Computing and Informatics, Vol. 42, 2023, 1378–1403, doi: 10.31577/cai_2023_6_1378

EMBEDDED PLANT DISEASE RECOGNITION USING DEEP PLANTNET ON FPGA-SOC

Taoufik Saidani, Refka Ghodhbani, Ahmed Alhomoud

Department of Computer Sciences
Faculty of Computing and Information Technology, Northern Border University
Rafha 91911, Saudi Arabia
e-mail: {Taoufik.Saidan, Refka.Ghodhbani, aalhomoud}@nbu.edu.sa

Mohamed Ben Ammar

Department of Information Systems
Faculty of Computing and Information Technology, Northern Border University
Rafha 91911, Saudi Arabia
e-mail: mohammed.ammar@nbu.edu.sa

Abstract. Technological breakthroughs have ushered in a revolution in a variety of industries, including agriculture, during the last several decades. This has given rise to what is now known as Agriculture 4.0, which emphasizes strategy and systems rather than the traditional obligations of the past. As a result, many human procedures have been replaced by a new generation of intelligent devices. Crop production management in Agriculture 4.0, on the other hand, poses a considerable challenge, particularly when it comes to prompt and accurate crop disease identification. Plant diseases are of special significance since they significantly reduce agricultural yield in terms of both quality and quantity. Deep learning neural network models are being utilized for early diagnosis of plant diseases in order to overcome this difficulty. These models can automatically extract features, generate high-dimensional features from low-dimensional ones, and achieve better learning results. In this research, we offer a joint solution involving image processing, phytopathology, and embedded platforms that intends to minimize the time necessary for human labor by leveraging AI to facilitate plant disease detection. We propose a learning-based PlantNet architecture for detecting plant diseases from leaf images, in which achieved about 97 % accuracy and about 0.27 loss on the PlantVillage dataset. However, because putting AI techniques on embedded systems can substantially cut energy consumption and processing times while also minimizing the costs and dangers involved with data transfer, it is worth considering. The second goal of this paper is to use high-level synthesis to accelerate the proposed PlantNet architecture. Moreover, we propose a hardware-software (HW/SW) design for integrating the suggested vision system on an embedded FPGA-SoC platform. Finally, we present a comparative study with state-of-the-art works, which demonstrates that the proposed design outperforms the others in terms of normalized GFLOPS (1.93), reduced power consumption (2.48 W), and minimum required processing time (0.04 seconds).

Keywords: FPGA, deep CNN, co-design, hardware acceleration, PYNQ-Z1

1 INTRODUCTION

The technologies of agricultural domain are quickly evolving towards a new paradigm, Agriculture 4.0. In this paradigm, digitization, artificial intelligence, and automation play a major role in plant production, including pest control and weed control [1].

This development presents many opportunities full of challenges, as well as the shift from animal and manual technologies to mechanized and automated equipment in developing countries and bridging the digital divide. Traditional mechanization of early agriculture, characterized by using engine power and tractors, will be matched and even surpassed by the automated mechanism and robotics and the precision they can provide in agricultural operations [2]. Agriculture 4.0 is considered to be one of the most important parts of the industry, which, according to the Food and Agriculture Organization of the United Nations (FAO), faces the challenge of increasing its productivity by up to 60 percent during the 21st century in order to provide an adequate food supply for the expanding population around the world [3]. Agriculture 4.0 is considered one of the most important parts of the industry. This target should be accomplished while taking into consideration the demand for computer resources in accordance with sustainable methods and strategies owing to the rising pressures on ecosystems, biodiversity, land, and water. This objective should be accomplished while taking into account the necessity of computing resources in accordance with sustainable techniques and strategies. The widespread use of precision agriculture (FP) methods, which provide pervasive conceptual advances and technological improvements from "smart" agricultural production in the direction of Agriculture 4.0, is a significant contributor to the advantages cited in [4]. Keeping in mind that in recent years, the expansion of global commerce and globalization, in addition to changes in the environment, have led to a rise in the number of plant diseases. This has reached the level of an epidemic in a number of nations, which has resulted in the loss of a significant number of crops and therefore threatens the food and nutritional security of people.

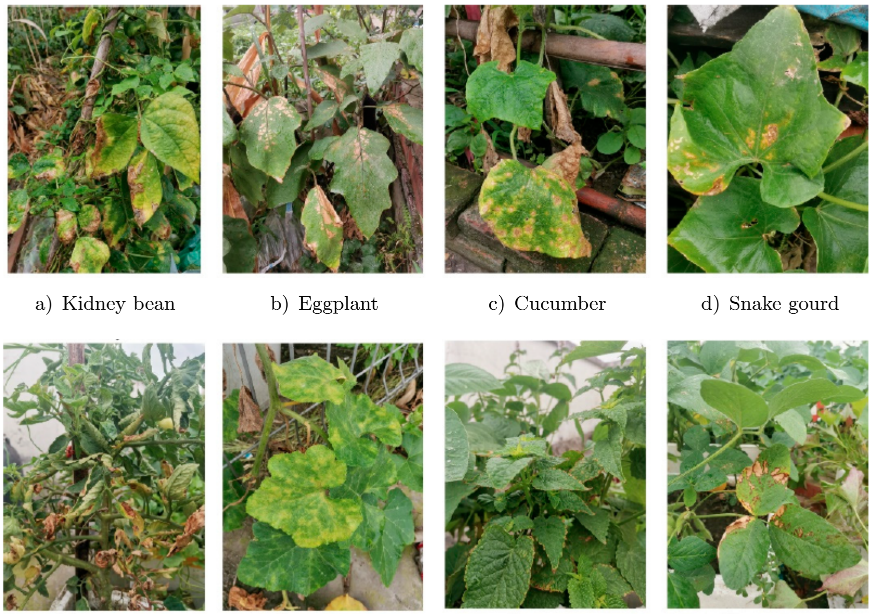
Plants are also vulnerable to several diseases types in their different phenophases, like humans [5]. Therefore, the total crop yield and therefore the farmer's net profit is negatively affected. Various diseases of plant have a huge effect on the growth of the crop's food. The Irish potato famine of 1845–1849, which claimed the lives of 1.2 million people in Ireland [6], is an example of iconic. The following Table 1 provides a summary of a number of common plant diseases. Diseases that affect plants may be organized hierarchically into the categories of hyphomycetes, fungi, bacteria, and viruses. Several examples of prevalent plant diseases may be found in Figure 1. Farmers and academics are continuously looking at ways to improve upon an existing system that is clever and effective for classifying plant diseases. Enzyme immunoassay, polymerase chain reaction, and loop-mediated isothermal amplification are some of the methods that have been tried on plant samples by a large number of researchers. These methods are quite specific and sensitive when it comes to identifying the disease mentioned in the previous works [7].

Plant	**Major Types of Disease**		
	Fungal	**Bacterial**	**Viral**
Cucumber	Downy mildew mildew, gray mold spot, anthracnose	Angular spot, brown spot, target spot	Mosaic virus, yellow spot virus
Rice	Rice stripe blight, false smut, rice blast	Bacterial leaf blight, bacterial leaf streak	Rice leaf smut, rice black-streaked dwarf virus
Maize	Leaf spot disease, rust disease, gray leaf spot	Bacterial stalk rot, bacterial leaf streak	Rough dwarf disease, crimson leaf disease
Tomato	Early blight, late blight, leaf mold	Bacterial wilt, soft rot, canker	Tomato yellow leaf curl virus

Table 1. Common plants diseases

Currently, AI technology, which crowns deep learning, is already seen as a reality in the perimeter of precision agriculture. The agricultural AI market was estimated at nearly 518.7 million dollars in 2017 and is expected to grow by more than 22.5 % per year to reach 2.6 billion dollars by 2025 [8]. A system of object recognition finds objects in an image, using several models. However, the description-based algorithmic of this scheme, with the aim is to perform the on board-implementation, has been very difficult that is why machine learning techniques have been proposed to facilitate the recognition-based tasks. In order to avoid damage to agricultural yield, protection of plants against disease is essential to ensure the crops quantity and quality. An effective and powerful protection method must provide an early detection of the disease in order to select the right treatment, to prevent the spread, at the right time [9].

In order to solve plants and crops, in the new generation of agriculture, early diseases detection of plant is necessary. Manual diseases-based detection in plants


a) Kidney bean b) Eggplant c) Cucumber d) Snake gourd

e) Tomato f) Pumpkin g) Ginger h) Soybean

Figure 1. Leaf spot in eight common plants

is carried out either by farmers or agronomists [10]. However, this is a very difficult task and it needs more time. To solve this issue, researchers around the world have presented various advanced systems for automatic detection of plant diseases using various machine and deep learning techniques [11]. These schemes are based on artificial neural networks (ANN) and their other variants, such as convolutional neural networks (CNN) and recurrent neural networks (RNN) to make an identification about the internal data structures. Deep learning techniques have two important advantages over machine learning methods [12]. Firstly, they extract automatically various characteristics from the raw data, and therefore, there is no requirement for an additional extraction module. Secondly, these intelligent methods lead to reducing the time required to the processing task with a high-dimensional data sets. Therefore, deep learning techniques can be exploited to create hybrid models. CNN are ones of the most powerful and effective schemes for modeling complex processes and performing pattern recognition applications with large amounts of data, such as recognizing patterns in images [13]. Authors in [14] presented a CNN system for automated plant recognition based on leaf images. Authors in [15] have developed a powerful neural network to identify three different legume species based on morphological models of leaf veins. In [16], authors have compared two well-known and established CNN models to identify 26 plant diseases, on the basis of an open database of leaf images from 14 different type of plants. Authors in [17] analyze several deep learning architectures and their merits and shortcomings in the context of plant disease detection. Their paper also reviews recent research in the subject and explores the problems and limitations of deep learning for plant disease detection. Overall, they offer an excellent summary of current research in this topic and show the promise of deep learning approaches to enhance plant disease detection and treatment. Further work related to this topic is presented in Section 2.

Beyond the real involvement of advanced technologies, Agriculture 4.0's most important challenge towards sustainable growth lies on the ability to provide integrated systems dynamically which implemented more efficient and sophisticated farming operations (such as irrigation, cultivation, crop disease detection, etc.) at a lower cost [18]. This is in order to provide more efficient and safer operating conditions for both parties and the environment (involving farmers, agricultural engineers, policymakers, professionals cooperation development, etc.), and synergies increasing between all stakeholders by giving them the capacity of making decisions even on issues that did not generally belong to their expertise [19]. Because of this, it is almost essential to include AI in vision applications in order to provide these systems the ability to think for themselves and come to conclusions that are comparable to or the same as ours [20]. In this setting, the integration of AI offers a number of challenges, notably with regard to the optimization of algorithms. Nevertheless, the effectiveness of AI systems is highly reliant on the quantity and quality of the data that they utilize to learn and advance [21]. Embedded systems, notably the FPGA-SoC which is the most often used, have limited processing, memory, and communication capacity. This is in addition to the constraints placed on them by energy consumption and cost [22]. Despite this, accelerating the construction of CNNs utilizing FPGA SoC has emerged as a novel alternative due to its ability to maximize parallelism data processing and power efficiency while lowering the costs and dangers connected with data transmission. Additionally, by taking reconfiguration benefit, different CNN models and architectures can be easily reconfigured in the FPGA for many application types [23]. In this context many research work has been proposed to implement vision systems on FPGA-SoC. Authors in [24] proposed an FPGA-based 20 kfp streaming camera system called BinaryEye that recognizes interest region within real-time streaming mode. Authors in [25] designed an FPGA-based hardware architecture for real-time object detection based on CNN. Authors in [21] proposed a hardware-software architecture to implement a deep traffic sign recognition application on FPGA SoC. Additional research related to this topic is presented in Section 2.

In this work we propose an intelligent vision system for detecting leaf disease that is both low-cost and high-performance. To do so:

- We conduct a review of the five most commonly used deep learning models in this context and compare their computational complexity on the imageNet dataset;
- We select the SqueezeNet model with the lowest computational complexity for further improvement and rectification;

- We propose improvements to the SqueezeNet model by adjusting the hyper-parameters and number of layers to create an enhanced topology with lower computational cost than the original model which is the PlantNet architecture;
- We evaluate their proposed PlantNet architecture on the PlantVillage dataset and demonstrate its efficient performance;
- We propose to accelerate the PlantNet architecture using high-level synthesis on reconfigurable FPGA;
- We propose a hybrid hardware/software (HW/SW) design based on the accelerated PlantNet architecture and the ARM processor.

We organized this paper as follows. We reviewed the latest CNN networks to provide leaf disease classification in Section 2. Section 3 introduces the suggested deep learning model and the GPU software training performance. Section 4 presents the PlantNet model acceleration on FPGA SoC. Section 5 presents HW/SW PlantNet architecture for the plant disease detection. After that, we present the implementation results in Section 6. Finally, the paper is concluded in Section 7.

2 RELATED WORK

Plant diseases cause significant production and economic losses in agriculture. For example, soybean rust caused significant economic loss and just by eliminating 20 % of the infection, farmers can enjoy a profit of around 11 million dollars. It is estimated that crop losses due to plant pathogens in the United States amount to approximately 33 billion dollars annually. Of that amount, about 65 % (21 billion dollars) could be attributed to non-native plant pathogens [26]. Bacterial, fungal, and viral infections, as well as insect infestations, lead to plant disease and damage. In the United States, over 50 000 parasitic and nonparasitic plant diseases affect plants. When a plant becomes infected, symptoms occur on many regions of the plant, resulting in a significant agricultural effect [27]. Many of these microbial illnesses have spread over time in orchards and farms due to the unintentional introduction of vectors or diseased plant debris [27]. Pathogens can also spread through ornamental plants that act as hosts. These plants are frequently sold in stores before the disease is discovered. An early disease detection system can assist decrease plant disease losses and limit disease spread.

In this context, authors in [28] proposed a deep CNN for an accurate detection and identification of apple leaf disease. This approach achieved an average accuracy of 97.62 %. Authors in [29] proposed a hybrid classification approach-based citrus diseases detection using feature selection and weighted segmentation techniques. Here, the Gaussian technique is used for efficient diseases spot segmentation. This approach achieved an average accuracy of 95.80 %. Another hybrid clustering-based plant leaves images segmentation was proposed in [30]. In this technique authors, applied the superpixel clustering-based method, which aims to divide the original leaf color disease into a few hundreds of small compact regions, where the EM algorithm is used for the segmentation of the images leaf color disease. This technique reaches a higher accuracy of 100 %. The authors of [31] have proposed five different CNN architecture for a disease-based detection tool for the banana plant. These models were ResNet-152, VGG-16, ResNet-50, InceptionV3, and ResNet-18. However, the ResNet-152 model outperformed the other architectures with about 99.2 % of accuracy. In the same context, a mobile application was developed also so that farmers can detect easily banana diseases by downloading images of their banana leaves taken with their smartphones. This application implemented the InceptionV3 model to predict the plant disease with 99 % confidence. Other similar work was proposed in [32], it uses the PlantVillage dataset and the CNN VGG-19 and InceptionV3 architectures for automated plant disease identification. They also employed data augmentation in their study to artificially increase the dataset. According to their article, the VGG-19 model beats the InceptionV3 model with 98 % driving accuracy and 95 % test accuracy. Deep Convolutional Encoder Networks were offered by the authors of paper [33] as a method for the detection of diseases affecting seasonal crops. They analyzed 900 leaf photographs of three crops: potato, tomato, and maize, arranged in six groups. They were successful in training with an accuracy of one hundred percent, however the testing accuracy of their model was only 86.78 %. The authors in [34] present a pattern recognition system for identification and classification of three cotton leaf diseases. A dependent natural image was used as a data set. Therefore, an active contour model has been used to handle images and the extracted features are used to train adaptive neuro-fuzzy inference system. The identification system has achieved an average accuracy of 85 %. By the way, an approach that integrated image processing and automatic learning to allow the diagnosis of leaves diseases was proposed in [35]. This automated method classifies diseases on potato plants of the PlantVillage dataset, which is a publicly available plant image database. The approach of segmentation and the use of an SVM has demonstrated that the classification of the disease, for more than 300 images, achieves an average accuracy of 95 %.

At the moment, a number of scientific researchers have rethought their approaches in order to develop deep learning frameworks that are scalable and parallel [36]. More recently, their concept has been altered even further to include moving the learning process to GPUs. GPUs are notorious for the leakage currents that they produce, which, in turn, prevent any realistic development of deep learning models on embedded devices [37]. The use of FPGAs is yet another potential answer. Accelerators for deep learning based on field-programmable gate arrays (FPGAs) have been utilized to boost outcomes dramatically by optimizing data access pipelines [38]. In the study, a scalable architecture known as a Deep Learning Accelerator Unit (DLAU) was employed [39]. There are three pipeline processing units used by the DLAU. By using locality and tiling approaches, they were able to attain a speed that was 36.1 times quicker than CPUs while only consuming 234 mW of electricity. Another method, which used an architecture that was built on low-end FPGAs and included leaks, arc losses, and other such phenomena, was able to reach a detection rate of 97 %. They were able to reach a processing speed that was 7.5 times quicker than that of the software implementation.

These studies show that convolutional neural networks have been widely applied to the field of recognition of crops and plants and have achieved good results. However, although several techniques have been presented in the literature for automatic plant disease detection, a hybrid embedded system that combines a CNN with hardware-software (HW/SW) architecture has not been proposed until now in any existing research work to the best of our knowledge. A new CNN-based intelligent HW/SW system detection has been developed in this paper and accelerated on an FPGA to provide an in-depth embedded learning approach for plant disease detection.

3 PROPOSED DEEP CNN-BASED PLANT DISEASE DETECTION

Following the requirement of autonomous embedded systems in Agriculture 4.0, this paper proposes an embedded system for plant disease detection based on a deep learning approach. To do so, the design methodology of this paper is subdivided into three phases, as depicted in Figure 2. In the first stage (Phase 1), the deep PlantNet model will be introduced and optimized, and its results will be analyzed accordingly. In the second stage (Phase 2), the proposed model will be accelerated using the HLS tool of Xilinx and then the hardware-software design will be introduced. In the latest stage (Phase 3), the proposed embedded plant disease detection will be implemented on an FPGA SoC to create an autonomous embedded system able to recognize crop disease.

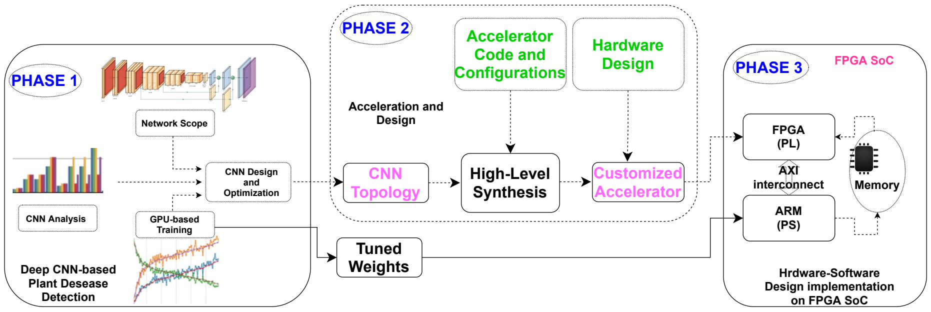


Figure 2. Design methodology

3.1 Deep PlantNet Model

Although research in the CNN field is very active and new architectures are emerging every day, much of the awareness currently appears to be focused on improving accuracy and efficiency. As we worked on how to improve CNN, we noticed that there is a huge lack of tools to visualize, analyze and compare CNN topologies which remains a critical issue. Therefore, we found the Netscope CNN analyzer [40] which represents a web-based tool written in CoffeScript, CSS, and HTML to analyze CNNs flow's data and memory requirements.

In the first step we starting by evaluating, using Netscope CNN analyzer, the most used deep pretrained CNN as well as InceptionV3, ResNet-50, VGG-16, SqueezeNet, and AlexNet. These CNN models compared are usually trained on ImageNet and expect input images of 227×227 or 224×224 pixels. The evaluated parameters are in term of Cnv_layers that indicates the number of convolutional layers. Then, the multiply operations number for one forward pass is considered and denoted as MACC. The Activation parameter is computed for each deep pretrained model to quantify the total number of pixels in all output feature maps. In addition, the ImageNet top-5 error rate is listed for each model to demonstrate the performance of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in terms of the percentage of times that the target label does not appear among the 5 highest-probability predictions [41]. The evaluation between these models is summarized in Table 2.

Models	Cnv_layers	MACC (M)	Parameters (M)	Activation (M)	Top-5 Error (%)
VGG-16	16	15 470	183.3	29	8.1
ResNet-50	50	3 870	25.5	46.7	7.0
InceptionV3	48	5 710	23.8	32.6	5.6
AlexNet	5	1 140	60.7	2.4	19.7
SqueezeNet	18	861	1.2	12.5	19.7

Table 2. CNN models evaluation for image classification on ImageNet

Table 1 shows that SqueezeNet has the lowest computational complexities in terms of MACCs, followed by AlexNet. In addition, the same four CNNs require the least amount of enabled memory. Looking at the amount of parameters, the SqueezeNet model has the fewest with a 19.7 % inaccuracy in the top 5. SqueezeNet, on the other hand, was chosen as the foundation for our CNN model because of its appropriateness for a SoC-based FPGA implementation in terms of decreased computational complexity and limited activations and parameter sets that can fit into a SoC FPGA memory. To do so, three types of improvements were implemented during the transformation of the original SqueezeNet, as depicted in Figure 3, to PlantNet CNN, which are efficiency improvements, FPGA-related improvements, and accuracy improvements.

As denoted in Figure 3, an initial convolutional layer, eight stacked firing units, a final convolutional layer, three grouping layers, and a layered dropout layer comprised the SqueezeNet architecture. A nonlinear ReLU enables these convolutional levels, and the structure concludes with a global mean pool level. Each firing unit, however, includes a compression layer (1×1 convolution and ReLU) as well as two parallel layers (1×1 and 3×3 convolutions and ReLU). In this context, 1×1 and

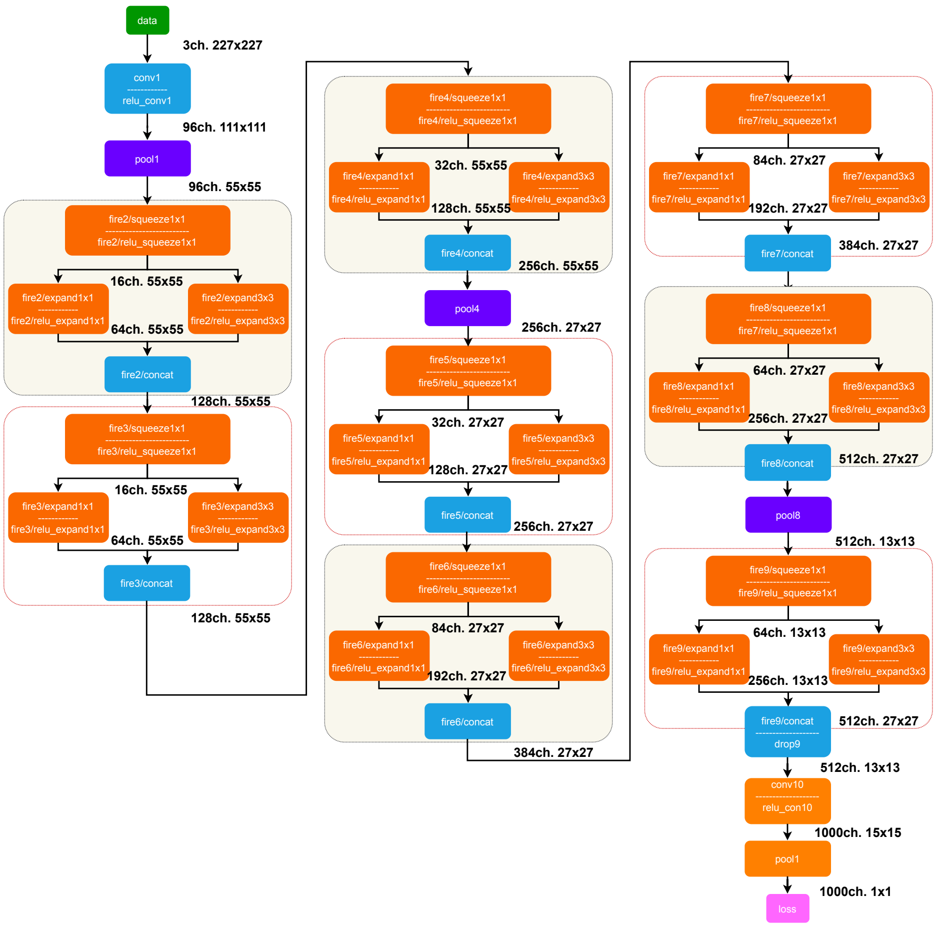


Figure 3. SqueezeNet topology

3×3 convolutions are essential operations in convolutional neural networks (CNN) for feature extraction. They apply a sliding window of size $1 \times 1/3 \times 3$ on the input data and perform element-by-element multiplication between the kernel values and the corresponding input values. 1×1 convolution provides dimensionality reduction, nonlinearity, and feature fusion by compressing channel information. On the other hand, 3×3 convolution captures local spatial dependencies, facilitates hierarchical feature extraction and enables translation invariance. Each compression layer has its own set of output channels that are in charge of compressing the internal representation. The 1×1 and 3×3 cores are assessed for the expansion level in this feature map, and their outputs are concatenated along the channel size.

As our goal is to rectify the SqueezeNet architecture to provide a PlantNet model that reaches a minimum computational complexity to fit the FPGA implementation, Figure 4 depicts the PlantNet model, the layer widths w_{out}, the layer capacities $w_{out} \times h_{out} \times ch_{out}$, and the number of output channels ch_{out} in each network stage. However, the proposed PlantNet model consists of six fire modules (fire1, fire2, fire3, fire4, fire5, and fire6) of SqueezeNet followed by and output convolution and pooling layers. Each convolution layer is complemented by ReLU function. In addition, we designed the spatial layer dimensions (width and height) for the proposed PlantNet to be powers of 2, such as 256 × 256 pixels. The intent behind this design choice is to allow repeated scaling of stride 2, resulting in integer dimensions.

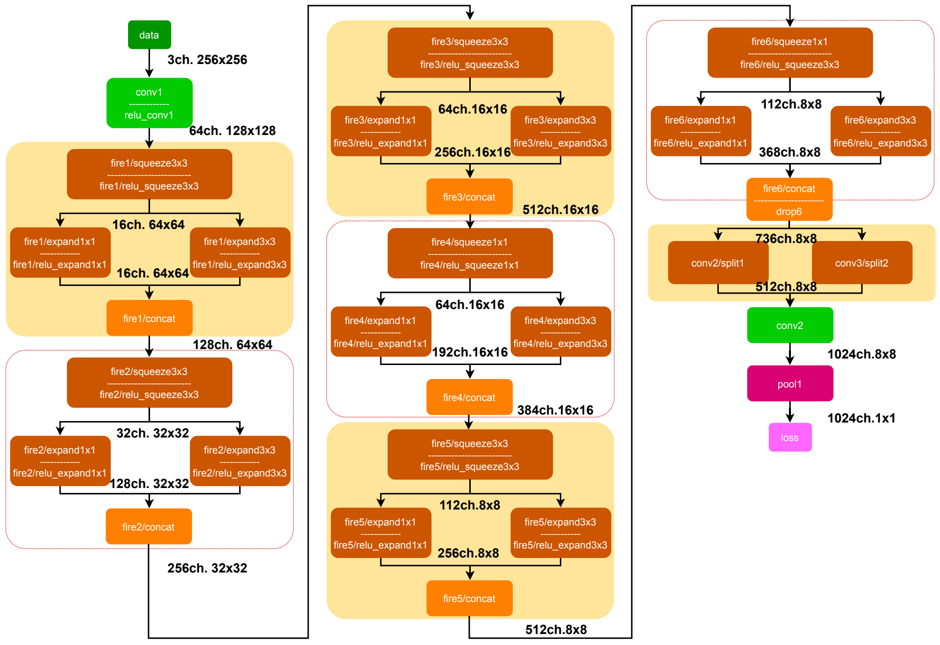


Figure 4. PlanNet topology

In each PlantNet individual layer the computational complexity has been analyzed with Netscope, and then the results are presented in Table 3. The most computational layer in SqueezeNet has been reduced in our proposed PlantNet model. However, the number of convolutional layers has been reduced from 18 (SqueezeNet) to 14 layers in PlantNet. In addition, the multiply and accumulate operations number for one forward pass (MACC) is also reduced from 861 M to 428.64 M which prove that our proposed PlantNet can reduce about 50 % of the computational complexity of the network. The number of activations also has been reduced from 12.5 M to 6.05 M in the proposed PlantNet model means about 50 % has been saved.

Layers	MACC	Parameters	Activation
conv1	28.31 Millions	1.79 k	1.05 Millions
fire1 (sub-module)	79.69 Millions	19.6 k	1.7 Millions
fire2 (sub-module)	79.69 Millions	78.11 k	851.97 k
fire3 (sub-module)	79.69 Millions	311.87 k	425.98 k
fire4 (sub-module)	39.85 Millions	156.1 k	327.68 k
fire5 (sub-module)	43.12 Millions	674.42 k	112.64 k
fire6 (sub-module)	30.05 Millions	470.35 k	155.65 k
drop6	–	–	47.1 k
conv2/split1, conv3/split2	48.23 Millions	754.69 k	65.54 k
conv4	–	–	65.54 k
pool1	–	–	1.02 k
Total	**428.64 Millions**	**2.47 Millions**	**6.05 Millions**

Table 3. Layer's PlantNet topology evaluation for image classification on ImageNet

So the proposed plantNet model was trained on the PlantVillages dataset consisting of about 20 000 images [42] of healthy and diseased plants. The used dataset contains 15 sub-directories in which each of them contains a number of crops images, as well as the pepper-bell-bacteria, the pepper-bell-healthy, the potato-early-blight, the potato-late-blight, the potato-healthy, the tomato-bacterial-spot, the tomato-early blight, the tomato-late-blight, the tomato-leaf-mold, the tomato-septoria-leaf, the tomato-spider-mites, the tomato-target-spot, the tomato-tomato-yello, the tomato-mosaic-virus, and the tomato-healthy. The Python programming language was used to train and evaluate the suggested deep learning model. Adam's optimization is a sophisticated optimization approach that changes the weights at each iteration and minimizes the gradient error between the ground truth labels and the prediction outputs. Furthermore, the studies were carried out using an Intel®core TM i7-3770@3.4 GHz CPU with 16 GB of RAM. We also employ an NVIDIA GeForce RTX 2070 GPU to speed up the deep trained model. The number of epochs, learning rate, and batch size were experimentally adjusted at 350, 10^{-4}, and 64, respectively, for deep learning parameters used in this study.

The datasets are gathered into two sub-sets, 75 % for training and 20 % for testing, and 5 % for validation including healthy and diseased plants. Figure 5 shows the accuracy and loss curves of the training and validation process in order to evaluate the performance of the proposed PlantNet model, after about 10 hours of training. The achieved training and validation accuracy is about 97 %. Similarly, the training and validation loss is about 0.27 for the proposed model. Consequently, the proposed deep CNN model reaches a good classification performances.

The suggested PlantNet architecture attained the best accuracy of 97 % when evaluated on the PlantVillage dataset, according to the performance statistics shown in Table 4. This outperforms the other models, including SqueezeNet, which is a common option in many similar publications. The results indicate that the suggested architecture is successful at detecting plant diseases from leaf images and has real-world applicability. As a result, the suggested PlantNet architecture may be regarded as a viable solution for plant disease detection and classification tasks.

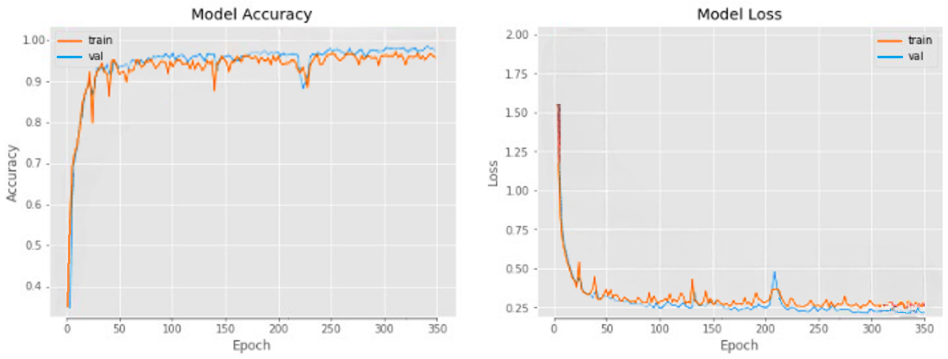


Figure 5. PlanNet training performance

Study	**Architecture**	**Dataset**	**Accuracy**	**Loss**
[43]	SqueezeNet	–	92.78 %	–
[44]	SqueezeNet	PlantVillage	95.05 %	–
[45]	SqueezeNet-MOD2	Strawberry Leaves	92.61 %	–
[46]	SqueezeNet	Strawberry Leaves	93 %	–
Our work	PlantNet	PlantVillage	97 %	0.27

Table 4. Performance comparison of CNN models for plant disease detection

4 PLANTNET MODEL ACCELERATION ON FPGA SOC

Within the context of the agricultural industry, the PlantNet model was developed specifically for high-performance disease detection in plants. The Xilinx PYNQ-Z1 System-on-a-Chip (SoC) is the foundation of the embedded platform. This device integrates a programmable FPGA fabric and a dual-core ARM Cortex-A9 CPU into a single package. The PYNQ-Z1 is equipped with a System-on-Chip (SoC) manufactured by Xilinx called the ZYNQ XC7Z020-1CLG400C, 512 MB of DDR3 memory dedicated to the ARM CPU, 4 GB of independent external SD memory, and a wide variety of connectivity options including USB, HDMI, and Gigabit Ethernet. The PYNQ-Z1 System-on-a-Chip (SoC) has a total of 133×10^3 logic slices, 53.2×10^3 look-up tables (LUTs), 630×10^3 (kB) of block RAM, 106.4×10^3 flip-flops, and 220 DSP slices [47].

Convolutional and average pooling layers make up the deep PlantNet model, and the ReLU nonlinearities are put to use as the activation function. The network is highly organized, which demonstrates the greatest layer-by-layer organization possible in the fire module architecture. Each firing module is made up of three convolutional layers, including one compression layer, two expansion layers, and one intermediate layer. After that, the output channels of the two expansion layers are concatenated to create a single feature map with twice as many channels as the original map had. While the dropout layer, also known as drop6, is only useful during the training phase of the process, it is safe to disregard it entirely during the inference phase. In this context, the concatenation capability of two layers is re-utilized in the convolution layer, which is computed into two separate divisions conv2 \ split1 and conv3 \ split2 to reduce memory requirements followed by a convolutional layer. After that, the pooling layer is used in order to reduce the dimensions by averaging from 8×8 to 1×1 pixels, while leaving the intactness of the channel dimension. In the final stage, the softmax function is applied to predict the class probabilities.

The computational complexity of the Deep PlantNet results entirely from the 1×1 and 3×3 convolution operations, which accumulate approximately 428.64 million MACC operations. The ReLU nonlinearities function add about 1.048 million comparisons operations. While, the average pooling needs 65 536 additions, and the final softmax executes 1 024 additions, divisions, and exponentiation. However, the softmax layer will be implemented on the ARM processor. While the convolutional layers, the ReLU, the concatenation layers, and the global average pooling layer are left to the FPGA. These layers must be efficiently accelerated in order to successfully run the PlantNet on the PYNQ-Z1 SoC.

The two-dimensional convolution (2DC) of several input feature is denoted as the most important operation that needs acceleration on FPGA SoC. The 2DC for an input image is represented as the result from filter sliding over the image, and computes the dot product between the filter and the pixels at each filter position. Thus, the convolution formula, of 2DC for an input image I with a height H and width W and a filter F with a kernel of $k \times k$, is denoted for each pixel $(y,\ x)$ in Equation (1). Given an RGB image, we typically consider three channels with an input channel size of ch_{in}. Therefore, the input feature maps of an image can be denoted as $I^{(c_i)}_{(y,x)}$. The output maps of feature with a number of output channel ch_{out} by applying a filters bank $F^{(c_i,c_o)}$ is denoted by $O^{(c_o)}_{(y,x)}$ in Equation (2). Although computationally intensive, the mathematical operations of convolutional layers are not complex and offer many opportunities for data reuse and pipelining.

$$CONV_{(x,y,k)} = \sum_{j=-[k/2]}^{k/2} \sum_{i=-[k/2]}^{k/2} I_{y-j,x-i} \cdot F_{j,i}, \tag{1}$$

$$O^{(c_o)}_{(y,x)} = \sum_{0}^{ch_{in}-1} \left(\sum_{j=-[k/2]}^{k/2} \sum_{i=-[k/2]}^{k/2} I^{(c_i)}_{(y-j,x-i)} \cdot F^{(c_o)}_{(j,i)} \right). \tag{2}$$

To optimize the performance of the PlantNet accelerator, it is crucial to utilize the different available resources on the FPGA SoC board, including DSP slices, block memories, and other components. In light of this, we introduce the PlantNet algorithm, which will be accelerated using the Xilinx High-Level Synthesis (HLS) tool. The proposed algorithm organizes the loops in a specific order: layer, height, width, input channels, output channels, and kernel elements. Within each layer, the outer loops traverse the pixels from left to right and top to bottom. At each pixel position, the algorithm focuses on one input channel at a time, calculating and accumulating the corresponding output channels. This systematic organization of loops ensures efficient computation and optimal utilization of resources. The detailed algorithm can be found in Algorithm 1.

Our suggested PlantNet model will be accelerated using the Xilinx Vivado High-Level Synthesis (HLS) tool, which transforms C, C++, or SystemC sources into RTL implementations. The ZYNQ board is capable of synthesizing and implementing this design. A C-based testbench has been developed to assess the functioning of the deployed function and test the inputs into the HLS process. To do this, we will need a gold standard to evaluate the results of the synthesized function, which might be a predefined set of output values or an element of the testbench.

After going through the process of HLS synthesis, the internal hardware layout of the PlantNet accelerator may be seen in a high-level overview in Figure 6, in this case, upon the completion of the synthesis process using the HLS tool. The cost of the accelerated PlantNet model's hardware is shown in Table 5. At a working frequency of 150 MHz, the PlantNet accelerator uses up 30 % of the Block RAM, 6 % of the DSP, 4 % of the flip-flops (FF), and 7 % of the LUTs. In addition to the cost, the number of cache memory blocks is required, knowing that our image size of 256 × 256 × 3 and DDR3 memory with a capacity of 512 MB and a 16-bit bus (PYNQ-Z1). The memory size required for one image is approximately 192 kB. While, the available memory size in DDR3 memory is 512 MB, but considering the 16-bit bus width, the effective memory size is approximately 256 MB. Dividing the effective memory size by the memory size per frame gives a value of approximately 1 393.67 blocks of cache memory. Since we cannot have a fraction of a memory block, we round to the nearest integer, which gives 1 393 cache memory blocks.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	10	24	520	–
Expression	–	26	0	953
FIFO	5	50	165	–
Register	12	70	1 826	–
Memory	50	–	40	–
Multiplexer	4	30	–	570
Instance	2	5	1 561	2 381
Total	83	15	3 387	3 904
Available	280	220	106 400	53 200
Utilization (%)	30	6	4	7

Table 5. PlantNet IP hardware cost

Algorithm 1: Deep PlantNet-based Accelerator for HLS tool

```
Input: Layer Configs, Image $I$, Weights $W$.
Output: Images Classes.
BEGIN
for the whole model do
    for Each layer do
        Load Output_width $w_{out}$ and Output_height $h_{out}$
        Load Input_channels $ch_{in}$ and Output_channel $ch_{out}$
        Load Kernel_size $k$ and stride_length $s$
        Load Indicator $A$ to designate layer types: $A = 10$ for split1 layer and $A = 01$ for
          split2 layer.
        Feat_maps = L
        for $y = 0$ to $h_{out} - 1$ do
            for $x = 0$ to $w_{out} - 1$ do
                for $c_i = 0$ to $ch_{in} - 1$ do
                    for $c_o = 0$ to $ch_{out} - 1$ do
                        Conv_Product = 0
                        for $j = -k/2$ to $k/2$ do
                            for $i = -k/2$ to $-k/2$ do
                                Image_pixel = input [L, $s.y + j$, $s.x + i$, $c_i$]
                                Filter_pixel = W [L, $c_i$, $c_o$, $j$, $i$]
                                Conv_Product = Conv_Product+Filter_pixel*Image_pixel
                            END for
                        END for
                        if split2 then
                          Feat_maps[L, y, x, $c_o + ch_{out}$] = Feat_maps[L, y, x,
                          $c_o + ch_{out}$] + Conv_Product
                        else
                          Feat_maps[L, y, x, $c_o$] = Feat_maps[L, y, x, $c_o$] + Conv_Product
                        END if
                    END for
                END for
                for $c_o = 0$ to $ch_{out} - 1$ do
                    Feat_maps $[L, y, x, c_o] = ReLU(Feat\_maps[L, y, x, c_o] + W[L, bias, c_o])$
                END for
            END for
            if split1 then
              Feat_maps $input[L + 1, \dots] =$ Feat_maps $input[L, \dots]$
              Feat_maps $output[L + 1, \dots] =$ Feat_maps $output[L, \dots]$
            else
              Feat_maps $input[L + 1, \dots] =$ Feat_maps $output[L, \dots]$
            END if
            for $c_o = 0$ to $ch_{out} - 1$ do
                Feat_maps $output[layers, 0, 0, c_o] =$
                  $\sum_{y,x}$ Feat_maps $input[layers, y, x, c_o] \cdot \frac{1}{h_{out} \cdot w_{out}}$
            END for
        END for
        Layers classes = Softmax(Feat_maps output[layers, ...])
    END for
END for
```

5 HW/SW PLANTNET ARCHITECTURE FOR THE PLANT DISEASE DETECTION

The IP created by the HLS till must be integrated into an HW/SW architecture for the FPGA SoC which has been realized by the Xilinx Vivado Design Suite. After importing the exported "IP Catalog" as a new repository file into the Vivado cata-

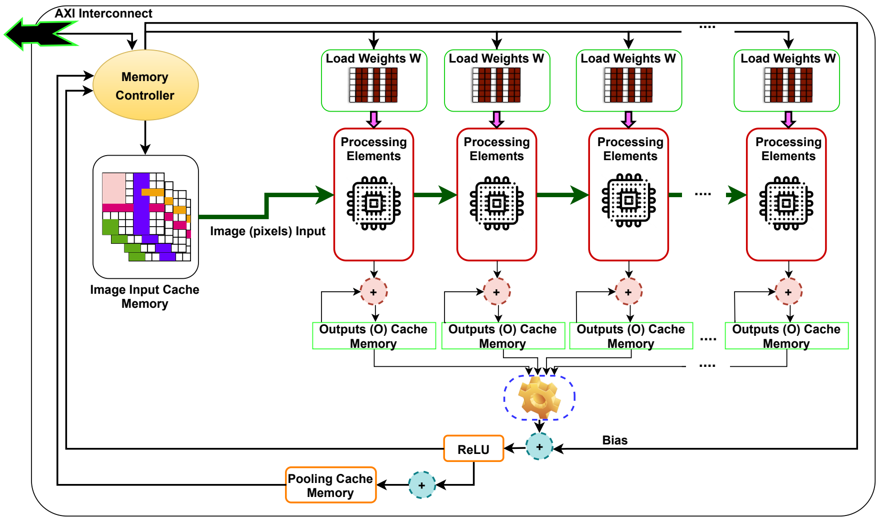


Figure 6. PlanNet accelerator

log, the block diagram of the design is created and the accelerator design is added as a new IP element. As shown in Figure 7, the HW/SW architecture for plant disease detection based on the PlantNet accelerator is composed of two main elements: the ZYNQ processing system (PS) which includes ARM Cortex dual cores and the programmable logic part (PS). The PS IP has been added and configured, before making connection between all IP components automatically. In this context, the m_axi and s_axilite interfaces are used in our conception, to reduce the interconnection of the design into a larger system architecture. The AXI-Master (M00_AXI) interface is used to connect to the memory via the AXI bus. The AXI-Lite interconnection is used for configuring, starting and stopping the PlantNet accelerator. The block RST_processing_System provides a reset action for the whole system while the IP processing system AXI_periph is used to route all transaction between FPGA fabric and PS system. In addition, the PS is used to communicate with the PL part and process the softmax layer. This is done via the Jupyter Notebook Overlay interface that offers the possibility to test the application by entering test data. After the allocation of the physical memory addresses and the validation of the design, the implementation steps which cover the synthesis and implementation levels will be launched. The purpose is to convert RTL components to Netlist before determining the best balance of runtime optimization, area optimization, routing optimization, and so on. After that, the optimized architecture is saved in the bitstream file.

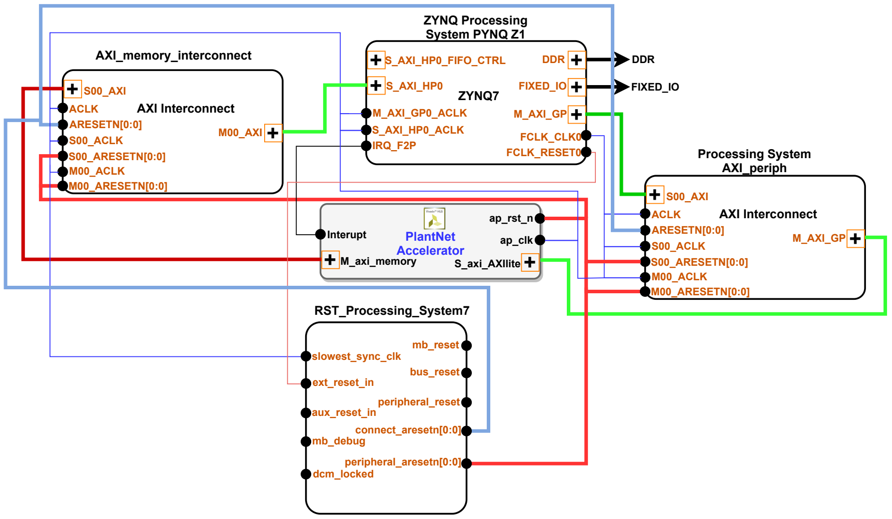


Figure 7. HW/SW PlantNet architecture

6 RESULTS AND DISCUSSION

The costs associated with the PYNQ-Z1 board's hardware are outlined in Table 6 for the prototype that was recommended based on the accelerated PlantNet model. It is clear from this that the proposed architecture makes use of around 7 % of LUTs, 20 % of LUTRAMs, 6 % of FFs, 32 % of BRAMs, 7 % of DSPs, and 3 % of BUFGs. However, with the Vivado IPI power analysis tool, it was determined that the power consumption of the design is 2.485 W. In addition, using the bitstream file that was generated together with the weights and biases that were saved, we generate our very own bespoke overlay in order to carry out the implementation of the hybrid architecture on the PYNQ-Z1 board. At this stage, we will test the application and determine how long it takes to execute a photograph once it has been processed. As a consequence of this, we were able to determine the mean inference time per frame for the accelerated PlantNet model by using a pseudo-code implementation on the ARM processor of the FPGA. It has been determined that acceleration has been achieved because the time it takes to process a photograph is around 0.04 seconds. In addition, by using Equations (3) and (4), we determine the maximum computational roof as well as its maximum bandwidth roof [48].

$$Computation\ roof = 2 \times \frac{N_{DSP}}{N_{R_{DSP}}} \times f = 33\,\text{GFLOPS}, \tag{3}$$

$$Bandwidth\ roof = \frac{64}{8} \times N_{HP} \times f = 1.9\,\text{GByte/s}. \tag{4}$$

Resource	Utilization	Available	Utilization (%)
LUT	3 700	53 200	6.9
LUTRAM	3 407	17 400	19.9
FF	5 989	106 400	5.6
BRAM	90.50	280	32.3
DSP	16	220	7.2
BUFG	1	32	3.13

Table 6. Hardware cost for HW/SW architecture

The suggested architecture has a maximum processing capacity of about 33 GFLOPS and a maximum bandwidth of approximately 1.9 GB/s. The PYNQ-Z1 board has a total of N_{DSP} digital signal processors (DSPs). The highest frequency of the system, which is about 120 MHz, is denoted by f, and the minimum number of digital signal processors, denoted by $N_{R_{DSP}}$, is reported as the value. The value of the N_{HP} variable represents the total number of high-performance ports. The design that has been offered has also been contrasted with many relevant architectures that have been proposed in related publications. The needed references are included, as well as an illustration of the performance comparison, in Table 7. The purpose of the suggested study that is mentioned in [48] is to develop a CNN model by hastening the process of depth-separable convolution in addition to regular convolution. The Xilinx ZYNQ 7100 board was used in the implementation of the design. In contrast to our designs, which are built on the XC7Z020 board, the implementation results demonstrate a greater hardware cost occupancy. This is the case despite the fact that our designs are used. In addition, our solutions were able to achieve a lower power consumption of 2.48 watts and a higher computational roof of 33 GFLOPS in comparison to the identical workload, which required 3.99 watts and 17.11 GFLOPS, as well as a higher operating frequency of 120 MHz as opposed to 100 MHz. In addition, the proposed vision system has the capability of processing one image per 4 800 000 clock cycles, with the clock period being around 8.33 nanoseconds.

	[48]	[49]	Our Design
Platform	ZYNQ 7100	XC7VX690T	XC7Z020
LUTs	51 %	62.9 %	6.9 %
BRAMs	46 %	50.2 %	32.3 %
FFs	38 %	26.6 %	5.6 %
DSPs	95 %	99.8 %	7.2 %
Power (W)	3.99	15.8	2.485
Frequency (MHz)	100	120	120
Clock Cycle (nanoseconds)	–	–	8.33333
Number of Clock Cycles (per image)	–	–	4 800 013
Computational roof (GFLOPS)	17.11	–	33
Bandwidth roof (GByte/s)	3.2	–	1.9

Table 7. Comparative study

This is because the solutions that we have provided are optimized with high-performance connections in order to speed up the data transfers. The work that was suggested was compared to the standard design that was mentioned in [49] in order to further investigate the efficacy of our collaborative designs. The latter intends to increase the processing speed of 2D and 3D CNNs running on the Xilinx VC709 platform. Nevertheless, it is interesting that the proposed design uses practically all of the available resources on the FPGA (99.8 % of DSPs, 60 % of LUTs, 50 % of FFs, and 26.6 % of BRAMs) while having a greater power usage (15.8 watts) than ours does. Based on this comparison with methods that are considered to be state-of-the-art, our proposed design is superior to the methods that are currently being used since it delivers excellent performance in terms of low power consumption, low hardware resource occupancy on the FPGA, more processing, and bandwidth cap. Our designs were able to achieve the ideal trade-off between resource cost, power consumption, bandwidth, and computatational roof as a consequence of this, which made them suitable for deployment on embedded devices with constrained resource budgets.

On the other hand, it is critical to normalize the findings based on the unique hardware specifications to ensure a fair comparison across different hardware platforms utilized to evaluate the proposed approach. Depending on the nature of the task and the available hardware resources, one common approach is to normalize performance metrics such as execution time or throughput by the number of GFLOPS or the bandwidth limit of the memory interface in GByte/s. However, because different hardware platforms have different GFLOPS and memory bandwidth, we can normalize the execution time or throughput by dividing it by the number of GFLOPS or memory bandwidth of each platform. This normalization approach can assist to reduce the impact of hardware heterogeneity and give a more realistic comparison of the performance of various hardware platforms. In this context, to normalize the measured performance of each platform (from Table 7) by dividing it by the corresponding maximum achievable performance. This will give us a value between 0 and 1, which represents the percentage of the maximum achievable performance that the platform is able to achieve.

- For the first platform (ZYNQ 7100):
  - Normalized GFLOPS = 17.11 GFLOPS/17.11 GFLOPS = 1,
  - Normalized Bandwidth = 3.2 GByte/s/3.2 GByte/s = 1.
- For the second platform (XC7Z020):
  - Normalized GFLOPS = 33 GFLOPS/17.11 GFLOPS = 1.93,
  - Normalized Bandwidth = 1.9 GByte/s/3.2 GByte/s = 0.59.

By comparing the normalized performance values, of both platforms to make a fair comparison, we can see that the XC7Z020 platform (used in our work) has a higher normalized GFLOPS value (1.93) than the ZYNQ 7100 platform (1). This indicates that it can achieve a higher percentage of its maximum achievable GFLOPS performance. However, it has a lower normalized bandwidth value (0.59), indicating that it may be limited in its ability to transfer data efficiently.

7 CONCLUSION

On the basis of the PlantNet model, a hybrid architecture for plant disease detection is developed. The concept was initially executed on the GPU platform, where training and test results demonstrate its usefulness. Following that, a HW/SW architecture is shown to accelerate data streaming and processing on the PYNQ Z1 platform. The implementation findings reveal that the suggested design outperforms all others in terms of processing time, computational roof, and bandwidth roof. The quantization approach and Xilinx's new Vitis tool will be utilized in future work to use the new deep learning processor on new embedded platforms.

Declarations

The authors have no relevant financial or non-financial interests to disclose. The authors have no competing interests to declare that are relevant to the content of this article.

Acknowledgments

The authors gratefully acknowledge the approval and the support of this research study by the grant No. CSCR-2022-11-1144 from the Deanship of Scientific Research at Northern Border University, Arar, K.S.A.

REFERENCES

[1] Zhai, Z.—Martínez, J. F.—Beltran, V.—Martínez, N. L.: Decision Support Systems for Agriculture 4.0: Survey and Challenges. Computers and Electronics in Agriculture, Vol. 170, 2020, Art. No. 105256, doi: 10.1016/j.compag.2020.105256.

[2] Megeto, G. A. S.—da Silva, A. G.—Bulgarelli, R. F.—Bublitz, C. F.—Valente, A. C.—da Costa, D. A. G.: Artificial Intelligence Applications in the Agriculture 4.0. Revista Ciência Agronômica, Vol. 51, 2020, Art. No. e20207701, doi: 10.5935/1806-6690.20200084.

[3] Symeonaki, E.—Arvanitis, K.—Piromalis, D.: A Context-Aware Middleware Cloud Approach for Integrating Precision Farming Facilities into the IoT Toward Agriculture 4.0. Applied Sciences, Vol. 10, 2020, No. 3, Art. No. 813, doi: 10.3390/app10030813.

[4] Symeonaki, E. G.—Arvanitis, K. G.—Piromalis, D. D.: Current Trends and Challenges in the Deployment of IoT Technologies for Climate Smart Facility Agricul-

ture. International Journal of Sustainable Agricultural Management and Informatics, Vol. 5, 2019, No. 2-3, pp. 181–200, doi: 10.1504/IJSAMI.2019.101673.

[5] Ahmed, K.—Shahidi, T. R.—Alam, S. M. I.—Momen, S.: Rice Leaf Disease Detection Using Machine Learning Techniques. 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1–5, doi: 10.1109/STI47673.2019.9068096.

[6] Hughes, D.—Salathé, M.: An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics. CoRR, 2015, doi: 10.48550/arXiv.1511.08060.

[7] Kianat, J.—Khan, M. A.—Sharif, M.—Akram, T.—Rehman, A.—Saba, T.: A Joint Framework of Feature Reduction and Robust Feature Selection for Cucumber Leaf Diseases Recognition. Optik, Vol. 240, 2021, Art. No. 166566, doi: 10.1016/j.ijleo.2021.166566.

[8] Sharma, R.: Artificial Intelligence in Agriculture: A Review. 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 937–942, doi: 10.1109/ICICCS51141.2021.9432187.

[9] Panigrahi, K. P.—Das, H.—Sahoo, A. K.—Moharana, S. C.: Maize Leaf Disease Detection and Classification Using Machine Learning Algorithms. In: Das, H., Pattnaik, P. K., Rautaray, S. S., Li, K. C. (Eds.): Progress in Computing, Analytics and Networking (ICCAN 2019). Springer Singapore, Advances in Intelligent Systems and Computing, Vol. 1119, 2020, pp. 659–669, doi: 10.1007/978-981-15-2414-1_66.

[10] Kartikeyan, P.—Shrivastava, G.: Review on Emerging Trends in Detection of Plant Diseases Using Image Processing with Machine Learning. International Journal of Computer Applications, Vol. 174, 2021, No. 11, pp. 39–48, doi: 10.5120/ijca2021920990.

[11] Sladojevic, S.—Arsenovic, M.—Anderla, A.—Culibrk, D.—Stefanovic, D.: Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. Computational Intelligence and Neuroscience, Vol. 2016, 2016, Art. No. 3289801, doi: 10.1155/2016/3289801.

[12] Liu, J.—Wang, X.: Plant Diseases and Pests Detection Based on Deep Learning: A Review. Plant Methods, Vol. 17, 2021, No. 1, Art. No. 22, doi: 10.1186/s13007-021-00722-9.

[13] LeCun, Y.—Bottou, L.—Bengio, Y.—Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, Vol. 86, 1998, No. 11, pp. 2278–2324, doi: 10.1109/5.726791.

[14] Lee, S. H.—Chan, C. S.—Wilkin, P.—Remagnino, P.: Deep-Plant: Plant Identification with Convolutional Neural Networks. 2015 IEEE International Conference on Image Processing (ICIP), 2015, pp. 452–456, doi: 10.1109/ICIP.2015.7350839.

[15] Grinblat, G. L.—Uzal, L. C.—Larese, M. G.—Granitto, P. M.: Deep Learning for Plant Identification Using Vein Morphological Patterns. Computers and Electronics in Agriculture, Vol. 127, 2016, pp. 418–424, doi: 10.1016/j.compag.2016.07.003.

[16] Mohanty, S. P.—Hughes, D. P.—Salathé, M.: Using Deep Learning for Image-Based Plant Disease Detection. Frontiers in Plant Science, Vol. 7, 2016, Art. No. 1419,

doi: 10.3389/fpls.2016.01419.

[17] Hasan, R. I.—Yusuf, S. M.—Alzubaidi, L.: Review of the State of the Art of Deep Learning for Plant Diseases: A Broad Analysis and Discussion. Plants, Vol. 9, 2020, No. 10, Art. No. 1302, doi: 10.3390/plants9101302.

[18] Tan, L.: Cloud-Based Decision Support and Automation for Precision Agriculture in Orchards. IFAC-PapersOnLine, Vol. 49, 2016, No. 16, pp. 330–335, doi: 10.1016/j.ifacol.2016.10.061.

[19] Selmani, A.—Oubehar, H.—Outanoute, M.—Ed-Dahhak, A.—Guerbaoui, M.—Lachhab, A.—Bouchikhi, B.: Agricultural Cyber-Physical System Enabled for Remote Management of Solar-Powered Precision Irrigation. Biosystems Engineering, Vol. 177, 2019, pp. 18–30, doi: 10.1016/j.biosystemseng.2018.06.007.

[20] Messaoud, S.—Bradai, A.—Bukhari, S. H. R.—Quang, P. T. A.—Ahmed, O. B.—Atri, M.: A Survey on Machine Learning in Internet of Things: Algorithms, Strategies, and Applications. Internet of Things, Vol. 12, 2020, Art. No. 100314, doi: 10.1016/j.iot.2020.100314.

[21] Maraoui, A.—Messaoud, S.—Bouaafia, S.—Ammari, A. C.—Khriji, L.—Machhout, M.: PYNQ FPGA Hardware Implementation of LeNet-5-Based Traffic Sign Recognition Application. 2021 18th International Multi-Conference on Systems, Signals & Devices (SSD), 2021, pp. 1004–1009, doi: 10.1109/SSD52085.2021.9429480.

[22] Liang, S.—Yin, S.—Liu, L.—Luk, W.—Wei, S.: FP-BNN: Binarized Neural Network on FPGA. Neurocomputing, Vol. 275, 2018, pp. 1072–1086, doi: 10.1016/j.neucom.2017.09.046.

[23] Shawahna, A.—Sait, S. M.—El-Maleh, A.: FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review. IEEE Access, Vol. 7, 2019, pp. 7823–7859, doi: 10.1109/ACCESS.2018.2890150.

[24] Jokic, P.—Emery, S.—Benini, L.: BinaryEye: A 20 kfps Streaming Camera System on FPGA with Real-Time On-Device Image Recognition Using Binary Neural Networks. 2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES), 2018, pp. 1–7, doi: 10.1109/SIES.2018.8442108.

[25] Sharma, A.—Singh, V.—Rani, A.: Implementation of CNN on Zynq Based FPGA for Real-Time Object Detection. 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019, pp. 1–7, doi: 10.1109/ICCCNT45670.2019.8944792.

[26] Roberts, M. J.—Schimmelpfennig, D.—Ashley, E.—Livingston, M.—Ash, M.—Vasavada, U.: The Value of Plant Disease Early-Warning Systems: A Case Study of USDA's Soybean Rust Coordinated Framework. USDA Economic Research Service, 2006.

[27] Pimentel, D.—Zuniga, R.—Morrison, D.: Update on the Environmental and Economic Costs Associated with Alien-Invasive Species in the United States. Ecological Economics, Vol. 52, 2005, No. 3, pp. 273–288, doi: 10.1016/j.ecolecon.2004.10.002.

[28] Liu, B.—Zhang, Y.—He, D.—Li, Y.: Identification of Apple Leaf Diseases Based on Deep Convolutional Neural Networks. Symmetry, Vol. 10, 2018, No. 1, Art. No. 11, doi: 10.3390/sym10010011.

[29] Sharif, M.—Khan, M. A.—Iqbal, Z.—Azam, M. F.—Lali, M. I. U.—Javed, M. Y.: Detection and Classification of Citrus Diseases in Agriculture Based on Optimized Weighted Segmentation and Feature Selection. Computers and Electronics in Agriculture, Vol. 150, 2018, pp. 220–234, doi: 10.1016/j.compag.2018.04.023.

[30] Zhang, S.—You, Z.—Wu, X.: Plant Disease Leaf Image Segmentation Based on Superpixel Clustering and EM Algorithm. Neural Computing and Applications, Vol. 31, 2019, No. Suppl 2, pp. 1225–1232, doi: 10.1007/s00521-017-3067-8.

[31] Sanga, S.—Mero, V.—Machuve, D.—Mwanganda, D.: Mobile-Based Deep Learning Models for Banana Diseases Detection. 2020, doi: 10.48550/arXiv.2004.03718.

[32] Chohan, M.—Khan, A.—Chohan, R.—Katpar, S. H.—Mahar, M. S.: Plant Disease Detection Using Deep Learning. International Journal of Recent Technology and Engineering, Vol. 9, 2020, No. 1, pp. 909–914, doi: 10.35940/ijrte.A2139.059120.

[33] Kim, W. S.—Lee, D. H.—Kim, Y. J.: Machine Vision-Based Automatic Disease Symptom Detection of Onion Downy Mildew. Computers and Electronics in Agriculture, Vol. 168, 2020, Art. No. 105099, doi: 10.1016/j.compag.2019.105099.

[34] Rothe, P. R.—Kshirsagar, R. V.: Cotton Leaf Disease Identification Using Pattern Recognition Techniques. 2015 International Conference on Pervasive Computing (ICPC), 2015, pp. 1–6, doi: 10.1109/PERVASIVE.2015.7086983.

[35] Islam, M.—Dinh, A.—Wahid, K.—Bhowmik, P.: Detection of Potato Diseases Using Image Segmentation and Multiclass Support Vector Machine. 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), 2017, pp. 1–4, doi: 10.1109/CCECE.2017.7946594.

[36] Dahl, G. E.—Yu, D.—Deng, L.—Acero, A.: Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. IEEE Transactions on Audio, Speech, and Language Processing, Vol. 20, 2012, No. 1, pp. 30–42, doi: 10.1109/TASL.2011.2134090.

[37] Fadhel, M. A.—Al-Shamma, O.—Oleiwi, S. R.—Taher, B. H.—Alzubaidi, L.: Real-Time PCG Diagnosis Using FPGA. In: Abraham, A., Cherukuri, A. K., Melin, P., Gandhi, N. (Eds.): Intelligent Systems Design and Applications (ISDA 2018). Springer, Cham, Advances in Intelligent Systems and Computing, Vol. 940, 2018, pp. 518–529, doi: 10.1007/978-3-030-16657-1_48.

[38] Paramesh, V.: DLAU: A Scalable Deep Learning Accelerator Unit on FPGA. Technical Report. 2019, `https://easychair.org/publications/preprint_download/ckW5`.

[39] Zhang, C.—Li, P.—Sun, G.—Guan, Y.—Xiao, B.—Cong, J.: Optimizing FPGA-Based Accelerator Design for Deep Convolutional Neural Networks. Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '15), 2015, pp. 161–170, doi: 10.1145/2684746.2689060.

[40] Bäuerle, A.: Visualization-Based Neural Network Introspection. Ph.D. Thesis. Universität Ulm, 2023, doi: 10.18725/OPARU-47825.

[41] Guo, Y.—Liu, Y.—Bakker, E. M.—Guo, Y.—Lew, M. S.: CNN-RNN: A Large-Scale Hierarchical Image Classification Framework. Multimedia Tools and

Applications, Vol. 77, 2018, No. 8, pp. 10251–10271, doi: 10.1007/s11042-017-5443-x.

[42] Emmanuel, T. O.: PlantVillage Dataset. Kaggle, 2018, `https://www.kaggle.com/emmarex/plantdisease`.

[43] Hidayatuloh, A.—Nursalman, M.—Nugraha, E.: Identification of Tomato Plant Diseases by Leaf Image Using Squeezenet Model. 2018 International Conference on Information Technology Systems and Innovation (ICITSI), IEEE, 2018, pp. 199–204, doi: 10.1109/ICITSI.2018.8696087.

[44] Restrepo-Arias, J. F.—Branch-Bedoya, J. W.—Awad, G.: Plant Disease Detection Strategy Based on Image Texture and Bayesian Optimization with Small Neural Networks. Agriculture, Vol. 12, 2022, No. 11, Art. No. 1964, doi: 10.3390/agriculture12111964.

[45] Shin, J.—Chang, Y. K.—Heung, B.—Nguyen-Quang, T.—Price, G. W.—Al-Mallahi, A.: A Deep Learning Approach for RGB Image-Based Powdery Mildew Disease Detection on Strawberry Leaves. Computers and Electronics in Agriculture, Vol. 183, 2021, Art. No. 106042, doi: 10.1016/j.compag.2021.106042.

[46] Abbas, I.—Liu, J.—Amin, M.—Tariq, A.—Tunio, M. H.: Strawberry Fungal Leaf Scorch Disease Identification in Real-Time Strawberry Field Using Deep Learning Architectures. Plants, Vol. 10, 2021, No. 12, Art. No. 2643, doi: 10.3390/plants10122643.

[47] Bobrowicz, S.: PYNQ-Z1 Reference Manual – Digilent Reference. `https://digilent.com/reference/programmable-logic/pynq-z1/reference-manual`.

[48] Liu, B.—Zou, D.—Feng, L.—Feng, S.—Fu, P.—Li, J.: An FPGA-Based CNN Accelerator Integrating Depthwise Separable Convolution. Electronics, Vol. 8, 2019, No. 3, Art. No. 281, doi: 10.3390/electronics8030281.

[49] Liu, Z.—Chow, P.—Xu, J.—Jiang, J.—Dou, Y.—Zhou, J.: A Uniform Architecture Design for Accelerating 2D and 3D CNNs on FPGAs. Electronics, Vol. 8, 2019, No. 1, Art. No. 65, doi: 10.3390/electronics8010065.

Taoufik Saidani is currently Assistant Professor in the Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Rafha, Kingdom of Saudi Arabia. He received his Ph.D. degree in computer science and engineering from the Faculty of Sciences, Monastir University, Tunisia, in 2014. His research interests include real-time image and video processing, embedded system, high level synthesis, machine learning, deep learning, image compression, JPEG2000, FPGA accelaration of image and processing system.

Refka Ghodhbani is currently serving as Assistant Professor in the Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Rafha, Kingdom of Saudi Arabia. She received her Ph.D. degree in computer science and engineering from the Faculty of Sciences, Monastir University, Tunisia, in 2021. Her research interests include real-time image and video processing, embedded system, high level synthesis, machine learning, deep learning, image compression, JPEG2000, FPGA accelaration of image and processing system.

Ahmed Alhomoud is Assistant Professor in the Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Rafha, Kingdom of Saudi Arabia. He received his Ph.D. in computer science from the University of Southampton, United Kingdom. His research interests include but are not limited to digital forensics, cyber security, Internet of Things and blockchain.

Mohamed Ben Ammar is currently serving as Assistant Professor in the Department of Information Systems at the Faculty of Computing and Information Technology, Northern Border University, Rafha, Kingdom of Saudi Arabia. He received his Ph.D. degree in engineering of information systems from the National Engineering School of Sfax (ENIS), Sfax University, Tunisia. His research interests include affective computing, intelligent tutoring systems, and multimodal emotion recognition.