Embedded Plant Disease Recognition using Deep PlantNet on FPGA-SoC

Taoufik Saidani (Taoufik.Saidan@nbu.edu.sa)
Northern Border University

Refka Ghodhbani
Northern Border University

Research Article

Keywords: Deep CNN, FPGA, Acceleration, Co-design, PYNQ-Z1

Posted Date: October 3rd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2107827/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Embedded Plant Disease Recognition using Deep PlantNet on FPGA-SoC

Taoufik Saidani, Refka Ghodhbani

Abstract. During the past decades, global technological advancement has allowed a general revolution in many sectors such as agriculture. Therefore, we are talking about Agriculture 4.0 that focuses on strategy and systems, leaving behind the tasks of the past and making way for a new generation of intelligent machines. In this context, crop process production management, in Agriculture 4.0, is a very challenging task for timely and accurate diagnosis of crop diseases. Within this range, plant diseases are the main problems that lead to a significant reduction in the quality and quantity of crop production. To solve this problem, early diagnosis of plant diseases using deep learning neural network models, which are able to extract features automatically, obtain high-dimensional features from low-dimensional features, and achieve better learning outcomes, plays an important role in improving agricultural crops. In this paper, we take up the challenge of establishing a collaborative solution between image processing and phytopathology. This solution will reduce the human labor time required by the use of algorithms to facilitate the identification of plant diseases. Through this work, we built a deep transfer learning model to detect plant diseases from infected leaves images. Thus, given that its performance depends on data volume and quality they assimilate to learn and improve, the implementation of AI techniques on embedded systems can drastically reduce energy consumption and processing times while reducing the costs and risks associated with data transmission. Therefore, the second aim of this paper is to implement the proposed vision system on an embedded...
After discussing and analyzing the research status of deep learning methods applied in these three areas, and the difficulties and challenges faced at present, prospective comments on the technology bottlenecks in this field.

**Keywords** Deep CNN, FPGA, Acceleration, Co-design, PYNQ-Z1

1 Introduction

The technologies of agricultural domain are quickly evolving towards a new paradigm, namely as, 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 is considered as one of the main sectors of the industry which, according to the Food and Agriculture Organization of the United Nations (FAO), faces the challenge of increasing its productivity up to 60% during the century of 21st to provide an adequate food supply to the growing population in the world [3]. This objective should be achieved while considering into account the need to use a natural resources in accordance with sustainable techniques and strategies due to the increasing pressures on ecosystems, biodiversity, land, and water. The adoption of precision agriculture (FP) practices providing ubiquitous conceptual innovations and computer advancements from “smart” agricultural production towards agriculture 4.0 is an important factor for the h benefits [4]. Knowing that in recent years, trade development and globalization, as well as the changing of the climate, leading to an increase in the plant diseases. This has reached the level of an epidemic, in many countries, which led to lose many crops and thus threaten 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. An example of emblematic is the Irish potato famine of 1845-1849 that killed 1.2 million people [6]. Many common plant diseases are summarized in Table. 1. Plants diseases can be divided systematically into hyphomycete, fungal, bacterial and viral types. Some pictures of common diseases of plants are depicted in Figure. 1. Farmers and researchers are still exploring how to expand a smart and efficient method to classify plant diseases. Many researchers have tested several techniques on plant samples, such as enzyme immunoassay, polymerase chain reaction, and
loop-mediated isothermal amplification, which are very specific and sensitive to identify the disease [7].

### Table 1: Common plants diseases

<table>
<thead>
<tr>
<th>Plant</th>
<th>Fungal</th>
<th>Bacterial</th>
<th>Viral</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cucumber</td>
<td>Downy mildew, powdery mildew, gray mold, black spot, anthracnose</td>
<td>Angular spot, brown spot, target spot</td>
<td>Mosaic virus, yellow spot virus</td>
</tr>
<tr>
<td>Rice</td>
<td>Rice-ripe blight, false smut, rice blight</td>
<td>Bacterial leaf blight, bacterial leaf streak</td>
<td>Rice leaf smut, rice black-streaked dwarf virus</td>
</tr>
<tr>
<td>Maize</td>
<td>Leaf spot disease, rust disease, gray leaf spot</td>
<td>Bacterial stalk rot, bacterial leaf streak</td>
<td>Rough dwarf disease, crimson leaf disease</td>
</tr>
<tr>
<td>Tomato</td>
<td>Early blight, late blight, leaf mold</td>
<td>Bacterial wilt, soil rot, canker</td>
<td>Tomato yellow leaf curl virus</td>
</tr>
</tbody>
</table>

![Fig. 1: Leaf spot in eight common plants](image)

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 in 2017 and is expected to grow by more than 22.5% per year to reach 2.6 billion 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 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. A 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 is carried out either by farmers or agronomists [10]. However, this is a very difficult and need 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 data raw, 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 a hybrid models. CNNs are one of the most powerful and effective scheme 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 CNNs 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. Further work related to this topic is presented in the Related Work section.

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 [17]. This, in order to provide more efficient and safer operating conditions for the both stakeholders and 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 [18]. For this, it becomes almost imperative to harness AI in vision applications that make these systems intelligent and able in making decisions close to or similar to ours [19]. In this context, AI integration poses many challenges, especially in algorithms optimization. However, the AI systems performance depends on data volume and quality they assimilate to learn and improve [20]. In addition to energy consumption and cost constraints, embedded systems, in particular the FPGA-SoC which is the most
widely used, have limited processing, memory, and communication capacity [21]. Despite this, accelerating the CNNs implementation using FPGA SoC has become a new solution, due to its capacity in maximizing parallelism data processing and power efficiency, while reducing the costs and risks associated with data transmission. Additionally, by taking reconfiguration benefit, different CNN models and architectures can be easily reconfigured in the FPGA for many application types [22]. In this context many research work has been proposed to implement vision systems on FPGA-SoC. Authors in [23] proposed an FPGA-based 20 kfp streaming camera system called BinaryEye that recognize interest region within real-time streaming mode. Authors, in [24], designed an FPGA-based hardware architecture for real-time object detection based on CNN. Authors in [20] proposed a hardware-software architecture to implement a deep traffic sign recognition application on FPGA SoC. More than work related to this topic is presented in the Related Work section.

This paper proposes a novel and hybrid FPGA architecture for smart disease detection in plant based on a deep learning approach with larger training parameters. Although, several techniques have been presented in the literature for automatic plant disease detection, a hybrid embedded system that combined 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. We organized this paper as follows: we reviewed the latest CNN networks to provide leaf disease classification in section 2. The proposed deep learning model and the GPU software training performance is introduced in section 3. Section 3.1 presents the embedded the HW/SW architecture of the plant decease detection. After that, we present the implementation results in section 3.2. Finally, the paper is concluded in section 4.

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 [25]. Bacterial, fungal, and viral infections, as well as insect infestations, lead to plant disease and damage. There are about 50,000 parasitic and non-parasitic plant diseases that infect plants in the United States. When infected, the plant shows symptoms that appear on different parts of the plants, which leads to a significant agricultural effect [26]. Many of these microbial diseases spread over time over a larger area in orchards and farms through the accidental introduction of vectors or through infected plant material [26]. Another route for the spread of pathogens is through ornamental plants that act as hosts. These plants are frequently sold through mass distribution before
the infection is known. An early disease detection system can help reduce these losses caused by plant diseases and can prevent the spread of diseases.

In this context, authors in [27] 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 [28], proposed a hybrid classification approach-based citrus diseases detection used 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 [29]. 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 [30], have proposed five different CNN architecture for a disease-based detection tool for the bananas plant. These models were ResNet-152, VGG-16, ResNet-50, InceptionV3, and ResNet-18. However, the ResNet-152 model outperformed the others architecture 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 with their smartphones. This application implemented the InceptionV3 model to predict the plant disease with 99% confidence. Other similar work was proposed in [31], which used the CNN VGG-19 and InceptionV3 architectures for automatic detection of plant diseases using the Plant-Village dataset. In their research, they also used data augmentation to artificially enlarge the dataset. The VGG-19 model outperformed the InceptionV3 model with 98% drive accuracy and 95% test accuracy as they claim in their article.

Authors in [32] proposed a Deep Convolutional Encoder Network system for seasonal crops disease identification. They considered 900 leaf images of three crops: potato, tomato, and maize, distributed in six classes. They achieved 100% training accuracy while the testing accuracy of their model was 86.78%. The authors in [33] identify a pattern of 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 a nervous 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 [34]. This automated method classifies diseases on potato plants of the "plant village", 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%.

Currently, several researchers have changed their ideas to implement scalable and parallel deep learning frameworks [35]. More recently, their idea was further modified to shift the learning task to GPUs. GPUs are infamous for their leakage currents, which in turn ignore any reasonable achievement of deep learning models on embedded devices [36]. Another solution is the use
of FPGAs. FPGAs have been used as deep learning accelerators to optimize data access pipelines for significantly improved results [37]. A scalable architecture is known as the Deep Learning Accelerator Unit (DLAU) was used in [38]. The DLAU uses three pipeline processing units. Compared to processors, they achieved 36.1 times faster with 234 mW power consumption using locality and tile techniques. Another approach achieved a detection rate of 97%, which used an architecture based on low-end FPGAs with leaks, arc losses, etc. Compared to the software implementation, they achieved 7.5 times faster processing speed.

These studies show that convolution neural networks have been widely applied to the field of recognition of crops and plants and have achieved good results. However, on the one hand, these studies apply only CNN-based models to identify diseases of crops and plants without implementing these models on a real embedded system. A new CNN-based technique 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 deep learning approach. To do so, the global contribution of this paper is subdivided into three phases, as depicted in Fig. 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 1), 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 crops disease.
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 that remain a critical issue. Therefore, we found the Netscope CNN analyzer [39] which represents a web-based tool written in CoffeeScript, 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 Inception v3, ResNet-50, VGG-16, SqueezeNet, and AlexNet. 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 also computed for each deep pretrained model to designate the the total pixel 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 [40]. The evaluation between these models is summarized in Tab. 2.

Table 2: CNN models Evaluation for Image Classification on ImageNet.

<table>
<thead>
<tr>
<th>Models</th>
<th>Cnv_layers</th>
<th>MACC(M)</th>
<th>Parameters (M)</th>
<th>Activation (M)</th>
<th>Top-5 Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16</td>
<td>16</td>
<td>15470</td>
<td>183.3</td>
<td>29</td>
<td>8.1</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>50</td>
<td>3870</td>
<td>25.5</td>
<td>46.7</td>
<td>7.0</td>
</tr>
<tr>
<td>Inception v3</td>
<td>48</td>
<td>5710</td>
<td>23.8</td>
<td>32.6</td>
<td>5.6</td>
</tr>
<tr>
<td>AlexNet</td>
<td>5</td>
<td>1140</td>
<td>60.7</td>
<td>2.4</td>
<td>19.7</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>18</td>
<td>861</td>
<td>1.2</td>
<td>12.5</td>
<td>19.7</td>
</tr>
</tbody>
</table>

Tab. 1 shows that SqueezeNet has the lowest computational complexities in terms of MACCs, followed by AlexNet. In addition, the least amount of Activation Memory are used for the same four CNNs. When looking at the number of Parameters, the SqueezeNet model scored the least numbers with 19.7% top-5 error. However, SqueezeNet was chosen to be the ground for our CNN model, due to its good suitability for an FPGA SoC-based implementation in terms of lower computational complexity and minimal activations and parameter sets that can fit into an FPGA SoC memory. To do so, three improvements types were implemented during the transformation of the original SqueezeNet, as depicted in Fig. 3, to PlantNet CNN, which are efficiency improvements, FPGA-related improvements, and accuracy improvements.

As denoted in Fig. 3, the SqueezeNet architecture consisted of an initial convolutional layer, eight stacked fire units, final convolutional layer, three pooling layers, and a nested dropout layer. These convolutional layers are activated by a nonlinear ReLU, while the structure is terminated by a global
mean pooling layer. However, each fire unit contain of a squeeze layer (convolution 1x1 and ReLU) and two parallel layers (convolutions 1x1 and 3x3 and ReLU). A few number of output channels for each squeeze layer which are responsible for internal representation compressing. For this feature map, the both 1x1 and 3x3 kernels are evaluated for the expand layer and their outputs are concatenated along the channel dimension.

As our goal is to rectify the SqueezeNet architecture to provide a PlantNet model that reach a minimum computational complexity to fit the FPGA implementation, Fig. 4 depicts the PlantNet model, the layer widths $w_{out}$, the layer capacities $w_{out} \times h_{out} \times c_{out}$, and the number of output channels $c_{out}$ in each network stage. However, the proposed PlantNet model consist 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 each PlantNet individual layer the computational complexity has been analyzed with Netscope, and then the results are presented in Tab. 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 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.

So the proposed plantNet model was trained on the PlantVillages dataset consisting of about 20,000 images [41] of healthy and diseased plants. The used dataset contains 15 sub-directories in which each of them contain 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 proposed deep learning model was trained and tested using the python programming language. The Adam optimizer technique is used as a powerful optimizer technique, which updates weights at each iteration and minimizes the gradient error between the ground truth labels and the prediction outputs.
Table 3: Layer’s PlantNet Topologie evaluation for Image Classification on ImageNet.

<table>
<thead>
<tr>
<th>Layers</th>
<th>MACC</th>
<th>Parameters</th>
<th>Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>28.31 Millions</td>
<td>1.79 K</td>
<td>1.05 Millions</td>
</tr>
<tr>
<td>fire1 (sub-module)</td>
<td>79.69 Millions</td>
<td>19.6 K</td>
<td>1.7 Millions</td>
</tr>
<tr>
<td>fire2 (sub-module)</td>
<td>79.69 Millions</td>
<td>78.11 K</td>
<td>851.97 K</td>
</tr>
<tr>
<td>fire3 (sub-module)</td>
<td>79.69 Millions</td>
<td>311.87 K</td>
<td>425.98 k</td>
</tr>
<tr>
<td>fire4 (sub-module)</td>
<td>39.85 Millions</td>
<td>156.1 K</td>
<td>327.68 k</td>
</tr>
<tr>
<td>fire5 (sub-module)</td>
<td>43.12 Millions</td>
<td>674.42 K</td>
<td>112.64 k</td>
</tr>
<tr>
<td>fire6 (sub-module)</td>
<td>30.05 Millions</td>
<td>470.35 K</td>
<td>155.65 k</td>
</tr>
<tr>
<td>drop6</td>
<td>——</td>
<td>——</td>
<td>47.1 K</td>
</tr>
<tr>
<td>conv2/split1, conv3/split2</td>
<td>48.23 Millions</td>
<td>754.69 k</td>
<td>65.54 k</td>
</tr>
<tr>
<td>conv4</td>
<td>——</td>
<td>——</td>
<td>65.54 k</td>
</tr>
<tr>
<td>pool1</td>
<td>——</td>
<td>——</td>
<td>1.02 k</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>428.64 Millions</strong></td>
<td><strong>2.47 Millions</strong></td>
<td><strong>6.05 Millions</strong></td>
</tr>
</tbody>
</table>

In addition, the overall experiments were implemented on Intel® core™ i7-3770@ 3.4 GHz CPU and 16 GB RAM. We also use the NVIDIA GeForce RTX 2070 GPU to improve the speed of the proposed trained deep model. The deep learning parameters applied in this work are; the number of epochs, the learning rate, and the batch size was experimentally set at 350, $10^{-4}$, and 64, respectively. The datasets are gathered into two sub-sets, 75% for training and 20% for testing, and 5% for validation including healthy and diseased plants. Fig. 5 shows the accuracy and loss curves of the training and validation process in order to evaluate the performance of the proposed PlantNet model. The achieved training and validation accuracy is about 97%. Similarly, the training and validation loss was is about 0.27 for the proposed model. Consequently, the proposed deep CNN model reaches a good classification performances.

![Model Accuracy](image1.png)

![Model Loss](image2.png)

**Fig. 5: PlanNet training performance.**

subsectionPlantNet model acceleration on FPGA SoC
The PlantNet model has been designed for high-performance plant disease detection in the agriculture domain. The embedded platform is based on the Xilinx PYNQ-Z1 SoC, which combines a dual-core ARM Cortex-A9 processor with programmable FPGA fabric in a single device. The PYNQ-Z1 includes a Xilinx Zynq XC7Z020-1CLG400C SoC, 512 MB DDR3 memory for the ARM processor, 4GB independent external SD memory, and plenty of connection options (USB, HDMI, Gigabit Ethernet, etc). The PYNQ-Z1 SoC includes 13.3k logic slices, 53.2k LUTs, 630 kB Block RAM, 106.4 k Flip-flops, and 220 DSP slices [42].

The deep PlantNet model consists of convolutional and average pooling layers, while the ReLU nonlinearities is used as an activation function. The network is very structured which proves the most layers arrangement in fire modules format. Each firing module consists of three convolutional layers: one compression layer and two expand layers. Then, the two expand layers channels output are concatenated to provide a single twice feature map as many output channels. While the dropout layer (drop6) is suitable only during the training phase and can be completely ignored during the inference phase. In this context, the concatenation capability of two layers is re-utilized the 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 1024 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 need acceleration on FPGA SoC. The 2DC for an input image represented as the result from filter sliding over the image, and compute 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×k, is denoted for each pixel (y, x) in the equation 1. Knowing that, for an RGB image there is three dimensions channel with an input channel \( ch_{in} \) so that the input feature maps of an image is \( I^{(c_i)}_{(y,x)} \). 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 the equation 2. Although computationally intensive, the convolutional layers
mathematical operations are not complex and offer many opportunities for data reuse and pipelining.

\[
CONV(x, y, k) = \sum_{j=\lceil-k/2\rceil}^{\lceil-k/2\rceil} \sum_{i=\lceil-k/2\rceil}^{\lceil-k/2\rceil} I_{y-j,x-i} \cdot F_{j,i}
\]

\[
O^{(c_o)}_{(y,x)} = \sum_{c_i=0}^{c_{in}-1}\sum_{j=\lceil-k/2\rceil}^{\lceil-k/2\rceil}\sum_{i=\lceil-k/2\rceil}^{\lceil-k/2\rceil} I^{(c_i)}_{(y-j,x-i)} \cdot O^{(c_o)}_{(j,i)}
\]

To achieve good performance, the PlantNet accelerator must use almost resources available on the FPGA SoC board, as well as the DSP slices, the Block memories, etc. In this context, we introduce the PlantNet algorithm, that will be accelerated using HLS tool of Xilinx, into the algorithm. In the proposed algorithm, the loops are organized in the order of layer, then height, width, input channels, output channels, and after that the kernel elements. For each layer, the outer loops run through all pixels from left to right and top to bottom. At each pixel position, one input channel after the other is focused, and all corresponding output channels are calculated and accumulated.

The Vivado High-Level Synthesis (HLS) that transforms a C, C++ or SystemC sources into an RTL implementation, is the Xilinx tool that will be used to accelerate our proposed PlantNet model. This design can be synthesised and implemented onto the ZYNQ board. Regarding the inputs into the HLS process which is a software function, with a C based testbench, which has been developed to test the deployed function and verify the functionality. This will involve a gold benchmark against which may take the form of a prepared set of output values or be part of the testbench itself with the aim is to test the outputs produced by the function intended for synthesis. The output files that include the design files for the desired RTL language are produced after the HLS process is complete. Once the design has been validated and the implementation has been achieved correctly, the last step is to package the output RTL. This can be done directly by using the RTL files created automatically by the HLS process, but it may be more convenient to use the functionality of Vivado HLS to package IP. Packaging the output produced by Vivado HLS means that HLS designs can be easily fed into other Xilinx tools, as well as Vivado IP Integrator that will be used for designing the whole design for plant disease detection.

An overview, is depicted by Fig. 6, of the internal hardware structure of the PlantNet accelerator after synthesizing using HLS. In this context, after the synthesis process by HLS tool, Table. 4 denotes the hardware cost of the accelerated PlantNet model. As depicted in this Table, the PlantNet accelerator occupies 30% of Block RAM, 6% of DSP, 4% of Flip Flops (FF), and 7% of LUTs with a working frequency of 150 Mhz.
Algorithm 1: Deep PlantNet-based Accelerator for HLS tool

Input: Layer Configs, Image $I$, Weights $W$.
Output: Images Classe.

1 BEGIN
2 for the whole model do
3   for Each layer do
4      Load Output width $w_{out}$ and Output height $h_{out}$
5      Load Input channels $c_{in}$ and Output channel $c_{out}$
6      Load Kernel size $k$ and stride_length $s$
7      Load Indicator $A$ to designate layer types: $A = 10$ for split1 layer and $A = 01$ for split2 layer.
8      Feat_maps = $L$
9      for $y = 0$ to $h_{out} - 1$ do
10         for $x = 0$ to $w_{out} - 1$ do
11             for $c_{i} = 0$ to $c_{in} - 1$ do
12                for $c_{o} = 0$ to $c_{out} - 1$ do
13                   Conv_Product = 0
14                   for $j = -k/2$ to $k/2$ do
15                      for $i = -k/2$ to $-k/2$ do
16                         Image_pixel = input $[L, s.y + j, s.x + i, c_{i}]$
17                         Filter_pixel = $W[L, c_{i}, c_{o}, j, i]$
18                         Conv_Product = Conv_Product + Filter_pixel * Image_pixel
19                   END for
20                END for
21             END for
22         END for
23     END for
24     if split2 then
25        Feat_maps$[L, y, x, c_{o} + c_{out}] = Feat_{maps}[L, y, x, c_{o} + c_{out}] + Conv_{Product}$
26     else
27        Feat_{maps}[L, y, x, c_{o}] = Feat_{maps}[L, y, x, c_{o}] + Conv_{Product}
28     END if
29     END for
30     if split1 then
31        Feat_{maps} input$[L, 1,...] = Feat_{maps} input [L,...]$  
32        Feat_{maps} output$[L, 1,...] = Feat_{maps} output[L,...]$  
33     else
34        Feat_{maps} input$[L, 1,...] = Feat_{maps} output[L,...]$  
35     END if
36     END for
37     for $c_{o} = 0$ to $c_{out} - 1$ do
38        $\sum_{y,x} Feat_{maps} input[layers,y,x,c_{o}] * \frac{1}{h_{out}w_{out}}$
39        END for
40     Layers classes = Softmax(Feat_{maps} output [layers,...])
41 END for
Embedded Plant Disease Recognition using Deep PlantNet on FPGA-SoC

Load Weights \(W\)

Processing Elements

Image (pixels) Input

Memory Controller

Image Input Cache

Outputs (O) Cache

Memory

Outputs (O) Cache

Memory

 Outputs (O) Cache

Bias

ReLU

Pooling Cache

Memory

AXI Interconnect

Fig. 6: PlanNet Accelerator.

Table 4: PlantNet IP hardware cost

<table>
<thead>
<tr>
<th>Name</th>
<th>BRAM (K)</th>
<th>DSP48E</th>
<th>FF</th>
<th>LUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP</td>
<td>10</td>
<td>24</td>
<td>520</td>
<td>-</td>
</tr>
<tr>
<td>Expression</td>
<td>-</td>
<td>26</td>
<td>0</td>
<td>953</td>
</tr>
<tr>
<td>FIFO</td>
<td>5</td>
<td>50</td>
<td>165</td>
<td>-</td>
</tr>
<tr>
<td>Register</td>
<td>12</td>
<td>70</td>
<td>1826</td>
<td>-</td>
</tr>
<tr>
<td>Memory</td>
<td>50</td>
<td>-</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td>Multiplexer</td>
<td>4</td>
<td>30</td>
<td>-</td>
<td>570</td>
</tr>
<tr>
<td>Instance</td>
<td>2</td>
<td>5</td>
<td>1561</td>
<td>2381</td>
</tr>
<tr>
<td>Total</td>
<td>83</td>
<td>15</td>
<td>3871</td>
<td>3904</td>
</tr>
<tr>
<td>Available</td>
<td>280</td>
<td>220</td>
<td>106400</td>
<td>53200</td>
</tr>
<tr>
<td>Utilization (%)</td>
<td>30</td>
<td>6</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>

3.2 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 catalog, 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_axi 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, via the Jupyter Notebook Overlay interface that offer 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 objective is to convert the RTL components to Netlist and then to find the best compromise between run time optimization, area optimization, routing optimization, etc. Then the bitstream file is generated to contain the optimized architecture.

Fig. 7: HW/SW PlantNet Architecture.

4 Results and Discussion

Concerning the proposed prototype based on the accelerated PlantNet model, the table 5 gives the hardware cost on the PYNQ-Z1 card. It shows that the proposed architecture occupies about 7% of the LUTs, about 20% of the LUTRAMs, about 6% of the FFs, 32% of the BRAMs, 7% of the DSPs, and 3% of the BUFGs. However, the design power consumption is approximately 2.485w. In addition, we design our custom overlay to implement the hybrid architecture on the PYNQ-Z1 card by exploiting the generated bitstream file
and the stored weights and biases. This phase aims to test the application and evaluate the execution time of a processed image. Thus, the execution time is approximately 0.04 second to process an image which proves that acceleration is achieved. In addition, we evaluate the computational roof and the maximum bandwidth roof of this design according to the equation. The proposed design achieves approximately 33 GFLOPS of compute roof and 1.9 GByte/s of bandwidth roof. Where $N_{DSP}$ is the total number of DSP on the PYNQ Z1 board. The required number of DSP to fulfill the processing tasks is denoted by $N_{R_{DSP}}$ and $f$ is the system frequency which is about 120MHz. While $N_{HP}$ is the number of high performance ports.

Table 5: Hardware Cost for HW/SW Architecture

<table>
<thead>
<tr>
<th>Resource</th>
<th>Utilization</th>
<th>Available</th>
<th>Utilization(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT</td>
<td>3700</td>
<td>53200</td>
<td>6.9</td>
</tr>
<tr>
<td>LUTRAM</td>
<td>3407</td>
<td>17400</td>
<td>19.9</td>
</tr>
<tr>
<td>FF</td>
<td>5989</td>
<td>106400</td>
<td>5.6</td>
</tr>
<tr>
<td>BRAM</td>
<td>90.50</td>
<td>280</td>
<td>32.3</td>
</tr>
<tr>
<td>DSP</td>
<td>16</td>
<td>220</td>
<td>7.2</td>
</tr>
<tr>
<td>BUFG</td>
<td>1</td>
<td>32</td>
<td>3.13</td>
</tr>
</tbody>
</table>

\[
\text{Computation roof} = 2 \times \frac{N_{DSP}}{N_{R_{DSP}}} \times f = 33 \text{ GFLOPS} \quad (3)
\]

\[
\text{Bandwidth roof} = \frac{64}{8} \times N_{HP} \times f = 1.9 \text{ GByte/s} \quad (4)
\]

The proposed design has been also compared to other related works. Table 6 shows the performance comparison, in which the relevant references are given. The suggested work cited in [43] whose aim is to implement a CNN model, in which the depth-separable convolution and the standard convolution are accelerated. The Xilinx ZYNQ 7100 board was used to implement the design. However, the implementation results prove a higher hardware cost occupation compared to our designs which is implemented on the XC7Z020 board. In addition, our designs achieved a lower power consumption (2.48W) and a higher computational roof (33 GFLOPS) compared to the same work (3.99 W and 17.11 GFLOPS), in addition our design achieved a higher working frequency of 120MHz compared to 100MHz. This is due to the fact that our proposed designs are optimized with high-performance interconnection to speed up the data exchanges. To further explore the effectiveness of our co-designs, the proposed work was compared to the uniform design cited in [44]. The latter aims to speed up 2D and 3D CNNs on Xilinx VC709 platform. However, it is remarkable that this suggested architecture occupies almost all of the FPGA resources (99.8% of DSPs, 60% of LUTs, 50% of FFs, and 26.6% BRAMs) with a higher power consumption (15.8W) compared to ours. After this comparison with the state of the art approaches, our proposed design outperforms these approaches by achieving high performance in terms of
Table 6: Comparative Study

<table>
<thead>
<tr>
<th>Platform</th>
<th>[43]</th>
<th>[44]</th>
<th>Our Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZYNQ7100</td>
<td>51%</td>
<td>62.9%</td>
<td>6.9%</td>
</tr>
<tr>
<td>XC7VX690T</td>
<td>46%</td>
<td>50.2%</td>
<td>32.3%</td>
</tr>
<tr>
<td>XC7Z020</td>
<td>38%</td>
<td>26.6%</td>
<td>5.6%</td>
</tr>
<tr>
<td>LUTs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRAMs</td>
<td>95%</td>
<td>99.8%</td>
<td>7.2%</td>
</tr>
<tr>
<td>FFs</td>
<td>99.8%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSPs</td>
<td>95%</td>
<td>99.8%</td>
<td>7.2%</td>
</tr>
</tbody>
</table>

Power (W) 3.99 15.8 1.52
Frequency (MHz) 100 120 120
Computational roof (GFLOPS) 17.11 - 33
Bandwidth roof (GByte/s) 3.2 - 1.9

low power consumption, low occupation of hardware resources on the FPGA, higher compute and bandwidth. Therefore, our designs could achieve a best tradeoff between resource cost, power consumption, and compute and bandwidth. making it suitable for implementation on embedded devices with a minimum resource budget.

5 Conclusion

A hybrid architecture for the plant disease detection based on PlantNet model is proposed. Initially, the design was implemented on the GPU platform, in which the training and testing results prove its efficiency. After that, a HW/SW design is proposed to accelerate data streaming and processing on PYNQ Z1 platform. The implementation results show that the proposed design reached the best performances in terms of processing time, computational roof, and bandwidth roof. As a future work, the quantization technique and the new Vitis tool of xilinx will be used to exploit the new deep learning processor on the new embedded platforms.

Acknowledgements 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


41. https://www.kaggle.com/emmarex/plantdisease

