Journal of Agriculture and Food Research 10 (2022) 100389

Contents lists available at ScienceDirect

Journal of Agriculture and Food Research

journal homepage: www.sciencedirect.com/journal/journal-of-agriculture-and-food-research

Comparative study on the performance of deep learning implementation in the edge computing: Case study on the plant leaf disease identification

Soo Jun Wei [a], Dimas Firmanda Al Riza [b,*], Hermawan Nugroho [a]

[a] Department of Electrical and Electronic Engineering, University of Nottingham in Malaysia, Selangor, Semenyih, 43500, Malaysia
[b] Department of Biosystems Engineering, Faculty of Agricultural Technology, University of Brawijaya, Jl. Veteran, 65145, Malang, Indonesia

ARTICLE INFO

Keywords:
Deep learning
Convolutional neural networks
Edge computing
Plant leaf disease

ABSTRACT

This research examines and explores four different pre-trained CNN deep learning models (AlexNet, VGG16, ResNet50, and DenseNet121) to be adopted as edge solution. The model is developed and evaluated using the PlantVillage dataset. Image transformation techniques and down sampling were carried out to mitigate the unbalanced class distribution problem. From the preliminary works, DenseNet121 was selected as the implementation model since it had the best accuracy (96.4%) in comparison with other models. The model is then tested on different endpoint devices (CPU, GPU and VPU) in different programming environments (standard PyTorch and OpenVINO) to test the consistency of models that work across different hardware and software configurations. Results show that the adoption of model on the edge (VPU) is able to maintain relatively high recall values, precision and F1 scores. The successful application of the model on VPU device shows the potential of the model to be implemented for the detection of plant diseases as edge solution/embedded application.

1. Introduction

Plant diseases, a deterioration in the original condition of the plant that upsets and changes its vital functions, are common problems that farmers and agronomists face from season to season. Various aspects can cause plant diseases, such as environmental conditions, the presence of pathogens, and the variety of crops located nearby the plant. In the initial period of most of the diseases, symptoms can be observed by looking at alterations of the physical properties of plants, such as alteration in color, size, and shape, due to the disease [1].

Plant diseases significantly disturb the progress of the plant and are recognized as one of the factors which affects food security. Consequently, fast, and accurate identification is essential to contain the spread of disease. However, the identification procedure is hampered by the complexity of the procedure. It is found that experienced agronomists and plant biologists find it difficulty difficult in differentiating certain plant diseases because of the multitude of symptoms that occur. The, where misdiagnosis leads to inadequate or inadequate treatment [2]. An automation system at this level will support and reduce the workload of farmers and agronomists [3].

Over the last few years, the exponential growth in computing power, particularly in the graphics processing unit (GPU), has resulted in machine learning algorithms being able to run efficiently on servers due to their ability to compute processes in parallel. Additionally, the introduction of GPU platforms such as Compute Unified Device Architecture (CUDA) from Nvidia has introduced a new platform for developers to accelerate applications which are compute intensive. This is accomplished by utilizing the computing power of GPU in a way that is optimized for computation of complex algorithms such as gradient descent and back-propagation and gradient descent [4]. Advances and breakthroughs in technology have enabled deep learning approaches to be widely used in many applications in nearly every field [5–8].

The deep learning model is created and based on combination of neural network architectures. In contrast to the conventional neural network architecture, which comprises of a few hidden layers only, a deep learning model may contain hundreds of hidden layers. This is one of the reasons which make deep learning, generally, can have higher accuracy in comparison with conventional methods. The Convolutional Neural Network (CNN) [9], which is characterized by its function for feature extraction in image recognition and image classification [10], is considered as one of the deep learning models. With different types of design, CNN model can conduct diverse types of operations on images.

Beside deep learning, researchers also worked on other machine learning algorithms in developing solutions for detecting plant diseases.

* Corresponding author.
E-mail address: dimasfirmanda@ub.ac.id (D.F. Al Riza).

https://doi.org/10.1016/j.jafr.2022.100389
Received 21 May 2022; Received in revised form 8 September 2022; Accepted 11 September 2022
Available online 21 September 2022

For instance, Yang and Guo [11] developed classification solution using SVM, Decision Tree, Random Forest and ANN (Artificial Neural Network) to classify plant diseases for citrus trees and tomato. The detection of plant diseases is carried out by identifying genes that are involved in interactions with plant pathogens. Daniya and Vigneshwari compared different types algorithms such as Faster R–CNN, CNN, PCA and SVM for the detection of rice diseases [12]. In addition, Ma et al. (Ma et al., 2018) run a comparative experiment using classifiers such as Random Forest and SVM and compare it with CNN for the detection of cucumber diseases. These studies showed that the deep learning approach, particularly the deep CNN model, is a powerful tool for identifying symptoms of plant diseases. From the literature, it is found that transfer learning can be considered as one of the most popular techniques used for plant diseases identification in many of the works [13–20]. The transfer learning method lets developers to change goals and adjust CNN models that have been developed to adapt it to their tasks. From the literature, it can be concluded that in-depth studies have been carried out on the detection and identification of plant diseases using deep learning approaches. However, limited attention has been given to applying robust deep learning models of plant disease detection to the edge. The latest article presents an example of a real-life plant disease detection application, where an in-depth learning model of coconut disease detection is implemented in a web application created with the Flask micro web framework [21]. A similar approach is shown in another article [22]. They developed a tomato disease detection model on the server and built an Android mobile application which can work collaborate with the server. Different than executing a deep learning model on a centralized server (which run multiple GPUs), applying the model on an integrated system operating at the edge will require higher computing power. Edge solution however will have lower latency because the inference procedure can be executed from the device itself [23]. It is especially significant when we consider the practicality of using deep learning model to detect plant diseases in real life. Users of such devices might be in rural areas where there is no internet or with limited internet connectivity.

With the emerging of microprocessor class such as the Tensor Processing Unit (TPU) and the Vision Processing Unit (VPU), we believe that the idea of edge computing can be facilitated, in which the microprocessor acted as an AI accelerator to run CNN for image classification tasks on the edge. The VPUs/TPUs units usually come in a form of USB stick/drive which is compatible to a lot of systems/devices. VPUs can easily be used to integrate and implement deep learning models to the edge using specialized development kits or programming libraries (i.ei. e. Edge TPU and PyCoral runtime for Coral accelerator, and OpenVINO for Intel NCS2) [24].

For the project, we used OpenVino which allows us to implement the best performing models on peripheral devices using the model optimizer function. The model optimizer function can adapt the developed model to be optimally executed on Intel based hardware such as CPU, GPU, FPGA and VPU. In the project, we explored four (4) pre-trained models (AlexNet, VGG16, ResNet50 and DenseNet121) which are supported by the OpenVINO toolkit. The model was then evaluated in the PyTorch environment. These four models were prepared and evaluated using the well-known PlantVillage dataset, which is commonly utilized to evaluate the performance of multi-plant diseases detection [25–27]. The adoption of the OpenVINO toolkit. Here, we evaluate the best performing model by running inference on different hardware configuration in different environments. The Intel NCS2 and OpenVINO Toolkit configurations will represent implementation on the edge. By evaluating the model running on different hardware and software configurations, we can assess the practicality of implementing the model on the edge.

The main contribution of the current study is the development of a deep learning model in the PyTorch environment which can identify plant diseases of various plant species based on plant leaf images on the edge. A comparative analysis of the best performing model's performance on different configurations is reported.

2. Materials and methods

2.1. Dataset

For the project, we use dataset from the PlantVilllage for training, validation, and testing. The dataset has a total of 55,446 images, with standardized images [28], and includes healthy and diseased plant leaves images which were obtained from 14 different types of plants with an additional class of 1143 background images. An example image from the used dataset is shown in Fig. 1.

The dataset is imbalance as images of each class is not the same. If the deep learning model is developed from this imbalanced dataset, it is expected that the probability of the predicted results will move towards the class with more images. To address this imbalance issue, we employed the under-sampling method. The method would cut the dominant class which would level the dataset. Furthermore, the plant types which has healthy images only, were removed to reduce the training cost and time. To reduce overfitting problems, image augmentation techniques were applied after we separated the dataset for training, validation, and testing in a ratio of 75:15:10 [29]. The augmentation techniques, which includes 45^0 rotation, vertical and horizontal flip, and addition of random noise were conducted on classes with have less images to increase the overall dataset [30]. In Fig. 2, sample images of each transformation isare presented. The distribution of the dataset is shown in Table 1.

2.2. Preliminary experiment - Model selection and system requirement

The four (4) pre-trained models, namely AlexNet, VGG16, ResNet50, and DenseNet12, were explored in this project, using the PyTorch library. These models were chosen because they were supported by the OpenVINO [31] and many works on plant disease detection are based on these models [32–37]. As the hardware used in the project was limited, only supported models were explored. Other image processing and plotting libraries such as the Python Imaging Library, NumPy, and Matplotlib were also used in the edge prototype development. The training of the models was conducted using a CUDA-enable graphics processing unit which then were converted into ONNX format. The OpenVINO toolkit was used to optimize the models for executing inference on of Intel hardware (i.e. CPUs, GPUs, and VPUs).

2.3. Selection and fine-tuning of the best performing Model

In this work, a transfer learning approach was adopted because of the limited computational power to develop CNN model which can run on the VPU/TPU devices as edge implementation. The trained model parameters from the selected model were adopted as initial values for the improvement of the models aimed at detecting plant diseases, without having to build a model architecture from scratch. All the four models were trained using the training data set shown in Table 1. Out of the four models, only the best performing model will be selected and implemented in the edge settings.

2.4. Edge implementation – Model conversion and OpenVINO settings

The optimized DenseNet121 was saved as.PTH files to analyze its performance in the stock PyTorch environment. To examine the inference results using Intel hardware (via the OpenVINO toolkit), the optimized DenseNet121 should be converted and exported into one of the deep learning frameworks which is supported by the hardware. For our work, the developed PyTorch model was converted to ONNX format, which is acceptable to the Intel Model Optimizer.

Models in the ONNX format were exported into two different Intermediate Representation (IR) files with the FP16 and FP32 floating point formats. The use of these two formats were because of the types of VPUs and CPU which we used [38]. In this project, the IR model files, that

Fig. 1. Sample images of plant leaf diseases (a) Apple – Apple Scab (b) Cherry – Powdery Mildew (c) Peach – Bacterial Spot (d) Potato – Early Blight.

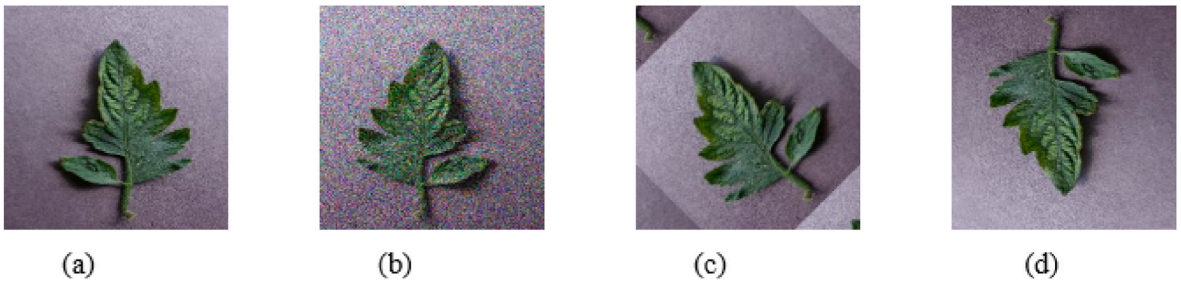

Fig. 2. Sample images of plant leaf diseases (a) Apple – Apple Scab (b) Cherry – Powdery Mildew (c) Peach – Bacterial Spot (d) Potato – Early Blight.

consists of.xml and.bin files, were configured as the inference engine which perform the inference based on optimized DenseNet121 on CPU and VPU.

Following Ming-Hsuan Tu work [39], all images were converted to input vectors in 224 x 224 size and normalized to DenseNet121 input requirements. This will guarantee that identical images are fed into the model when operating in the PyTorch and OpenVINO environments.

2.5. Performance evaluation

Accuracy of the classification result is one of the metrics used for evaluating the performance of a deep learning model. Nonetheless, it is not an optimal metric for evaluating the model's performance of this project, because the distribution of sample images in the training data set between classes is not the same.

To get a holistic view of how well the developed model performs on the modified data set and the impact of misclassification, evaluations using confusion matrices were employed to analytically determine the relationship between the predicted results and the expected results. Confusion matrices were computed by distinguishing the True Negative (TN), True Positive (TP), False Negative (FN) and False Positive (FP) cases. Next, values obtained from the confusion matrices were utilized to compute other metrics, such as Specificity, Recall, Precision and F1 Score [40]. The definitions and formulae for these metrics are shown in Table 2. For multiple-classes classification tasks, it is preferable to calculate the macro mean/macro averaged of the metrics mentioned above, since each class is given equal importance. The metric provides a better picture of how well the developed model is performing. Calculations of the macro mean/macro averaged are made by adding the sum of each metric, then split the result by the number of classes.

The experiments are carried in the CPU and GPU environments with processor Intel i7 7700HQ 2.8 GHz, and GPU card, Nvidia GTX 1050 4 GB. The memory is 16.0 GB.

3. Results and discussion

3.1. Preliminary experiments – Model selection

The classification layer of the pre-trained model was revised and transformed. The layer included and was connected related to ReLU activation, Dropout layers, and a fully connected layer with SoftMax. By adopting Dropout layers, the problem of overfitting would be overcome. And by neglecting a particular neuron in the network, it would hinder the co-dependency between neurons.

We freeze the feature extraction layer and in training, only the weights of the classification layer were updated. The batch size was tuned to 65 and the probability of dropping out was adjusted to 0.4. The learning rate was set to 0.001. And the training stages were finalized and set to 50 epochs [41]. As illustrated in Table 3, the uppermost accuracy was recorded by DenseNet121-based model with 96.4% followed by ResNet50 with 94.5%. DenseNet is an architecture which is developed to tackle the vanishing gradient and accuracy degradation issues. While ResNet initiates the idea of skip connection, DenseNet explores it further by linking each layer to all other layers using a feed-forward framework. This, so called Dense block, provides a powerful gradient flow to the network, and improves the accuracy of the model. It is because the residual error from the block can be propagated to the previous layer without difficulty via additional paths, and the magnitude of the error can be more correctly represented in the weight and bias correction. In addition, DenseNet also offers better efficiency in terms of its computing process as the amount of its channels can be reduced [42]. Based on Table 3, we can conclude that DenseNet121 was the best model in the experiments, as it achieves high accuracy. Therefore, DenseNet 121 was chosen as the best performing model. To improve the best performing model, the hyperparameters were then adjusted during the training stage. The training was conducted according to OpenVino documentation for End-to-End application [43]. Table 4 presents the optimized hyperparameters after the fine-tuning.

3.2. Edge implementation

The performance of the best performing model was assessed and appraised on several configuration (i.e., different hardware settings which run in different environments). The optimized DenseNet121 was evaluated with testing set. The evaluation of the model was conducted in four experiment settings.

For the first and second experiment settings, the models were tested in a standard PyTorch environment which run in CPU mode (using CPU-

Table 1
Number of images for training, validation, and testing.

Class Name	Total number of images	Training Data (75%)	Validation Data (15%)	Test Data (10%)
Apple – Apple Scab	1260	945	189	126
Apple – Black Rot	1242	932	186	124
Apple – Cedar Apple Rust	1375	1031	206	138
Apple – Healthy	1645	1234	247	164
Background	1143	857	171	115
Cherry – Healthy	1708	1281	256	171
Cherry – Powdery Mildew	1052	789	158	105
Corn – Cercospora Leaf Spot	1026	770	154	102
Corn – Common Rust	1192	894	179	119
Corn – Healthy	1162	872	174	116
Corn – Northern Leaf Blight	1970	1478	296	196
Grape – Black Rot	1180	885	177	118
Grape – Esca	1383	1037	207	139
Grape – Healthy	2115	1586	317	212
Grape – Leaf Blight	1076	807	161	108
Peach – Bacterial Spot	2297	1723	345	229
Peach – Healthy	1800	1350	270	180
Pepper bell – Bacterial Spot	1994	1496	299	199
Pepper bell – Healthy	1477	1108	222	147
Potato – Early Blight	1000	750	150	100
Potato – Healthy	760	570	114	76
Potato – Late Blight	1000	750	150	100
Squash – Powdery Mildew	1835	1376	275	184
Strawberry – Healthy	2280	1710	342	228
Strawberry – Leaf Scorch	1109	832	166	111
Tomato – Bacterial Spot	2127	1595	319	213
Tomato – Early Blight	1000	750	150	100
Tomato – Healthy	1591	1193	239	159
Tomato – Late Blight	1908	1431	286	191
Tomato – Leaf Mold	1904	1428	286	190
Tomato – Mosaic Virus	1865	1399	280	186
Tomato – Septoria Leaf Spot	1771	1328	266	177
Tomato – Spider Mites	1676	1257	251	168
Tomato – Target Spot	1404	1053	211	140
Tomato – Yellow Leaf Curl	1510	1133	227	150
TOTAL	**52837**	**39360**	**7926**	**5281**

Table 2
Description and equation of the performance metrics.

Performance metrics	Description	Equation
Precision	Focus on the positive classes. Defined as the correctness of a positive prediction.	$\frac{TP}{TP+FP}$
Recall	Also named as the sensitivity or true positive rate of the model. Recall is the ratio of predicted true positives over all the positives.	$\frac{TP}{TP+FN}$
Specificity	Known as the true negative rate of the model. The proportion of true negative out of all the predicted negatives.	$\frac{TN}{TN+FP}$
F1-Score	Harmonic mean of Precision and Recall	$\frac{2*TP}{2*TP+FN+FP}$

Table 3
Validation accuracies of the models after 50 epochs.

Pre-Trained Models	Validation Accuracy (%)
AlexNet	90.6
VGG16	89.7
ResNet50	94.5
DenseNet121	96.4

Table 4
Optimized hyperparameters.

Items	Values
Number of epochs	50
Learning rate	0.001
Batch size	65
Dropout Rate	0.4
Optimizer	Adam
Loss function	Negative Log-Likelihood
Validation data size	7926
Training data size	39,360

i7-7700HQ) and in GPU model (using GPU-GTX1050) to observe its accuracies and inference time difference. It is hypothesized that the accuracies of model run on this different hardware (CPU and GPU) will be identical because the model format does not change when operating in a standard PyTorch environment. Next for the third and fourth experiment settings, the model IR file was created following the model conversion method mentioned above, and model inference were performed in the OpenVINO environment which run on two different settings. The inference engine run on CPU (with the CPU-i7-7700HQ) and on VPU (with Intel NCS2).

3.3. Accuracies on different platforms

As shown in Table 5, the developed model achieves accuracies ranges from 75% (i.e Corn – Cercospora Leaf Spot) to 100% (i.e Potato - Early Blight) for all 35 classes. Results are slightly better than reported method by Barbedo [44]. The average test accuracies of the model in both environments on the CPU is similar with average accuracies of 95.6857%. A trivial difference, however, could be perceived when the inference was carried out on the Intel NCS2 with average accuracy of 95.742%. The differences can be explained because of quantization of model parameter weights in different floating-point formats, and the fact that the model has less layer supported by VPU plugins when using the OpenVINO toolkit. It is interesting to note that the accuracy is slightly better with the Intel NCS2 indicating a good potential for the model to be used in edge setting/embedded application.

3.4. Evaluation on different platforms

Getting a high recall value is important in this task because recall

Table 5
Testing accuracies of the proposed model in PyTorch and OpenVINO which run on CPU, GPU and VPU.

Class	Prediction Accuracies on Test Data		
	CPU-i7-7700HQ/ GPU - 1050 (Stock PyTorch)	CPU-i7-7700HQ (OpenVINO – FP32)	Intel NCS2 (OpenVINO – FP16)
Apple – Apple Scab	0.91	0.91	0.9
Apple – Black Rot	0.93	0.93	0.92
Apple – Cedar Apple Rust	0.97	0.97	0.96
Apple – Healthy	1	1	1
Background	1	1	0.99
Cherry – Healthy	0.98	0.98	0.99
Cherry – Powdery Mildew	0.95	0.95	0.95
Corn – Cercospora Leaf Spot	0.75	0.75	0.75
Corn – Common Rust	1	1	1
Corn – Healthy	1	1	1
Corn – Northern Leaf Blight	0.93	0.93	0.93
Grape – Black Rot	0.96	0.96	0.96
Grape – Esca	0.98	0.98	0.98
Grape – Healthy	1	1	1
Grape – Leaf Blight	0.97	0.97	0.97
Peach – Bacterial Spot	1	1	1
Peach – Healthy	1	1	1
Pepper bell – Bacterial Spot	0.92	0.92	0.94
Pepper bell – Healthy	0.99	0.99	0.99
Potato – Early Blight	1	1	1
Potato – Healthy	0.93	0.93	0.93
Potato – Late Blight	0.93	0.93	0.94
Squash – Powdery Mildew	1	1	1
Strawberry – Healthy	1	1	1
Strawberry – Leaf Scorch	1	1	1
Tomato – Bacterial Spot	0.94	0.94	0.94
Tomato – Early Blight	0.82	0.82	0.8
Tomato – Healthy	0.99	0.99	0.99
Tomato – Late Blight	0.94	0.94	0.96
Tomato – Leaf Mold	0.90	0.90	0.91
Tomato – Mosaic Virus	0.99	0.99	0.99
Tomato – Septoria Leaf Spot	0.97	0.97	0.97
Tomato – Spider Mites	0.95	0.95	0.95
Tomato – Target Spot	0.92	0.92	0.93
Tomato – Yellow Leaf Curl	0.97	0.97	0.97
AVERAGE	**0.956857**	**0.956857**	**0.95742**

values are sensitive to the False Negative (FN) count. A high FN count denotes that the model will detect positive cases of plant disease as negative, which can instigate the spread of the diseases and cause severe cost effects on crop yields. Table 5 shows the recall, specificity, precision, and F1 scores of the models operating in different configurations. For multiple-classes classification tasks, it is preferable to calculate the macro mean/macro averaged of the metrics mentioned above, since each class is given equal importance. The model's macro mean/macro averaged is shown in Fig. 3. As shown in Fig. 3, the performance of the VPU (Intel NCS2) can keep up with others. The model run on NCS2 has a high recall value with a comparable inference time count. This indicates that the developed model can be implemented as edge computing application to detect plant diseases. This setting will be enabled application of deep plant disease detection on embedded devices for plant monitoring or adopt it in robot as part of plant detection and monitoring.

3.5. *Inference time on different platforms*

Fig. 4 shows the inference time of the four experiment settings. The time was measured by calculating the average time that it took for the model to execute all test images of one of the classes. Executing the model in the stock PyTorch environment and process it on the GPU produced the shortest time to get the inference result (with the average time of 34 ms). In the experiments, the model was run on the CPU in two different environments (PyTorch and OpenVINO). The inference time was considerably less when the model was operating in OpenVINO. The Inference Engine of the OpenVINO used the CPU plugins as the model optimizer and re-constructed the model to optimally run-on Intel hardware. As for the VPU (Intel NCS2), it had an average of 86 ms to get the inference results in FP16 format. From Table 5, it can also be seen 6 that the VPU performance was slightly better that the CPU in both environment (with average accuracy of 0.95742) with comparable inference time.

4. Conclusion

In this work, a transfer learning approach was explored to develop an edge computing solution for plant leaf disease identification and detection. Four pre-trained models supported by OpenVINO toolkit were studied as the feature extraction tools. Image augmentation and down sampling techniques were applied to address the imbalance data problem. These four models were experimented with a dataset having a total of 52,837 images with 35 classes of healthy and diseases plant leaves. From the experiment, it was shown that the DenseNet121 was the best performing model out of the four models with the highest validation accuracy of 96.4% with is comparable with other reported methods. The model was then converted to different model formats using the Intel Model Optimizer. The new formats were required to run the inference on different Intel hardware (CPU, GPU and VPU). Results shows that there were no significant differences on the macro averaged metrics, even though the model is operated different configurations. The developed model was able to maintain relatively high recall values, precision and F1 scores. Conversely, we can observe a significant difference in inference time when time when the model on CPU but under two different environments. The shortest inference time was observed when the model operates on GPU mode. Lastly, result of operating the model on Intel NCS2 (VPU) with OpenVINO go beyond the result of operating the model on CPU mode in the PyTorch environment in terms of inference time and accuracies. This indicates the compact capabilities of the model and the possibility to implement it at the edge.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

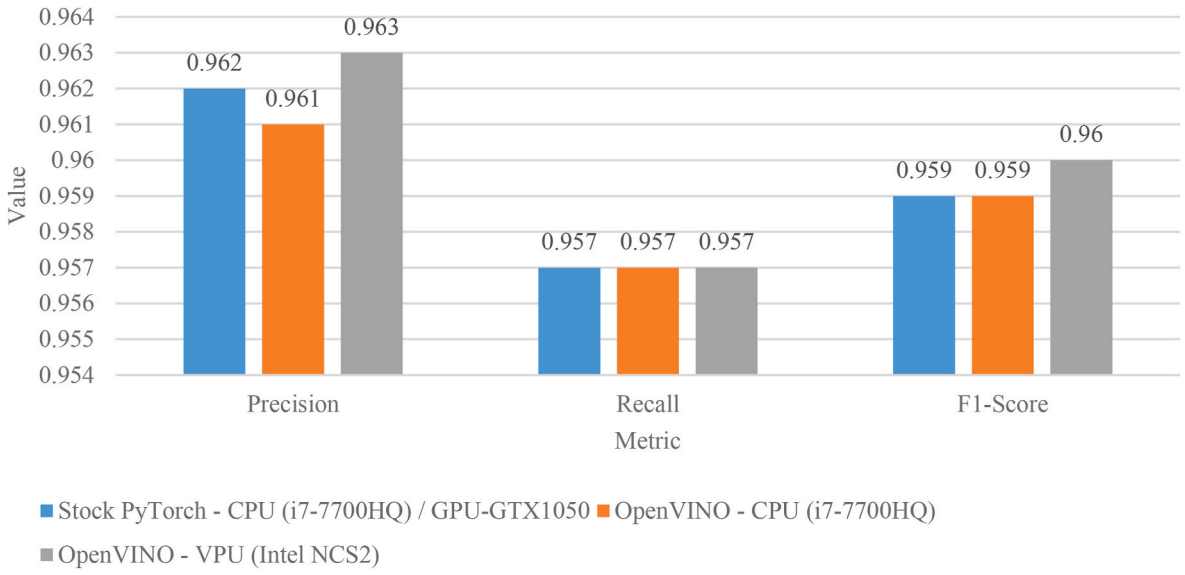

Fig. 3. Macro-averaged of the model.

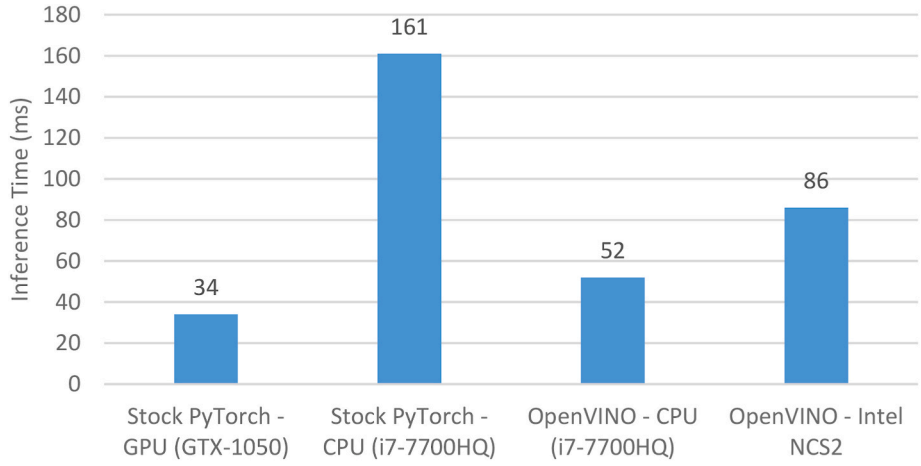

Fig. 4. Inference time of the model run on four experiment settings.

Data availability

The authors are unable or have chosen not to specify which data has been used.

Acknowledgments

Authors would like to express our gratitude to Mr. Ewe Kok Howg and Mr. Ho Nee Shen from Intel Malaysia for supporting this project.

References

[1] J.B. Ristaino, et al., The persistent threat of emerging plant disease pandemics to global food security, Proc. Natl. Acad. Sci. U. S. A 118 (2021) 23, https://doi.org/10.1073/pnas.2022239118.
[2] M. Riley, M. Williamson, O. Maloy, Plant disease diagnosis, Plant Health Instructor (2002), https://doi.org/10.1094/phi-i-2002-1021-01.
[3] K. Dokic, L. Blaskovic, D. Mandusic, From machine learning to deep learning in agriculture-the quantitative review of trends, IOP Conf. Ser. Earth Environ. Sci. 614 (1) (2020), https://doi.org/10.1088/1755-1315/614/1/012138.
[4] L. Ruetten, P.A. Regis, D. Feil-Seifer, S. Sengupta, Area-optimized UAV swarm network for search and rescue operations, 10th Ann. Comput. Commun. Workshop Conf. (2020) 613–618, https://doi.org/10.1109/CCWC47524.2020.9031197, 2020.
[5] R. Damayanti, N. Rachma, D.F. Al Riza, Y. Hendrawan, The prediction of chlorophyll content in african leaves (Vernonia amygdalina del.) using flatbed scanner and optimised artificial neural network, Pertanika J Sci Technol 29 (4) (2021) 2509–2530, https://doi.org/10.47836/PJST.29.4.15.
[6] A. Yuhao, N.N. Che'ya, N.A. Roslin, M.R. Ismail, Rice chlorophyll content monitoring using vegetation indices from multispectral aerial imagery, Pertanika J Sci Technol 28 (3) (2020) 779–795.
[7] N.A. Roslin, N.N. Che'ya, N. Sulaiman, L.A.N. Alahyadi, M.R. Ismail, Mobile application development for spectral signature of weed species in rice farming, Pertanika J Sci Technol 29 (4) (2021) 2241–2259, https://doi.org/10.47836/PJST.29.4.01.
[8] L.C. Ngugi, M. Abelwahab, M. Abo-Zahhad, Recent advances in image processing techniques for automated leaf pest and disease recognition – a review, Inform. Process. Agric. 8 (1) (2021) 27–51, https://doi.org/10.1016/j.inpa.2020.04.004.
[9] T.F. Gonzalez, Handbook of approximation algorithms and metaheuristics, Handb. Approximat. Algorith. Metaheuristics (2007) 1–1432, https://doi.org/10.1201/9781420010749.
[10] K.P. Ferentinos, Deep learning models for plant disease detection and diagnosis, Comput. Electron. Agric. 145 (2018) 311–318, https://doi.org/10.1016/j.compag.2018.01.009. February.
[11] X. Yang, T. Guo, Machine learning in plant disease research, Eur. J. Biomed. Res. 3 (1) (2017) 6, https://doi.org/10.18088/ejbmr.3.1.2017.pp6-9.
[12] T. Daniya, S. Vigneshwari, A review on machine learning techniques for rice plant disease detection in agricultural research, Int. J. Adv. Sci. Technol. 28 (13) (2019) 49–62.
[13] Ming-Hsuan Tu, The Inference Result Is Totally Different after Converting Onnx to Openvino IR - Intel Community, Intel Community, Jan. 2019.
[14] G. Geetharamani and A. P. J., "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," Comput. Electr. Eng., vol. 76, pp. 323–338, 2019, doi: 10.1016/j.compeleceng.2019.04.011..

[15] J. Ma, K. Du, F. Zheng, L. Zhang, Z. Gong, Z. Sun, A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network, Comput. Electron. Agric. 154 (Nov. 2018) 18–24, https://doi.org/10.1016/j.compag.2018.08.048.
[16] M. Agarwal, S.K. Gupta, K.K. Biswas, Development of Efficient CNN model for Tomato crop disease identification, Sustain. Comput.: Inform. Syst. 28 (Dec. 2020), 100407, https://doi.org/10.1016/j.suscom.2020.100407.
[17] S.P. Mohanty, D.P. Hughes, M. Salathé, Using deep learning for image-based plant disease detection, Front. Plant Sci. 7 (September, 2016), https://doi.org/10.3389/fpls.2016.01419.
[18] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, D.P. Hughes, Deep learning for image-based cassava disease detection, Front. Plant Sci. 8 (Oct. 2017) 1852, https://doi.org/10.3389/fpls.2017.01852.
[19] J.G.A. Barbedo, Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification, Comput. Electron. Agric. 153 (Oct. 2018) 46–53, https://doi.org/10.1016/j.compag.2018.08.013.
[20] J.G. Arnal Barbedo, Plant disease identification from individual lesions and spots using deep learning, Biosyst. Eng. 180 (2016) 96–107, https://doi.org/10.1016/j.biosystemseng.2019.02.002, 2019.
[21] P. Singh, A. Verma, J.S.R. Alex, Disease and pest infection detection in coconut tree through deep learning techniques, Comput. Electron. Agric. 182 (2021), 105986, https://doi.org/10.1016/j.compag.2021.105986. January.
[22] S. Verma, A. Chug, A.P. Singh, S. Sharma, P. Rajvanshi, Deep Learning-Based Mobile Application for Plant Disease Diagnosis, 2019, pp. 242–271, https://doi.org/10.4018/978-1-5225-8027-0.ch010. January.
[23] R. Sanchez-Iborra, A.F. Skarmeta, TinyML-enabled frugal smart objects: challenges and opportunities, IEEE Circ. Syst. Mag. 20 (3) (2020), https://doi.org/10.1109/MCAS.2020.3005467.
[24] B.C. Lai, P.J. McKerrow, J. Abrantes, The abstract vector processor, Microprocess. Microsyst. 30 (2) (2006), https://doi.org/10.1016/j.micpro.2005.06.002.
[25] S.H. Lee, H. Goëau, P. Bonnet, A. Joly, New perspectives on plant disease characterization based on deep learning, Comput. Electron. Agric. 170 (December 2019), 105220, https://doi.org/10.1016/j.compag.2020.105220, 2020.
[26] G. Geetharamani and A. P. J., "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," Comput. Electr. Eng., vol. 76, pp. 323–338, 2019, doi: 10.1016/j.compeleceng.2019.04.011.
[27] S.P. Mohanty, D.P. Hughes, M. Salathé, Using deep learning for image-based plant disease detection, Front. Plant Sci. 7 (September, 2016), https://doi.org/10.3389/fpls.2016.01419.
[28] A. Pandian, G. Gopal, Data for Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network, 2019.
[29] P. K., S. S., and S. P., "Automated classification of pulmonary tuberculosis-associated radiograph in the US hospital-scale chest X-ray database by using deep convolutional neural network," Open Forum Infect. Dis., vol. 5, 2018.
[30] L. Ruetten, P.A. Regis, D. Feil-Seifer, S. Sengupta, Area-optimized UAV swarm network for search and rescue operations, 10th Ann. Comput. Commun. Workshop Conf. (2020) 613–618, https://doi.org/10.1109/CCWC47524.2020.9031197, 2020.
[31] OpenVINO Developer, Converting a ONNX* Model - OpenVINO™ Toolkit, OpenVINO Documentation, 2021.
[32] L. Li, S. Zhang, B. Wang, Plant disease detection and classification by deep learning - a review, IEEE Access 9 (2021) 56683–56698, https://doi.org/10.1109/ACCESS.2021.3069646.
[33] K. Deeba, B. Amutha, ResNet - Deep Neural Network Architecture for Leaf Disease Classification, Microprocess Microsyst, 2020, 103364, https://doi.org/10.1016/j.micpro.2020.103364. October.
[34] V. Kumar, H. Arora, Harsh, J. Sisodia, ResNet-based approach for detection and classification of plant leaf diseases, in: Proceedings of the International Conference on Electronics and Sustainable Communication Systems, ICESC 2020, 2020, pp. 495–502, https://doi.org/10.1109/ICESC48915.2020.9155585. Icesc.
[35] B. Chellapandi, M. Vijayalakshmi, S. Chopra, Comparison of pre-trained models using transfer learning for detecting plant disease, in: Proceedings - IEEE 2021 International Conference on Computing, Communication, and Intelligent Systems, ICCCIS 2021, 2021, pp. 383–387, https://doi.org/10.1109/ICCCIS51004.2021.9397098.
[36] S. Vallabhajosyula, V. Sistla, V.K.K. Kolli, Transfer learning-based deep ensemble neural network for plant leaf disease detection, J. Plant Dis. Prot. (2021), 0123456789, https://doi.org/10.1007/s41348-021-00465-8.
[37] V.K. Vishnoi, K. Kumar, B. Kumar, Plant disease detection using computational intelligence and image processing, Springer Berlin Heidelberg 128 (1) (2021), https://doi.org/10.1007/s41348-020-00368-0.
[38] OpenVINO Developer, Supported Devices - OpenVINO™ Toolkit, OpenVINO Documentation, 2021.
[39] Ming-Hsuan Tu, The Inference Result Is Totally Different after Converting Onnx to Openvino IR - Intel Community, Intel Community, Jan. 2019.
[40] G. Ciaburro, B. Venkateswaran, Neural Netw. World R 91 (2017).
[41] U. Michelucci, Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks, 2018, https://doi.org/10.1007/978-1-4842-3790-8.
[42] T. Sik-Ho, Review: DenseNet — Dense Convolutional Network (Image Classification), Sik-Ho Tsang, Nov. 2018. Towards Data Science," towardsdatascience.
[43] OpenVino, Tuning for Performance, Sep. 2022. *https://docs.openvino.ai/latest/openvino_docs_optimization_guide_dldt_optimization_guide.html#doxid-openvino-docs-optimization-guide-dldt-optimization-guide*.
[44] J.G. Arnal Barbedo, Plant disease identification from individual lesions and spots using deep learning, Biosyst. Eng. 180 (2016) (2019) 96–107, https://doi.org/10.1016/j.biosystemseng.2019.02.002.