Initialization time of 127 ms and 11 ms respectively, requires only 7.4 MB of memory in total, and hence the most appropriate choice for a real farm.**

***Index Terms*—deep learning, edge computing, precision agriculture, multi-leaf disease, image processing**

I. Introduction

In the last decade, climate change has not only impacted directly the masdeep learning models and quality of agricultural yields but also increased the intensity and complexity of crop disease. To counter that, smart farming has been developing rapidly. Incorporate with many advanced technologies, smart agriculture able to monitor environmental conditions and develops schemes to archive optimum health and yield of the crops while saving natural resources. Particularly, with computer vision, smart farming helps farmers quickly identify and accurately provide solutions for many plant diseases [1].
Thank to the success of smart farming, the number of Internet of Things (IoT) devices skyrocket. However, the widespread use of agricultural IoT has preceded the explosive growth of sensors and growing amounts of data that increase the load on the cloud server. Edge computing can solve this challenge by handling several parts of a task at the local computer nearby the end-users. The edge devices have low-cost and limited resources that can be used with great efficiency for some specific tasks [2].
AI is applied not only in medicine, transportation, security but also in agriculture thanks to image recognition. In general, implementing systems based on AI solutions often requires large resources of data as well as computing power. Therefore, most of these models usually operate on high-powered devices at the center. Incorporating AI at the edge has been proposed recently to promote the strengths of edge computing, called edge intelligence. However, the development of edge intelligence systems faces several challenges, and one of these key challenges is that the computing power of edge devices cannot meet the same resource demands as the normal center devices [2] [3]. Nevertheless, Edge intelligence system is usually deployed in agricultural areas where connections are not secure, latency is large and varied, memory capacity and power are not guaranteed. This shows the need of promoting the research on Computer Vision to be more and more optimized.
Due to several mobile DNN models for detecting plant diseases that have been proposed in recent years, they can bring lightweight resource consumers that can be applied to edge computing devices. However, to the best of our knowledge, there is currently no study to benchmark the state-of-art deep learning models in multiple types of plant databases. We will fill the gap and evaluate these models in a practical edge device to estimate the ability of deploying to a real IoT agriculture system. The organization of the paper as follows. Section II outlines the related works which concern our proposal. Section III provides the main principles of modern deep learning models. Section IV focuses on our works with experimental results. Last but not least, in section V, we will discuss scaling down depth of models for smaller edge device/mobile to processing for low-quality images such as those were taken from drones with limited computational resources. The conclusion and our future work will be shown in the last section.

II. Related work

Currently, artificial intelligence (AI) applications are growing strongly with breakthroughs in deep learning and many innovations in architectural hardware. Since then, there has been a prevailing trend to integrate Edge Computing and AI to create Edge Intelligence to effectively handle a large amount of IoT data. Edge intelligence leverages the available data and resources of end devices, edge nodes, and cloud centers to optimize the total training and reasoning performance of the deep learning model. AI is primarily employed in video analysis, unmanned agricultural machinery, pest identification, and plant species identification [3] [4]. Along with the development of deep learning architectures, researchers applied them to image recognition and classification. These architectures have also been implemented for different agricultural applications. On top of that, deep learning (DL) approaches are also used for

critical tasks like plant disease detection and classification, which is the main focus of the review in [4]. However, in [4] the authors used data from only one specific plant to compare between different DL models.

To detect plant diseases and pests based on deep learning, the authors in [3] provided a novel classification of detection methods based on deep learning and analyzed the main aspects of 8 DNN models based on classification network, detection network, and segmentation network. The study used several common datasets to find out the advantages and disadvantages of each method. However, this proposal does not concern with the efficiency of a practical image recognition method.

To classify tomato plant diseases, the authors in [5] focused on fine-tuning based on the comparison of the state-of-the-art architectures: AlexNet, GoogleNet, Inception V3, ResNet 18, and ResNet 50. The dataset used for the experiments is contained by nine different classes of tomato diseases and a healthy class from PlantVillage. An evaluation of the comparison was finally performed on several performance metrics such as accuracy, precision, sensitivity, specificity, F-Score, Area Under the Curve (AUC), and receiving operating characteristic (ROC) curve.

In [6], the authors propose a multi-plant disease diagnosis method using the deep learning technique. In which, the author surveyed that most of the techniques are plant-specific or disease-specific. They propose a novel multi-label classification for leaf images of six plants, including tomato, potato, rice, corn, grape, and apple in various online datasets. The performance of DL models is compared by precision, recall, and F1-score metrics. There is no efficient computation concerned with the experimental results.

Intending to find a solution using deep learning suitable for edge computing to serve the problem of disease plant diagnosis of many crops in practice, the above studies have not mentioned the computational performance parameters of deep learning models on resource-constrained edge devices. Therefore, in this paper, we will conduct experiments and evaluate some of the most modern deep learning models on a multi-tree dataset to suggest the appropriate approach to real agricultural IoT systems.

III. State-of-art Deep Learning Models

With the goal of focusing on low-power edge devices, we survey and evaluate some of the most modern and lightweight models available today. This session will briefly introduce the features of the models on which we will perform performance evaluations in the following section. In detailed, we evaluate the applicability of deep learning technique for some critical aspects described above for detecting leaf disease at the edge computing devices of agricultural IoT systems. In this study, we include several Models that improve the hardware efficiency such as MobileNets, MnasNet, and EfficentNets Lite, and popular DNN models like InceptionNets, ResNets.

1) InceptionNets: The GoogLeNet variation with Inception modules was introduced in 2016 [7]. Inception-v3 has achieved good classification performance in several biomedical applications using transfer learning. It proposed an inception model which concatenates multiple different sized convolutional filters into a new filter. The goal of the inception module is to act as a "multi-level feature extractor" by computing 1×1, 3×3, and 5×5 convolutions within the same module of the network as illustrated in Figure 1. Such design decreases the number of parameters to be trained and thereby reduces the computational complexity.

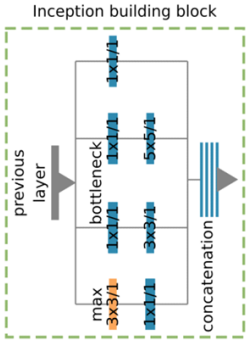

Fig. 1. The Inception building block [8]

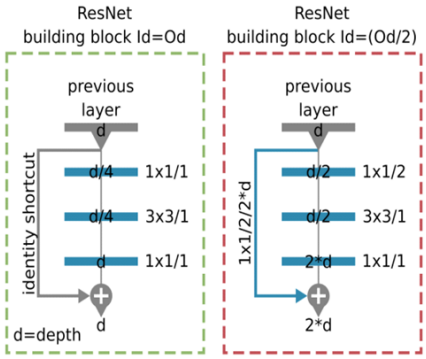

Fig. 2. Conceptual overview of the ResNet building block [8]

2) ResNets: The ResNet models, which are based on deep architectures that have shown good convergence behaviors and compelling accuracy, were developed by He et al. [7]. ResNet was built by several stacked residual units and developed with many different numbers of layers: 18, 34, 50, 101, 152, and 1202. The residual units are composed of convolutional, pooling, and layers as shown in figure 2. ResNet 50 contains 49 convolutional layers and a fully connected layer at the end of the network. To saving computing resources and training time, ResNet 50 was chosen for the comparison in later section.

3) MobileNets: There are three versions of MobileNets, the latest, MobileNet V3. The core architecture of MobileNetV1 is based on a streamlined architecture that uses depth-wise separable convolutions to build lightweight deep neural networks [9]. MobileNetV2 introduced two new features to the architecture: linear bottlenecks between the layers and shortcut connections between the bottlenecks [10]. MobileNetV3 [11] is the third version of the architecture (figure 3), powering the image analysis capabilities of many popular mobile applica-

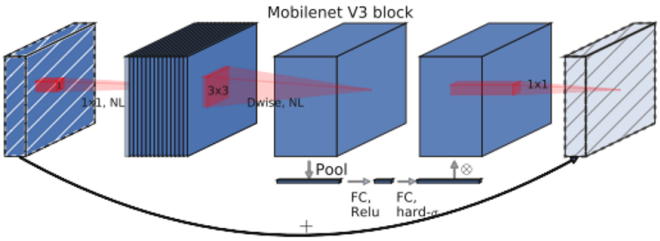

Fig. 3. MobileNet V3 building block [12]

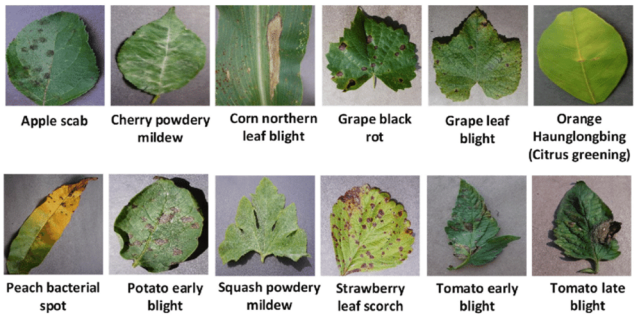

Fig. 4. Some of the plant diseases from the PlantVillage dataset [19]

tions. The main contribution of MobileNetV3 is the use of AutoML to find the best possible neural network architecture for a given problem. This contrasts with the hand-crafted design of previous versions of the architecture.

4) MnasNet: The main building block of MnasNet [13] is an inverted residual block (from the above-mention MobileNet V2). Inspired by recent progress in AutoML neural architecture search, the MnasNet architecture search approach for designing mobile models using reinforcement learning. The overall flow of the approach consists mainly of three components: an RNN-based controller for learning and sampling model architectures, a trainer that builds and trains models to obtain accuracy, and an inference engine for measuring the model speed on real mobile phones. MnasNet does a multi-objective optimization problem that aims to achieve both high accuracy and high speed.

5) EfficientNets Lite: EfficientNet-Lite [14] brings the power of EfficientNet[15] to edge devices and comes in five variants, allowing users to choose from the low latency/model size option (EfficientNet-Lite0) to the high accuracy option (EfficientNet-Lite4). Some of the operations in EffcientNet are not well supported by certain accelerators. To address the heterogeneity issue, the original EfficientNets were tailored with the following simple modifications:

- Removed squeeze-and-excitation networks since they are not well supported.
- Replaced all swish activation with RELU6, which significantly improved the quality of post-training quantization.
- Fixed the stem and head while scaling models up to reduce the size and computations of scaled models.

IV. Performance Metrics and Models Evaluation

A. Benchmark environment

In our work, the performance of DL models was assessed on the Raspberry Pi 3 Model B [16]. Despite its limited resources, this low-cost embedded platform features enough computational power for real-time DNN inference. It has a Quad-Core ARM Cortex-A53 1.2GHz 64-bit CPU that can work at frequencies ranging from 700 MHz up to 1.2 GHz. Its instantaneous value depends on the policy set by the user and the operation conditions: CPU load, temperature, etc. The system incorporates 1GB RAM LPDDR2 at 900MHz. Likewise, to reduce the impact of the operating system on the performance, the booting process of the RPi was configured to prevent needless processes and services from being started. We also disconnected all peripherals during the characterization. The framework we use to train and evaluate these models is TensorFlow Lite [17]. TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and IoT devices. Optimized for on-device machine learning, by addressing 5 key constraints: latency (there's no round-trip to a server), privacy (no personal data leaves the device), connectivity (internet connectivity is not required), size (reduced model and binary size), and power consumption (efficient inference and a lack of network connections).

B. PlantVillage Dataset

The datasets we use in this work is PlantVillage [18] were gathered from Kaggle. The data records are available through the website www.plantvillage.org. The dataset consists of 54303 healthy and unhealthy leaf images divided into 38 categories by species and disease. The images span 14 crop species: Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, Tomato. It contains images of 17 fungal diseases, 4 bacterial diseases, 2 molds (oomycete) diseases, 2 viral diseases, and 1 disease caused by a mite. 12 crop species also have images of healthy leaves that are not visibly affected by a disease. Figure 4 below is an example of different phenotype plants.

C. Evaluation

For performance evaluation and comparison, we used Python as the common coding language for all of the DNN models. The Benchmark tool in this work is the TensorFlow Lite benchmark tool [20]. This tool currently measures and calculate statistics for the following important performance metrics:

- Initialization time
- Inference time of warm-up state
- Inference time of steady-state
- Memory usage during initialization time
- Overall memory usage

Moreover, the parameters we recorded in the training process are also given for evaluation:

- Accuracy

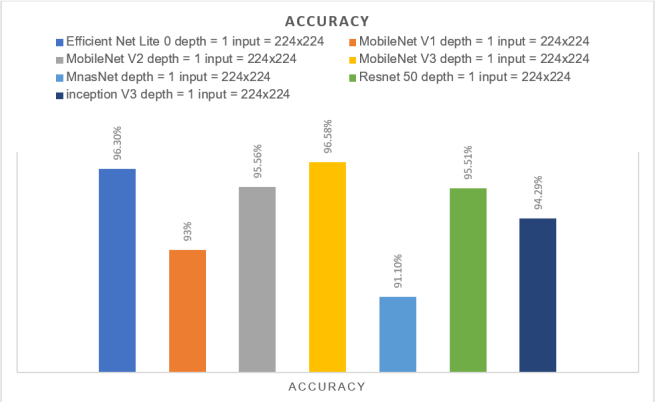

Fig. 5. Accuracy in comparison.

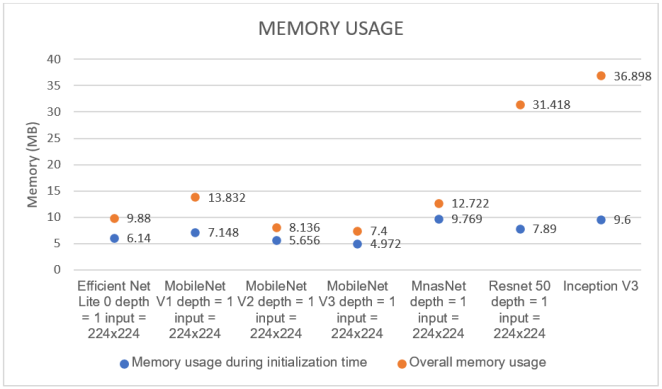

Fig. 6. Memory usage in comparison.

- Number of parameters
- Size of model
- Training time with GPU

The neural network architecture is implemented in TensorFlow Lite. For pre-training, we used ImageNet [21] weights for each of the models. The input shape of the leaf images is 256 × 256 × 3 and was resized into 224 x 224 x 3. In [22], [10], [11], [13], authors perform models as a function of different multipliers and resolutions. In their experiments, they have used multipliers 0.35, 0.5, 0.75, 1.0, and 1.25, with a fixed resolution of 224, and resolutions 96, 128, 160, 192, 224, and 256 with a fixed depth multiplier of 1.0. We evaluate all models with the depth = 1.0 and resolutions = 224 x 224, and further work in section V, we will discuss scaling down depth of models. We train our models using NVIDIA GEFORCE GTX 1080TI [23] with 11GB GDDR5X VRAM and 3584 CUDA Cores based on the NVIDIA Pascal – architecture.

Accuracy: To compare the result, we use Plant Village Dataset for all classification experiments and compare the accuracy versus various measures of resource usage such as latency, number of parameters, storage, and memory. The results are shown in figure 5. The results show that MobileNetV3 achieves the highest accuracy of 96.58%, 0.28% higher than the second place, EfficientNet Lite 0 (96.3%). The common point of these two networks is having the automatically designed Neural Architecture Search (NAS) [24], NetAdapt[25] algorithms, and inherit blocks with increasingly optimized structure for Edge/Mobile Device. Because of that, these two outperforming other hand-designed common network architectures such as InceptionNets 94.29%, ResNets 95.51%, or MobileNetV3's predecessors, MobileNetV2 and MobileNetV1 reach 95.56% and 93% respectively. Meanwhile, EfficientNets Lite uses the lightest EfficientNet B0 model and model scaling technique to scale down and find the variants of EfficientNets Lite while still achieving high accuracy and optimized FLOPS parameter. With MobileNetV1, it simply improves the classical convolution without many optimizations on network architecture. MobileNetV2 has improved with Linear Bottleneck Block and Residual Block [11] but designed manually and not guaranteed to be optimal. We also need to consider a significant offset of the search effort in the large/extremely large search space. With MnasNet, the results obtained in terms of accuracy are quite limited even though the network architecture is searched by NAS, but because the reward for the controller is not optimal in Multi-Objective. There is also no method to evaluate the most suitable model obtained from search results such as in NetAdapt.

Memory usage: In studies [26], [27], [28], [29], [30], Integer quantization is an optimization strategy that converts 32-bit floating-point numbers (such as weights and activation outputs) to the nearest 8-bit fixed-point numbers. This results in a smaller model and increased inferencing speed, which is valuable for low-power devices such as microcontrollers. Memory accesses contribute significantly to the energy consumption of DNNs [31] [32]. Therefore, after training DNNs models are optimized by parameter quantization, which will reduce hardware resources, memory usage, reduce computational costs, and consume less energy. To build low-power DNNs, recent research has investigated the trade-off between accuracy and the number of memory accesses. Figure 6 shows the memory using the models evaluated in the article. All models in the article are compressed as int8, the RAM usage of the models when compressed int8 is acceptable, with MobileNetV3 outperforming other architectures when using only 4,972MB in model initialization process and 7.4MB total in 1GB RAM of the RPi 3B. MobileNetV2 and EfficientNet Lite 0 also give a reliable result when they take up only 8,136MB and 9.88MB of total memory usage, respectively.

The number of parameters: Figure 7 shows the result of the number of parameters of each network. The MobileNetV3's number of parameters is the smallest compared to other network architectures (1.5 million), outperforming popular network architectures that are not optimized for Edge/Mobile such as ResNet 50 (23.6 million) and InceptionV3 (21.8 million).

Size of models: Parameters quantization can be used to reduce the size of the model. The smaller models have the following benefits:

- Smaller storage size: Smaller models take up less storage space on the user's device.
- Smaller download size: Smaller models require less time and bandwidth to download to the user's device.

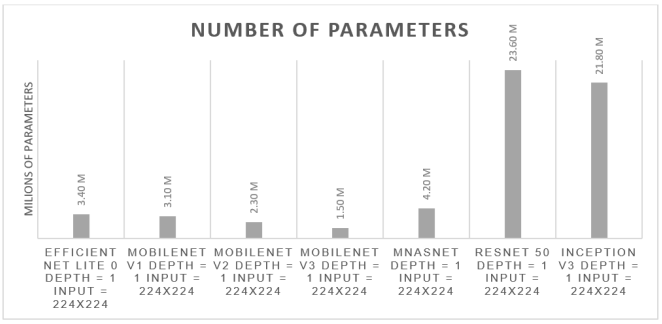

Fig. 7. Number of parameters in comparison in comparison.

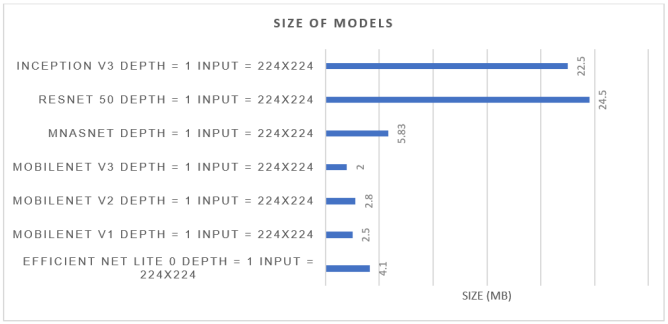

Fig. 8. Size of models.

- Less memory usage: Smaller models use less RAM as they are run, which frees up memory for use by other parts of your application and can translate into performance and stability better determination.

The results of the quantization get the size of the DNNs models as shown in figure 8. The models specially designed for Edge/Mobile are MobileNets, MnasNets, EfficientNets Lite, which are parameterized and optimized for memory, and the model size is better than the commonly used DNNs.

Inference time: Latency is the amount of time it takes to run a single inference with a given model. Some form of optimization can reduce the amount of computation needed to run the inference using the model, resulting in lower latency. Latency can also have an impact on power consumption. Quantization is a conversion technique that can reduce model size while also improving CPU and hardware accelerator latency, with little degradation in model accuracy. Currently, Model Quantization can be used to reduce latency by simplifying computations that occur during inference, potentially at the expense of some precision. The results below in figure 9, MobileNetV3 still gives the most satisfactory results.

Initialization time: The comparison result is demonstrated in figure 10.

Training time: The evaluated models are pre-trained on the standard ImageNet dataset and transfer learning to the PlantVillage dataset with a few dropout layers and dense layers added. From which we have quite fast training time results with GPU: GTX 1080TI 11GB as shown in figure 11 below.

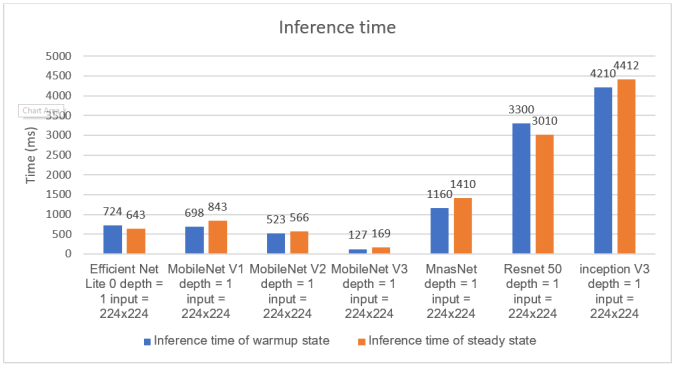

Fig. 9. Inference time in comparison.

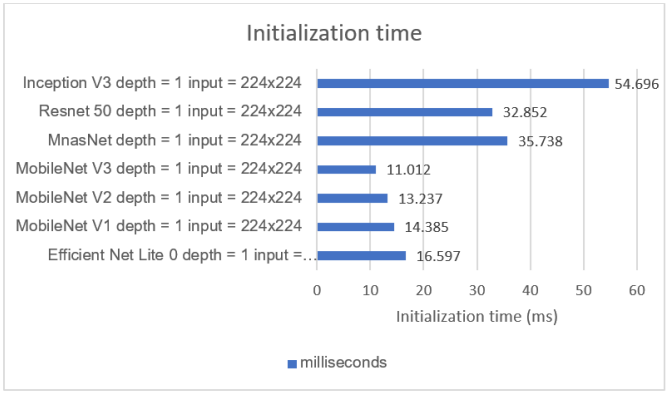

Fig. 10. Initialization time in comparison.

V. Conclution and Discussion

The presented benchmark is anticipated to serve as a preliminary reference when it comes to selecting DNN models among the broad ecosystem of DL components available. We have demonstrated that a cheap embedded computer like Raspberry Pi 3 Model B is capable of implementing real-time vision inference on the basis of complex DNN models. Thereby, this paper promoting new and more extensive research into integrating microcontrollers in the Edge Intelligence system. In section IV, a comparative analysis and performance evaluation was performed among SOTAs of modern deep learning architectures for edge/mobile devices. The performance of

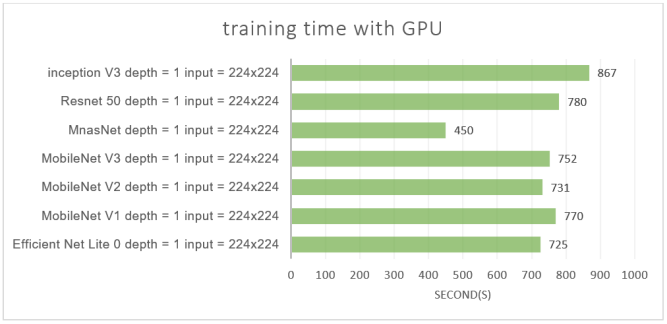

Fig. 11. Training time in comparison.

TABLE I
SCALE-DOWN MODEL

MobileNet V3	depth=1.0 input=224x224	depth=0.35 input=96x96
Accuracy	96.58%	94.4%
Memory usage (init/overall)	4.972 MB / 7.4 MB	4.464 MB / 5.644 MB
Time reference (init/overall)	0.127 s / 0.169 s	0.045 s / 0.043 s
Time init	11.012 ms	0.815 ms
Time training	752 s	166 s
Model size	2 MB	797 KB
Number of parameters	1.5 M	0.4 M

different models for each type of hardware may vary, so a model structure that is compatible with the hardware will result in better performance. According to our evaluation results, MobileNetV3 gives better results than the models evaluated with the pest data set on the Raspberry Pi 3 Model B. We also experiment with scaling down the depth of MobileNetV3 and reducing the resolution for smaller edge device/mobile to processing for a low-quality image and limited computational resources. The results are shown in Table 1. The scaled-down MobileNetV3 provides reliable accuracy of 94.4% while consumes an impressively small amount of resources, both time and memory.

In the future, we will expand our research further on Computer Vision and Machine Learning that can be deployed on microcontrollers. For practical implementation, we will also further evaluate energy usage, temperature, and stability under different environmental conditions in agriculture to empirically evaluate a stable system in future studies.

REFERENCES

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[2] J. Chen and X. Ran, "Deep learning with edge computing: A review." *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[3] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, "Bringing ai to edge: From deep learning's perspective," *arXiv preprint arXiv:2011.14808*, 2020.

[4] L. Li, S. Zhang, and B. Wang, "Plant disease detection and classification by deep learning—a review," *IEEE Access*, vol. 9, pp. 56 683–56 698, 2021.

[5] V. Maeda-Gutierrez, C. E. Galvan-Tejada, L. A. Zanella-Calzada, J. M. Celaya-Padilla, J. I. Galván-Tejada, H. Gamboa-Rosales, H. Luna-Garcia, R. Magallanes-Quintanar, C. A. Guerrero Mendez, and C. A. Olvera-Olvera, "Comparison of convolutional neural network architectures for classification of tomato plant diseases," *Applied Sciences*, vol. 10, no. 4, p. 1245, 2020.

[6] M. M. Kabir, A. Q. Ohi, and M. F. Mridha, "A multi-plant disease diagnosis method using convolutional neural network," in *Computer Vision and Machine Learning in Agriculture*. Springer, 2021, pp. 99–111.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[8] T. Hoeser and C. Kuenzer, "Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends," *Remote Sensing*, vol. 12, no. 10, p. 1667, 2020.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[12] "Programmersought.com. 2021. mobilenet v1,v2,v3 - programmer sought." https://www.programmersought.com/article/73408131416/, accessed: 2021-07-07.

[13] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[14] R. Liu, "Higher accuracy on vision models with efficientnet-lite," *TensorFlow Blog.[online] Available at: https://blog. tensorflow. org/2020/03/higher-accuracy-on-visionmodels-with-efficientnet-lite. html [Accessed 30 Apr. 2020]*, 2020.

[15] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[16] "Raspberrypi 2021," https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, accessed: 2021-06-28.

[17] "Tensorflow. 2021. tensorflow lite | ml for mobile and edge devices." https://www.tensorflow.org/lite, accessed: 2021-06-28.

[18] "Plantvillage.psu.edu. 2021. plantvillage." https://plantvillage.psu.edu/, accessed: 2021-06-28.

[19] M. H. Saleem, J. Potgieter, and K. M. Arif, "Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers," *Plants*, vol. 9, no. 10, p. 1319, 2020.

[20] "Github. 2021. tensorflow/tensorflow." https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/tools/benchmark, accessed: 2021-06-28.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[22] R. Liu, "Higher accuracy on vision models with efficientnet-lite," *TensorFlow Blog.[online] Available at: https://blog. tensorflow. org/2020/03/higher-accuracy-on-visionmodels-with-efficientnet-lite. html [Accessed 30 Apr. 2020]*, 2020.

[23] "Nvidia.com. 2021. geforce gtx 1080 ti | specifications | geforce." https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1080-ti/specifications, accessed: 2021-06-28.

[24] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[25] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "Netadapt: Platform-aware neural network adaptation for mobile applications," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 285–300.

[26] Y. Fan, W. Pang, and S. Lu, "Hfpq: deep neural network compression by hardware-friendly pruning-quantization," *Applied Intelligence*, pp. 1–13, 2021.

[27] P. Yin, S. Zhang, Y. Qi, and J. Xin, "Quantization and training of low bit-width convolutional neural networks for object detection," *arXiv preprint arXiv:1612.06052*, 2016.

[28] A. Goel, C. Tung, Y.-H. Lu, and G. K. Thiruvathukal, "A survey of methods for low-power deep learning and computer vision," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. IEEE, 2020, pp. 1–6.

[29] F. Zhu, R. Gong, F. Yu, X. Liu, Y. Wang, Z. Li, X. Yang, and J. Yan, "Towards unified int8 training for convolutional neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1969–1979.

[30] "Tensorflow lite 8-bit quantization specification, tensorflow, 2021," https://www.tensorflow.org/lite/performance/quantization_spec, accessed: 2021-06-29.

[31] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[32] R. Ding, Z. Liu, T.-W. Chin, D. Marculescu, and R. D. Blanton, "Flightnns: Lightweight quantized deep neural networks for fast and accurate inference," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.