electronics

MDPI

Article

FPGA-Based Low-Power High-Performance CNN Accelerator Integrating DIST for Rice Leaf Disease Classification

Jingwen Zheng [1], Zefei Lv [1], Dayang Li [2], Chengbo Lu [2], Yang Zhang [1], Liangzun Fu [1], Xiwei Huang [1,*], Jiye Huang [1], Dongmei Chen [2] and Jingcheng Zhang [2]

[1] Innovation Center for Electronic Design Automation Technology, Hangzhou Dianzi University, Hangzhou 310018, China; jingwen@hdu.edu.cn (J.Z.); lvzefei@hdu.edu.cn (Z.L.); yang_zhang@hdu.edu.cn (Y.Z.); liangzun.fu@hdu.edu.cn (L.F.); hjynet@hdu.edu.cn (J.H.)

[2] School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China; 232060224@hdu.edu.cn (D.L.); 222320063@hdu.edu.cn (C.L.); chendongmei@hdu.edu.cn (D.C.); zhangjcrs@hdu.edu.cn (J.Z.)

* Correspondence: huangxiwei@hdu.edu.cn

Abstract: Agricultural pest and disease monitoring has recently become a crucial aspect of modern agriculture. Toward this end, this study investigates methodologies for implementing low-power, high-performance convolutional neural networks (CNNs) on agricultural edge detection devices. Recognizing the potential of field-programmable gate arrays (FPGAs) to enhance inference parallelism, we leveraged their computational capabilities and intensive storage to propose an embedded FPGA-based CNN accelerator design aimed at optimizing rice leaf disease image classification. Additionally, we trained the MobileNetV2 network using multimodal image data and employed knowledge distillation from a stronger teacher (DIST) as the hardware benchmark. The solution was deployed on the ZYNQ-AC7Z020 hardware platform using High-Level Synthesis (HLS) design tools. Through a combination of fine-grained pipelining, matrix blocking, and linear buffering optimizations, the proposed system achieved a power consumption of 3.21 W, an accuracy of 97.41%, and an inference speed of 43 ms per frame, making it a practical solution for edge-based rice leaf disease classification.

Keywords: FPGA; accelerator; high-level synthesis; DIST; rice leaf disease classification

check for updates

Academic Editors: Fanyi Meng, Chiang Liang Kok, Teck Kheng Lee and Howard Tang

Received: 2 April 2025
Revised: 20 April 2025
Accepted: 20 April 2025
Published: 22 April 2025

Citation: Zheng, J.; Lv, Z.; Li, D.; Lu, C.; Zhang, Y.; Fu, L.; Huang, X.; Huang, J.; Chen, D.; Zhang, J. FPGA-Based Low-Power High-Performance CNN Accelerator Integrating DIST for Rice Leaf Disease Classification. *Electronics* **2025**, *14*, 1704. https://doi.org/10.3390/electronics14091704

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Plant disease issues significantly impact global food security and daily life, necessitating timely and accurate detection to mitigate their effects. Although chemical pesticides can control plant diseases, the wide variety of diseases presents difficulties for human observation- and experience-based diagnoses, often leading to misdiagnosis and delayed treatment. These delays can result in economic losses and environmental harm due to the overuse or incorrect application of pesticides. Consequently, precise and efficient disease identification has become a critical component of precision agriculture, highlighting the urgent need for automated detection technologies to minimize pesticide misuse and enhance disease management [1].

Recently, advancements in convolutional neural networks (CNNs) have shown remarkable potential for automating disease diagnosis, offering a powerful solution for image-based classification in agricultural applications [2–4]. Through the integration of Residual Channel Attention Block (RCAB), Feedback Block (FB), and Elliptical Metric Learning (EML), the accuracy and adaptability of CNN models to environmental variations

are enhanced. However, accurately classifying plant diseases against complex and noisy agricultural backgrounds remains a challenge.

To address the degradation in recognition accuracy caused by complex agricultural backgrounds, two primary strategies have been proposed. The first involves transmitting sensor-acquired images to edge terminals or servers via wireless communication for real-time processing [5]. The second approach focuses on deploying lightweight convolutional neural networks (CNNs) directly on edge devices [6,7]. While the latter approach is more compatible with the real-time, low-power, and cost-sensitive requirements of precision agriculture, it introduces additional challenges in model optimization. One common strategy to overcome these limitations is reducing the bit-width of model weights and activations, which effectively decreases model size and enhances computational efficiency for hardware implementation [8].

Although transfer learning is frequently employed to adapt models to new tasks quickly, it often requires large annotated datasets and results in high model complexity, limiting its feasibility on resource-constrained edge platforms. In contrast, knowledge distillation (KD) provides a more efficient alternative by adopting a teacher–student paradigm, where the student network learns to approximate the performance of a more complex teacher model. This technique improves classification accuracy and generalization while preserving a compact model structure, making it particularly suitable for edge deployment scenarios [9].

For instance, Progressive Soft Filter Pruning (PSFP) reduces the floating-point operations (FLOPs) of ResNet-50 by over 40%, substantially accelerating inference speed. Likewise, Hsieh et al. proposed a quantization framework that compresses VGG16 by a factor of 30 on an NVIDIA RTX2080Ti through the use of 8-bit weights and activations, illustrating the effectiveness of low-precision representations in model compression [10]. Mohammed et al. proposed a multimodal VGGish + MobileViT framework for public detection tasks, where knowledge distillation was used to transfer representations from a fine-tuned ViT teacher to a lightweight MobileViT student [11]. This approach improved the classification accuracy and F1-score to 97.13% and 0.97, respectively. The distilled model achieved real-time inference at 5–10 FPS on a Jetson Nano, demonstrating its suitability for edge deployment.

In practical applications, the fast inference of CNNs involves multiple considerations, including power consumption and real-time performance [12]. In the context of agricultural deployment, however, strict constraints on cost, portability, and power availability make traditional GPUs—despite their strong computational capabilities—unsuitable for on-site usage due to their large size and high energy demands [13,14]. In contrast, field-programmable gate arrays (FPGAs) offer low power consumption, high integration, and reconfigurability, making them well suited for edge computing in precision agriculture [15].

Compared to conventional microcontrollers, which are inherently constrained in computational capability, field-programmable gate arrays (FPGAs) offer substantial advantages in terms of parallelism and hardware-level customization. While microcontrollers are well suited for basic control-oriented tasks, they are inadequate for executing compute-intensive convolutional neural network (CNN) inference while meeting real-time performance requirements. In contrast, FPGAs support the implementation of parallel data paths and tailored computing architectures, making them highly efficient for accelerating deep learning workloads in resource-constrained edge environments.

Recent studies have explored various strategies for model compression and hardware acceleration to enhance deployment efficiency. Liao et al. proposed BearingPGA-Net, a lightweight fault diagnosis model trained via Decoupled Knowledge Distillation (DKD) guided by a pre-trained large-scale teacher model [16]. They further implemented a Verilog-

based FPGA accelerator with layer-wise quantization, achieving more than 200× speedup over CPU execution with less than 0.4% degradation in accuracy, recall, and F1-score. Zhang et al. introduced the Dual Distillation Double Gains (DDDG) method, combining generative self-supervised pre-training with bidirectional knowledge distillation [17]. Their approach improved the F1-score by up to 5.25% and achieved a 4.09× speedup on an ARM–FPGA heterogeneous system compared to software-only implementations. Stewart et al. conducted a comparative evaluation of the NEMOKD evolutionary KD algorithm and quantization techniques [18]. Their findings showed that NEMOKD improved inference accuracy by 82% on Intel's Movidius Myriad X VPU, albeit at the cost of a 38% increase in latency. In comparison, quantization on the Xilinx 7z020 FPGA achieved a more favorable trade-off between latency and hardware cost.

Motivated by these advancements, this work adopts a lightweight CNN model deployed on the ZYNQ-AC7Z020 FPGA platform to harness its low-power, reconfigurable computing capabilities, offering a practical and scalable solution for real-time plant disease detection in agricultural field environments.

To further enhance overall system performance, this work introduces a set of optimization strategies at both the model deployment and hardware design levels. Specifically, a stronger-teacher-based knowledge distillation (DIST) method is adopted to compress lightweight CNN models while preserving accuracy. Additionally, line buffering and matrix partitioning techniques are applied to improve memory access efficiency and computational throughput, and layer fusion is utilized to increase the parallelism of convolution operations. Leveraging the reconfigurable nature of FPGAs, these strategies effectively reduce computational overhead under stringent resource constraints, meeting the low-power and high-performance demands of on-site agricultural applications [19].

In terms of hardware implementation, this work employs High-Level Synthesis (HLS) tools in place of traditional Hardware Description Language (HDL) development. Compared to HDL, HLS significantly shortens development cycles and improves design flexibility and portability—particularly advantageous in scenarios that require rapid iteration or frequent architectural modifications. Moreover, HLS enables efficient hardware–software co-optimization, allowing algorithm-level performance tuning, which enhances both development efficiency and system robustness [20,21]. Based on HLS, a MobileNetV2 hardware accelerator is constructed with optimized task partitioning and resource scheduling, leading to notable improvements in execution efficiency. Furthermore, to address the bandwidth bottleneck between the Programmable Logic (PL) and Processing System (PS), a linear caching module is integrated to improve intermediate data locality and enhance overall data reuse. This reduces memory latency and further improves system throughput. In summary, this work proposes a comprehensive FPGA-based deployment framework that integrates model compression, efficient hardware acceleration, and system-level optimization. The proposed solution demonstrates practical feasibility and scalability for intelligent plant disease detection in resource-constrained agricultural environments.

2. Materials and Methods

2.1. DIST in Rice Disease Identification

In conventional knowledge distillation frameworks, knowledge transfer between the teacher and student models is typically achieved by directly matching probability distributions. That is, the student model is trained to approximate the output distribution of the teacher model as closely as possible. However, this direct matching approach presents inherent limitations. Specifically, when the teacher model is significantly more complex and powerful than the student model, the performance gap between them creates a "knowledge

gap", making it difficult for the student model to fully comprehend the teacher's output. This issue is particularly prevalent in deep learning applications.

To address these challenges, this study adopts the core principles of DIST, which emphasize capturing relative relationship information within the teacher's output. By designing a distillation strategy tailored to MobileNetV2, we aim to overcome the limitations of traditional distillation methods and achieve more effective knowledge transfer.

The key idea behind DIST is to leverage both inter-class relationships and intra-class relationships, thereby overcoming the constraints of conventional knowledge distillation, which focuses solely on absolute confidence matching. In this study, KL divergence is used to perform a full probabilistic distribution match between the teacher and student models, quantifying the asymmetry between their probability distributions. Typically, the student model is required to strictly align with the teacher's probability distribution, achieving optimal KL divergence only when the two distributions are identical. However, conventional distillation methods often overlook inter-class relationships and intra-class distribution characteristics, making KL divergence highly sensitive to distribution shifts, as shown in (1). Here, KL represents the KL divergence loss function, Yt and Ys are the teacher and student prediction vectors, T is the temperature parameter, and x is the class index. For instance, when the teacher model's output distribution fluctuates due to changes in training strategies or architectural modifications, the student model may struggle to adapt, ultimately degrading distillation performance.

To mitigate this issue, this study introduces a Pearson correlation coefficient-based matching function in DIST [22]. This approach emphasizes the relative relationships between class prediction scores in classification decisions, effectively enhancing the robustness of student models against variations in the teacher model. Additionally, leveraging the scale-invariance and translation-invariance properties of the Pearson correlation coefficient significantly improves the robustness of ResNet50-based knowledge distillation.

Considering that ResNet50′s deep structure and residual connections enhance feature extraction capabilities while maintaining an inference structure similar to MobileNetV2, it was selected as the teacher network in this study. To better utilize the output information of ResNet50 and reinforce the relative relationships between different classes, this research employs the Pearson correlation coefficient to measure feature distributions, optimizing the knowledge distillation process and improving student network performance. As expressed in (2), given two random variables u and v, where C is the number of classes, $Cov(u,v)$ represents the covariance of u and v, and Std(u) and Std(v) denote their respective standard deviations. Here, ui and vi represent the values of the i-th class, where $i = 1, 2, \ldots, C$. This transformation ensures translation and scale invariance.

Furthermore, by replacing the conventional KL divergence loss function in (3), where α is the weighting coefficient and $H(ytrue, Ps)$ represents the cross-entropy loss measuring the difference between the student network's predicted probability Ps and the ground-truth labels $ytrue$, the Pearson correlation-based feature distribution measurement allows a certain degree of variation in both scale and phase between teacher and student model outputs. Consequently, this modification relaxes the distillation loss, making it more adaptable and robust.

$$L_{KD} := \frac{\tau^2}{B}\sum_{i=1}^{B} KL\left(Y_{i,:}^{(t)}, Y_{i,:}^{(s)}\right) = \frac{\tau^2}{B}\sum_{i=0}^{B}\sum_{i=0}^{C} Y_{i,j}^{(t)}\log\left(\frac{Y_{i,j}^{(t)}}{Y_{i,j}^{(s)}}\right), \tag{1}$$

$$\rho_p(u,v) := \frac{Cov(u,v)}{Std(u)Std(v)} = \frac{\sum_{i=1}^{C}(u_i-\overline{u})(v_i-\overline{v})}{\sqrt{\sum_{i=1}^{C}(u_i-\overline{u})^2\sum_{i=1}^{C}(v_i-\overline{v})^2}}, \tag{2}$$

$$Loss = \alpha * H\left(y_{true}, P^S\right) + (1-\alpha) * KL\left(Q^T, Q^S\right), \tag{3}$$

In classification tasks, rather than directly matching the confidence scores for each class, this study employs the DIST framework, which better captures relative relationships such as "class A has a higher confidence than class B, which in turn has a higher confidence than class C". Under this paradigm, an inter-class relationship loss is formulated to measure the relative ranking among classes, as defined in (4)~(6). Specifically, Linter, L_{intra}, and L_{total} represent the inter-class relationship loss, intra-class relationship loss, and total loss, respectively. B denotes the batch size, while L_{cls} corresponds to the original classification loss between the student's predictions and ground-truth labels. The hyperparameters α, β, and γ are used to balance the loss components, where $j = 1, 2, \ldots, B$.

By computing inter-class relationship loss via (4), this study quantifies the extent to which ResNet50 and MobileNetV2 align in capturing inter-class relationships, such as determining whether "class A exhibits the highest confidence". Through a well-designed loss function, knowledge distillation significantly reduces the performance gap between student and teacher models while maintaining efficient inference capabilities. This approach enables the student network to acquire enhanced feature extraction abilities throughout the distillation process.

Beyond inter-class relationships, the intra-class relationships between different samples of the same class in ResNet50 and MobileNetV2 are also valuable for distillation learning. Therefore, this study extends the relationship matching framework to the intra-class level, with the derivation presented in (5). In particular, the distribution of predictions among different samples within the same class contains rich knowledge. For instance, in the "healthy" category of rice disease classification, while all samples belong to the same class, their predicted confidence scores may vary. These fine-grained variations reflect the teacher model's deeper understanding of data features. By computing the intra-class relationship loss, MobileNetV2 learns the distribution relationships among samples within the same class, further enhancing its feature representation capabilities for rice disease classification. The overall loss function is then formulated in (6).

Through this multi-dimensional knowledge extraction and transfer mechanism, MobileNetV2 effectively captures a more comprehensive knowledge structure from ResNet50, thereby improving its classification performance.

$$L_{inter} := \frac{1}{B}\sum_{i=1}^{B} d_p\left(Y_{i,:}^{(s)}, Y_{i,:}^{(t)}\right), \tag{4}$$

$$L_{intra} := \frac{1}{C}\sum_{j=1}^{C} d_p\left(Y_{:,j}^{(s)}, Y_{:,j}^{(t)}\right), \tag{5}$$

$$L_{tr} = \alpha L_{cls} + \beta L_{inter} + \gamma L_{intra}, \tag{6}$$

Figure 1 illustrates the process of inter-class and intra-class relationship transfer in the network. Here, s1, s2, s3, and s4 represent the student model's predicted outputs for the given samples, while t1, t2, t3, and t4 correspond to the teacher model's predictions. Compared to traditional knowledge distillation approaches, this method not only emphasizes cross-class relationships but also incorporates intra-class distribution information.

Suppose we have four samples belonging to three categories—(1) Leaf Blast, (2) Bacterial Blight, and (3) Healthy—as shown in Figure 1. The teacher model assigns probability distributions for each sample across these three categories, as presented in the table within the figure. The key aspect of inter-class relationship learning is to maintain the relative ranking among categories. Specifically, the student model is trained to capture the relative ordering of probabilities across different classes. For instance, if a sample belongs to the Leaf Blast category, its predicted probability for Leaf Blast should be higher than that

for Bacterial Blight and Healthy. Furthermore, the probability for Bacterial Blight should exceed that for Healthy, ensuring correct decision weighting where Leaf Blast receives the highest confidence. This establishes a relationship transfer mechanism that reinforces the ranking of class probabilities.

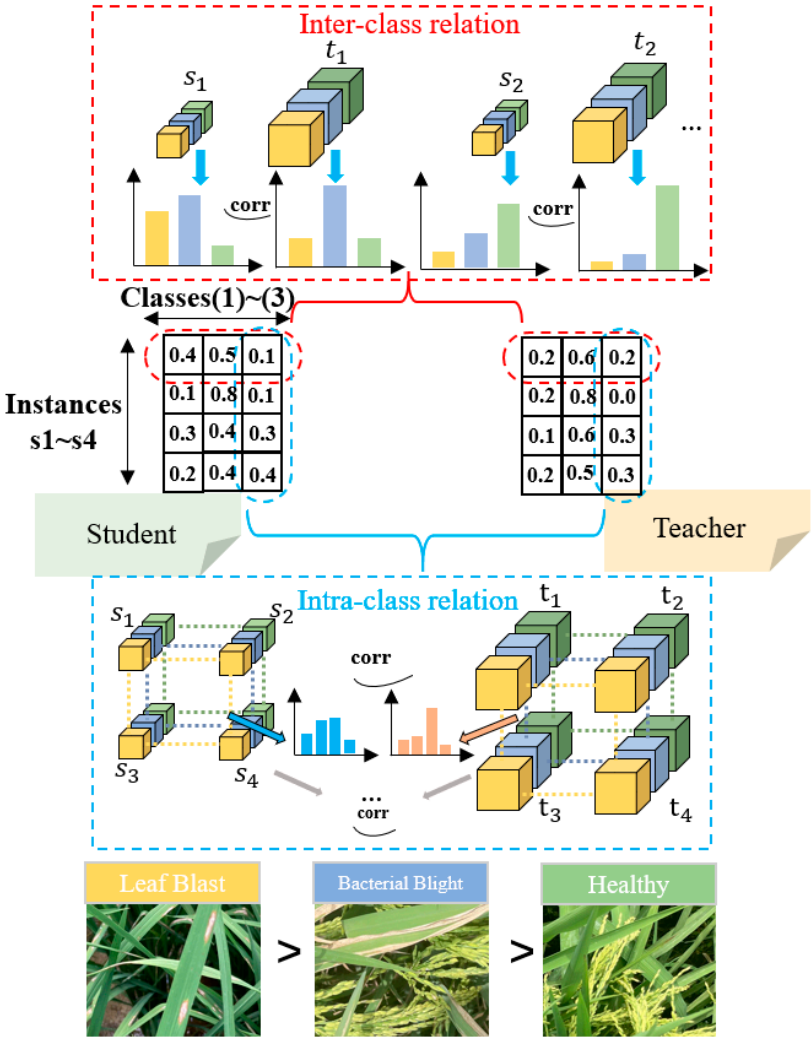


Figure 1. Schematic of knowledge distillation loss (KD Loss) for DIST.

Conversely, the same principle applies to other categories, ensuring that the student network effectively learns class-wise relationships as determined by the teacher. Through this approach, the teacher model provides the optimal predictions for each sample, and the student model refines its classification decisions iteratively through multiple relationship transfer steps. Ultimately, this method enhances the student model's ability to capture inter-class dependencies and achieve more accurate disease classification.

Through DIST knowledge distillation optimization, the teacher model conveys not only the predicted scores but also the relative relationships among these predictions to the student model, providing it with a more reliable decision-making framework. By capturing these nuanced inter-class and intra-class relationships, DIST knowledge distillation optimization enhances the robustness and generalizability of the student model, enabling it to learn more effectively from the teacher's guidance.

2.2. MobileNetV2 Acceleration Architecture Based on FPGA

After completing the network distillation of MobileNetV2, a hardware accelerator for rice disease classification was designed and implemented. As illustrated in Figure 2, the proposed architecture primarily consists of a ZYNQ-AC7Z020 FPGA development board along with peripheral components such as an image acquisition module. The ZYNQ platform integrates two components: the Programmable Logic (PL) and the Processing System (PS). The PS features an ARM Cortex™-A9 processor (Xilinx Inc., San Jose, CA, USA), which communicates with the PL through a high-speed AXI bus. This processor is responsible for managing parameter storage and retrieval, invoking IP cores, controlling computation engines, managing DDR3 memory, and transmitting results. The SD card is used to pre-store CNN parameters, while the DDR3 memory reads and stores network parameters from the SD card. During operations managed by the DDR3 control module, these parameters are transferred to the internal BRAM (Block RAM) on the PL side.

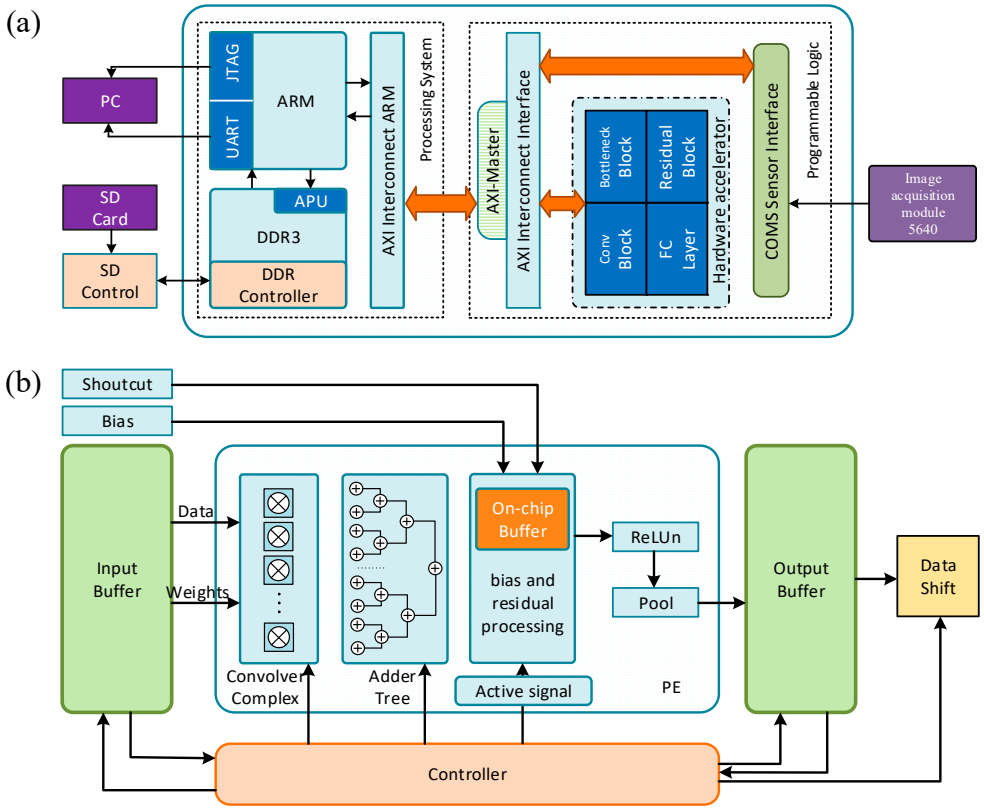


Figure 2. ZYNQ-AC7Z020-based hardware architecture. (**a**) Block diagram for the implementation of a rice disease classification system. (**b**) Processing element in rice disease image classification system based on FPGA.

The PL section includes a CNN hardware inference acceleration IP module, primarily composed of HLS-based acceleration engines. The IP cores consist of various convolution modules designed with multi-scale optimization. These modules employ advanced reuse and parallel algorithms to accelerate computations, such as convolutions, using weights and feature maps transmitted from the PS. Intermediate and final results are stored in BRAM, and enable signals are sent back to the PS upon task completion. Once the PS receives the enable signal and the final inference results, it retrieves the data via the AXI-Lite bus and transmits it to edge devices through a serial communication interface for visualization.

The PE module block diagram, based on an HLS design, is illustrated in Figure 2. It comprises several key components, including a Convolution Computation Module,

Adder Tree, Input/Output Buffers, Bias and Residual Processing Module, Data Shifter, and Controller. These components collaboratively handle data computation and transmission at the PE end. Convolution Computation Module: Responsible for performing convolution operations. Input data are buffered through the primary layout, where the buffer releases window selection functions for the input image. The selected window is processed by multipliers and the Adder Tree to compute convolution results. Adder Tree: aggregates all convolution results by summation. Bias and Residual Processing Module: adds inverse residual (Shortcut) data and bias data before the accumulator outputs become valid. Input/Output Buffers: store network parameters and intermediate computation results to ensure smooth data flow. Data Shifter: adjusts the data width through shift operations, restoring it to its original format as needed. Controller: manages parameter storage and retrieval, invokes IP cores, and oversees the operation of the computation engine. Together, these components enable efficient data computation and processing within the PE architecture, ensuring high performance and seamless operation.

2.3. Optimization of MobileNetV2 Circuit Based on HLS

In the acceleration of MobileNetV2 inference, optimizing the computational and memory architecture on the PL side is crucial for improving overall system performance. From a computational perspective, the design aims to enable efficient parallel processing and data reuse, fully leveraging the capabilities of the FPGA's DSP blocks and logic resources. On the memory side, the focus is on reducing external memory access latency and enhancing bandwidth utilization to meet the high-throughput data requirements of complex operations such as depthwise separable convolutions.

2.3.1. Irregular Matrix Partitioning

To address the limitation of FPGA resources in supporting the MobileNetV2 model [23], this work adopts a single-engine computation mode with a hierarchical array-partitioning approach, as illustrated in Figure 3. The computation and processing of sub-block convolutions are carried out by the accelerator in the PL (Programmable Logic) region. These sub-blocks are sequentially transferred to the on-chip cache, where the PL accelerator continues performing convolution computations for each sub-block. This iterative process is repeated until all convolution layers are fully processed. The size of the input sub-blocks is determined based on the width (col_o) and height (row_o) of the input feature maps, as well as the number of input channels (N_i) and output channels (N_o) in the convolution kernels. The dimensions of the output feature map sub-blocks are represented by T_{ro} and T_{co}, while the number of sub-blocks in the width and height directions are denoted as B_{rn} and B_{cn}, respectively. N_{bi} and N_{bo} represent the number of sub-blocks in the input and output channels of the convolution kernels. The total number of sub-blocks is given by B_{num}. This approach segments 2D or multi-dimensional data and stores them in multiple small RAMs, enabling efficient batch data access [24] and supporting multi-data reuse formats.

Traditional matrix partitioning techniques often address bandwidth limitations but overlook data reuse optimization. In contrast, this work implements a three-channel parallel convolution strategy by adjusting the dimensions of the partitioned sub-blocks, which enhances data parallelism and provides a foundation for optimizing on-chip caching. As shown in Figure 3, considering that the DW layers in the bottleneck structure use 3×3 convolutions, we adopt convolution kernels of the same size for three channels during inference. The width of the sub-blocks is fixed at 5, while the length is calculated using (7)~(15). This configuration enables the efficient inference of $3 \times T_{co}$ convolution results, which are then stored in BRAM in a one-dimensional format. This approach significantly

improves the read/write efficiency of the accelerator by reducing the likelihood of data access conflicts, thereby increasing overall system throughput and read/write performance.

$$row_o^l = \left(row_i^l - K_i^l + 2 \times p\right)/s + 1, \tag{7}$$

$$col_o^l = \left(col_i^l - K_j^l + 2 \times p\right)/s + 1, \tag{8}$$

$$T_{ro} = \left(T_{ri} - K_i^l + 2 \times p\right)/s + 1, \tag{9}$$

$$T_{co} = \left(T_{ri} - K_j^l + 2 \times p\right)/s + 1, \tag{10}$$

$$B_{rn} = row_i^l / T_{ro}, \tag{11}$$

$$B_{cn} = col_i^l / T_{co}, \tag{12}$$

$$N_{bi} = N_i^l / T_n, \tag{13}$$

$$N_{bo} = N_i^l / T_m, \tag{14}$$

$$B_num = B_{rn} \times B_{cn} \times N_{bi} \times N_{bo} \tag{15}$$

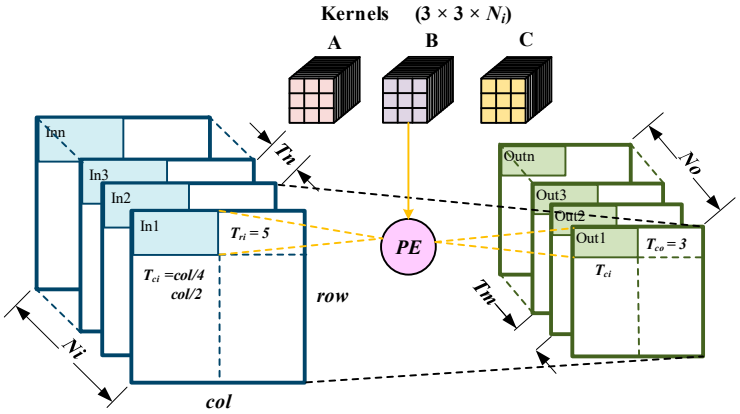


Figure 3. Matrix partition in the processing element.

2.3.2. Reusable Cache Structure

To minimize memory access and reduce latency, a linear buffer is designed for storing convolution kernel data (Figure 4) [25]. For 3 × 3 convolutions, this strategy leverages three-channel convolution kernels, where overlapping regions Line_A are reused. Feature values in Line_A are copied and concatenated with Line_B for efficient storage in a 1D buffer, sized 2–3 times Tr.

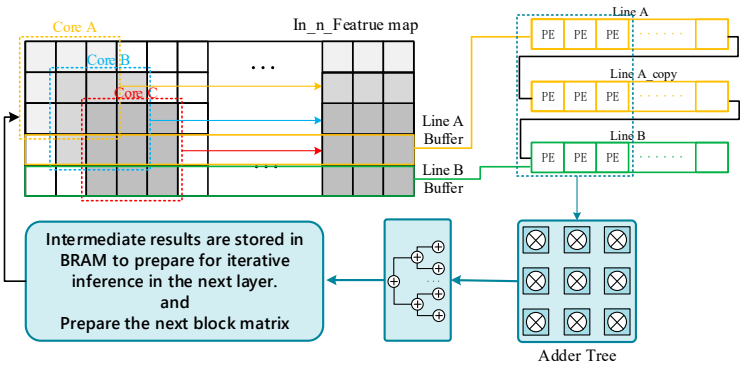


Figure 4. Reusable Cache Structure in the processing element.

Sliding windows, represented by 3 × 3 register arrays, are generated using shift registers. In a fully pipelined structure, convolution windows are created in each clock cycle for subsequent computations, enhancing throughput and reducing redundant data accesses.

2.3.3. Ping-Pong Operations

In most CNN architectures, ping-pong buffering is utilized to store data in two separate buffers: an input buffer and an output buffer [26]. During inference, the input buffer is divided into two regions to store either feature maps or intermediate results, along with weights and biases, referred to as the *Dbuf* and *Wbuf* blocks, respectively (as shown in Figure 5). In bottleneck structures, the weight data required for each stage are significantly smaller than the image data, particularly in PW layers. In contrast, architectures such as PW and FC layers exhibit the opposite characteristic, where the weight data volume far exceeds that of the input vector. Therefore, this work stores the weights for these layers in the larger-capacity *Dbuf* block, while input vectors are stored in the *Wbuf* block.

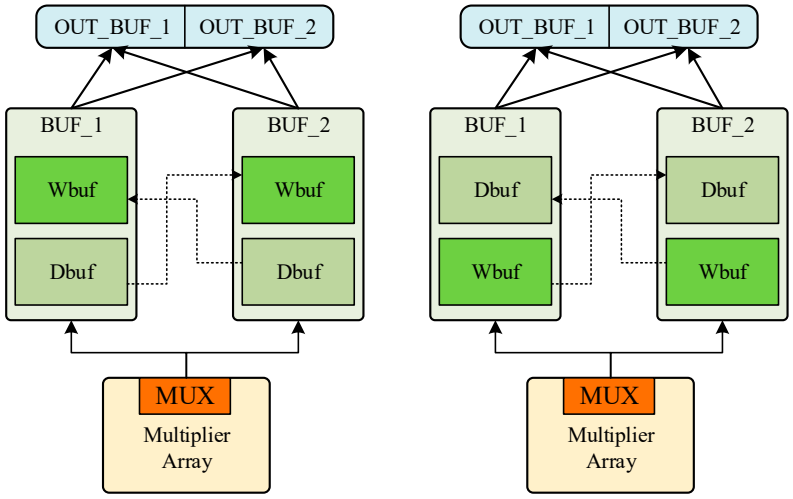


Figure 5. Ping-pong operations in different layer structures.

2.3.4. External Memory Access Optimization

The computation speed of PW layers in bottleneck blocks is primarily constrained by memory bandwidth [27]. Under such conditions, using dedicated hardware to accelerate the FC layer is inefficient. To address this, the PE computation units are optimized by leveraging convolution complexes to calculate PW layers. This approach maximizes the utilization of the external memory bandwidth of the current PL architecture.

In our system, each PE is allocated a 1024-length buffer to support 49 computation complexes, consistent with the matrix partitioning strategy. During the computation of CONV and DW layers, the buffer is filled sequentially. To minimize additional data routing logic for buffer filling and to maintain long burst lengths when accessing data for PW layer computations, the weight matrix is carefully arranged in external memory. The matrix is divided into 49 × 8-column blocks and 128 rows, with each block processed in a single stage. Within each block, the data are organized as shown in Figure 6.

Without the data arrangement for the PW layer, 49 × 1024 DMA transactions would typically be required to load a single block, with a burst length of only 8. However, by arranging the data as shown in Figure 6, only a single DMA transaction is needed to load the entire block. This arrangement ensures a long burst length, significantly improving the utilization of external memory bandwidth.

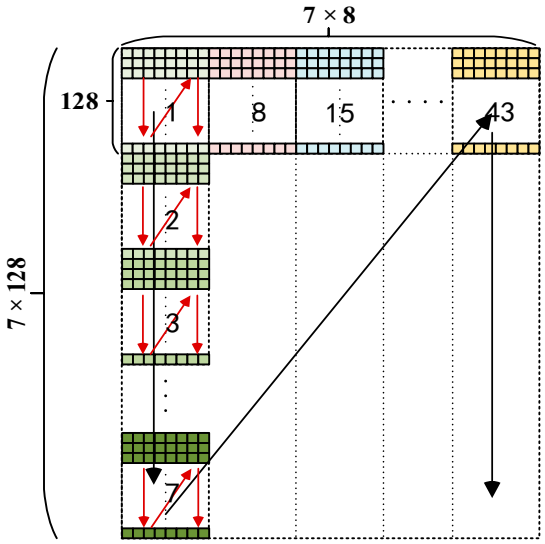


Figure 6. External memory access optimization.

2.3.5. Parallel Collaborative Computing Optimization

The combination of pipelining and tiling techniques is commonly employed in HLS to enable the parallel processing of neural network layers. In this approach, the computation layer is partitioned along the depth of the output channels (i.e., the number of output channels) using unrolling for parallelization (as shown in Figure 7). Each computation core is assigned to process a specific portion of the output channels. Every core accesses the input feature maps and processes its assigned subset of weight kernels, which correspond only to the output channels handled by that core. Each core independently calculates its assigned output channels and generates the respective segments of the output feature maps. The number of weight kernels allocated to each core is proportional to its unrolling or pipelining parallelism. This allocation ensures load balancing among the cores, allowing all cores to fully utilize their computational capacity.

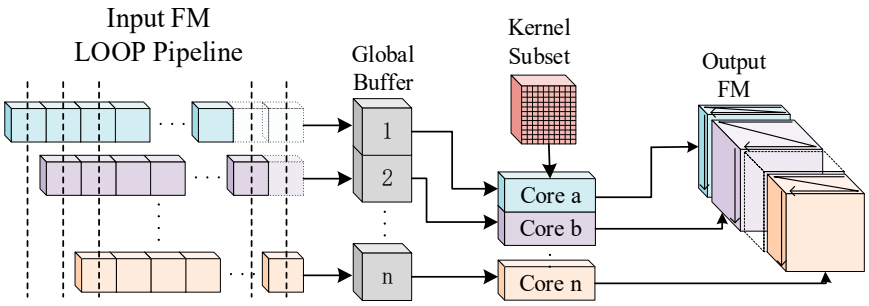


Figure 7. Parallel collaborative computing optimization.

To further optimize memory access, an on-chip global buffer is employed to store shared input feature maps. This global buffer serves as an intermediate layer between the off-chip memory and the local input buffers of each core. By avoiding redundant access to the off-chip memory, the global buffer significantly reduces memory access latency and improves system efficiency. This strategy effectively balances computational and memory resources, enhancing parallelism and throughput for CNN inference acceleration.

3. Results and Discussion

3.1. Multimodal Rice Disease Datasets

To optimize resource utilization in precision agriculture [28], it is essential to achieve more accurate classification of agricultural diseases. Here, we simulated real-world conditions by collecting images from both convenient mobile devices (smartphones) and unmanned aerial vehicle (UAV) sensor modules. The images are categorized based on the season (spring and autumn), disease severity (mild and severe), and UAV capture distances (1 m, 3 m, and 5 m above the rice canopy). Additionally, the images are taken from two angles, 45° and 90°. The details of the eight-category dataset are shown in Table 1.

Table 1. Dataset details.

Types of Diseases	Number of Images		
	Mobile Phone	UAV	All
white leaf blight	4256	4665	8921
rice stalk blight	2554	1064	3618
rice borer	3417	2608	6025
rice leaf roller	1287	2279	3566
neck blast	3629	-	3629
leaf plague	2136	1677	3813
glume blight	2738	1380	4118
healthy	3805	3381	7186

3.2. Plant Disease Identification Results of MobileNetV2

Testing was conducted on the mentioned dataset, resulting in the confusion matrix depicted in Figure 8, which illustrates the classification of each type of disease in rice by MobileNetV2. It is evident that the majority of the test results with large cardinalities are concentrated along the diagonal, indicating that the MobileNetV2 accelerator can reliably classify different diseases of varying severity with high reliability.

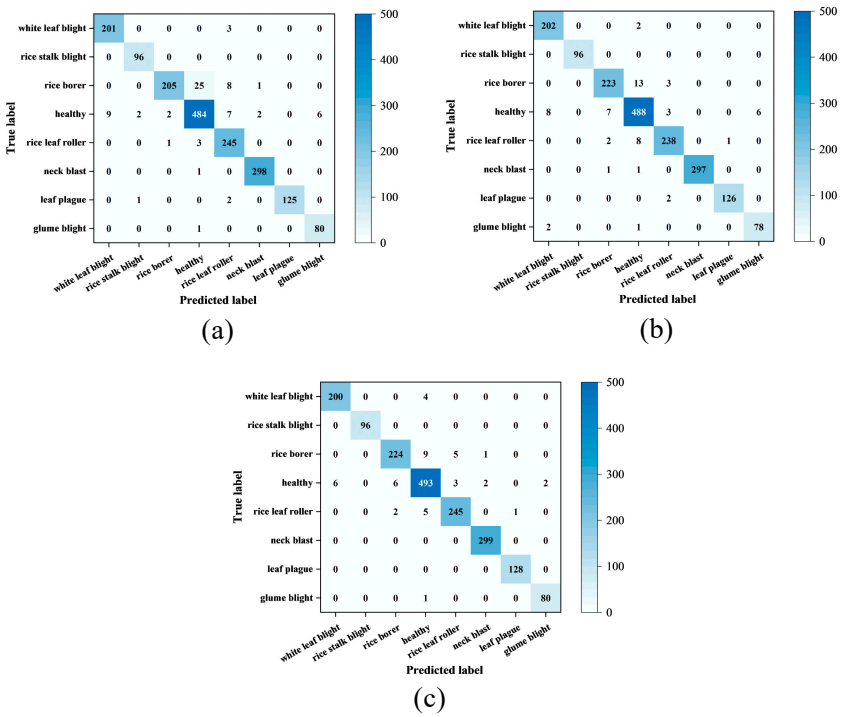


Figure 8. Classification of disease types in the rice test data: (**a**) MobileNetV2. (**b**) MobileNetV2-KD. (**c**) MobileNetV2-DIST.

The classification accuracy for neck blast disease is the highest, reaching 97.43%, attributed to its distinctive feature of light brown spots, which enables clear color differentiation compared to other diseases. The classification accuracy for sheath blight disease is relatively low, at 91.89%, mainly due to the presence of white or brown streaks. Additionally, the accuracy for rice blast disease and rice leaf folder disease is comparatively lower than the overall accuracy. The heatmap for rice disease classification, shown in Figure 9, demonstrates that the CNN can accurately extract features even for small disease characteristics. For instance, it effectively identifies the yellow-brown leaf edges or tips in white leaf blight, the bored and white panicles in rice borers, the white streaks in rice leaf rollers, and the dark green and spindle-shaped spots in leaf plague.

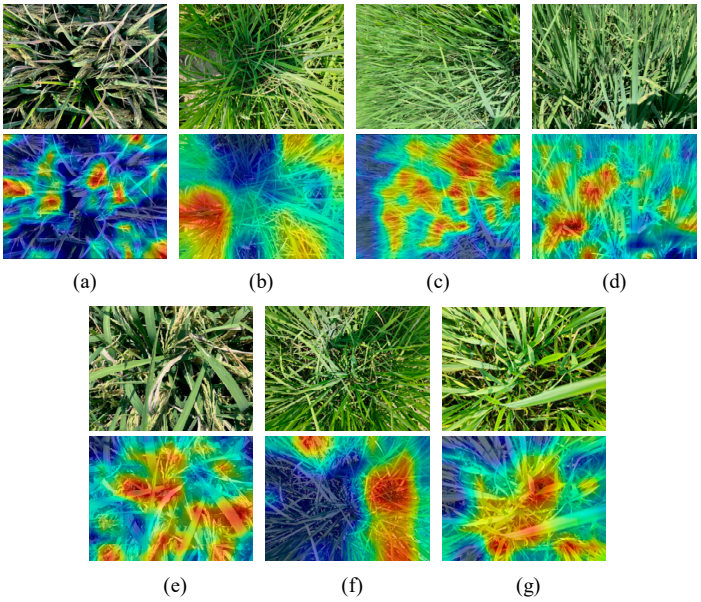


Figure 9. Heatmaps of four categories of rice diseases. (**a**) White leaf blight. (**b**) Rice borer. (**c**) Rice leaf roller. (**d**) Leaf plague. (**e**) Rice stalk blight. (**f**) Neck blast. (**g**) Glume blight.

3.3. Optimization of MobileNetV2 Acceleration Circuit

Table 2 compares the hardware resource utilization of the ZYNQ-AC7Z020 FPGA before and after HLS-based optimization. Following the accelerator circuit optimization, DSP utilization increased by 129.5%, indicating a significant improvement in computational parallelism. Additionally, the number of flip-flops (FFs) and lookup tables (LUTs) increased by 14.92% and 2.43%, respectively. Despite this increase in resource consumption, inference speed improved significantly, reducing from 0.138 s to 0.043 s, achieving a 320.93% acceleration. Furthermore, the utilization of BRAM and LUTRAM remained nearly unchanged, demonstrating that the optimization strategy effectively enhances performance while avoiding excessive memory overhead. However, due to the increased DSP computation, the overall power consumption increased from 2.704 W to 3.154 W, indicating that the improved computational efficiency comes at the cost of slightly higher power consumption.

Table 2. Comparison of hardware resource utilization.

	Hardware Unit Usage					Power (W)	Speed (s)
	DSP	BRAM	FF	LUTRAM	LUT		
without optimization	90	266	50,008	4698	43,626	2.704	0.138
after optimization	202	274	57,456	4526	46,262	3.154	0.043

Table 3 presents the power consumption breakdown across different hardware modules before and after optimization. The power consumption of DSP units increased from 0.212 W to 0.520 W, a 145.28% rise, which correlates with the substantial increase in DSP utilization. Additionally, clock power consumption increased from 0.140 W to 0.181 W, reflecting a higher clock frequency, which enhances parallelism. The power consumed by logic components also increased from 0.236 W to 0.301 W, highlighting the improved computational efficiency brought by the optimization strategy. Despite these increases, the total power consumption remained controlled at 3.154 W, aligning with the low-power design objectives of FPGA-based implementations. Compared to Xilinx's official DPU-P IP (2.932 W), the optimized architecture achieves higher computational efficiency while maintaining relatively low power consumption, demonstrating its cost-effectiveness and suitability for FPGA-based applications.

Table 3. Power consumption breakdown of different hardware modules.

Power Usage (W)	Clocks	Signals	Logic	BRAM	DSP	PS7	ALL
without optimization	0.140	0.481	0.236	0.333	0.212	1.302	2.704
using DPU-P (Xilinx IP)	0.139	0.474	0.252	0.330	0.437	1.300	2.932
after optimization	**0.181**	**0.506**	**0.301**	**0.344**	**0.520**	**1.302**	**3.154**

3.4. Effect of DIST

Here, we apply DIST to training for rice leaf disease detection, comparing it with samples that have not undergone KD (as shown in Table 4). The most notable improvement is observed in the accuracy of diagnosing rice borer, which increased by 7.56%. Overall accuracy improved from 95.91% to 97.41%, demonstrating that such methods can enhance the generalization of the classification system.

Table 4. Impact of DIST on accuracy.

Type of Knowledge Distillation	Accuracy (%)								
	White Leaf Blight	Rice Stalk Blight	Rice Borer	Healthy	Rice Leaf Roller	Neck Blast	Leaf Plague	Glume Blight	All
MBV2	98.5	100.0	85.8	94.5	98.4	99.7	97.7	98.8	95.9
MBV2-KD	99.0	100.0	93.3	95.3	95.6	99.3	98.4	96.3	96.7
MBV2-DIST	**98.0**	**100.0**	**93.4**	**96.3**	**96.8**	**100**	**100**	**98.8**	**97.4**

To evaluate the effectiveness of DIST in enhancing the performance of MobileNetV2, we conducted comparative experiments on eight-class classification tasks using MobileNetV2 with KD, with DIST, and without distillation optimization. As shown in Table 5, all distillation methods significantly improved the performance of MobileNetV2, while keeping the number of parameters constant.

Table 5. MobileNetV2 distillation performance metrics.

Type of Knowledge Distillation	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
MBV2	95.91	96.24	96.66	96.36
MBV2-KD	96.68	96.7	96.68	97.01
MBV2-DIST	**97.41**	**97.33**	**98.02**	**97.66**

Among these methods, DIST demonstrated the highest accuracy (97.41%), surpassing KD by 0.12%. Additionally, DIST achieved the highest recall (98.02%) and F1-score (97.66%), indicating its superior ability to both detect and classify various categories correctly. This suggests that DIST effectively captures inter-class and intra-class relationships, further boosting classification performance. These results confirm the capability of DIST to enhance the generalization and robustness of MobileNetV2, particularly in challenging classification scenarios like rice pest and disease datasets.

3.5. Evaluation of FPGA and Other Computing Devices

Table 6 presents a comprehensive benchmark of various MobileNet variants implemented across different edge computing platforms. In this table, speed denotes the inference time required to process a single image. To ensure fair comparison across platforms, we adopted optimized implementations for each hardware target. On Raspberry Pi 4B and Jetson TX2, MobileNetV2 was deployed using TensorFlow Lite v2.9.1 with post-training quantization to FP16. The Jetson platform further benefited from TensorRT acceleration. On Intel Core i5 and Xeon Bronze 3106 CPUs, we used ONNX Runtime with dynamic quantization enabled. In contrast, the FPGA platform employed a customized 16-bit fixed-point accelerator implemented via Vivado HLS. While GPUs and CPUs offer general-purpose acceleration, the FPGA implementation was tailored for low-latency and power-efficient inference using hardware-friendly scheduling and layer fusion. These variations reflect the platform-specific optimization tactics and explain the performance differences reported in Table 6.

Table 6. Evaluation of MobileNet implementation on different devices.

CNN	Equipment	Accuracy	Speed (s)	Power (w)	1/(Speed × Power)	Cost (USD)
MobileNetV2-large [29]	Raspberry Pi 4	95.67%	0.325	5	15.38	41.1
MobileNetV2 [30]	Cortex-A53	89.30%	0.096	-	-	171.1
MobileNetV1 [31]	Jetson Nano	97.84%	0.286	5.1	17.83	338.3
MobileNetV2 [32]	Raspberry Pi 4	97.70%	22.25	5	0.22	41.1
MobileNetV3-S [28]	Raspberry Pi 4	98.99%	0.251	5	19.92	41.1
MobileNetV2 [7]	Intel Core i5-6200U	95.24%	0.14	15	107.14	301.1
MobileNetV2 [7]	Intel Xeon Bronze3106	95.24%	0.18	14.54	80.78	260.2
MobileNetV2	Raspberry Pi 4B	95.91%	0.291	5.06	17.39	82.2
MobileNetV2	Jetson TX2	95.91%	0.029	5.1	175.86	794.5
MobileNetV2 (Use DPU-P)	ZYNQ-AC7Z020	95.91%	0.139	2.67	19.21	93.0
This work	**ZYNQ-AC7Z020**	**97.41%**	**0.043**	**3.15**	**73.26**	**93.0**

The proposed MobileNetV2 implementation on the Zynq-AC7Z020 FPGA achieves a classification accuracy of 97.41%, with an inference latency of only 43 ms and a power consumption of 3.15 W, at a total system cost of just USD 93. In contrast, the same MobileNetV2 model implemented using Xilinx's official DPU-P IP core on the same FPGA reaches only 95.91% accuracy with 139 ms latency and 2.67 W power consumption. This highlights that the proposed design improves accuracy by nearly 1.5% and reduces latency by approximately 70%, with only a modest 0.48 W increase in power.

Compared to other deployment scenarios, the FPGA-based accelerator demonstrates superior throughput and energy efficiency. On Raspberry Pi 4B, MobileNetV2 achieves 95.91% accuracy with a high inference latency of 291 ms and power consumption of 5.06 W. Jetson TX2, though faster at 29 ms, consumes 5.10 W—over 60% more power than our FPGA solution—and incurs a significantly higher system cost, making it less suitable for cost-sensitive, large-scale agricultural deployment. Similarly, CPU-based deployments on Intel Core i5-6200U and Xeon Bronze 3106 (Intel, Santa Clara, CA, USA) yield only 95.24% accuracy with latencies of 140 ms and 180 ms and power consumptions of 15 W and 14.54 W, respectively—substantially exceeding the power budget of edge applications.

Moreover, this work evaluates inference efficiency using the metric 1/(Speed × Power), which evaluates the comprehensive performance of speed and energy consumption. Our implementation achieves a value of 73.26, highlighting its advantage in delivering high-throughput, low-power inference—an essential requirement for practical, real-time plant disease classification in resource-constrained agricultural environments.

While the accuracy of our 16-bit quantized MobileNetV2 (97.41%) is slightly lower than that reported for MobileNetV1 (97.84% on Jetson Nano) and MobileNetV3-Small (98.99% on Raspberry Pi 4), this difference is primarily due to quantization noise introduced by INT16 precision. Nevertheless, under stringent resource constraints such as the 32-bit DDR3/AXI4 bandwidth on the Zynq platform and a similar cost envelope, our HLS-based FPGA implementation achieves faster inference, competitive accuracy, and low power consumption—demonstrating its practicality and efficiency for real-time, energy-constrained plant disease classification at the edge.

4. Conclusions and Future Work

This study employs MobileNetV2 as the CNN inference backbone and deploys it on the ZYNQ-AC7Z020 FPGA platform. By incorporating fine-grained pipelining, optimized memory access patterns, and a knowledge distillation strategy from a stronger teacher (DIST), the system effectively mitigates quantization-induced accuracy degradation. The proposed solution achieves a classification accuracy of 97.41% on rice disease images, with an inference latency of 43 ms per frame and a power consumption of only 3.21 W. Compared to GPU-based platforms, the FPGA implementation significantly reduces energy consumption and system cost while offering enhanced portability. Against devices like Raspberry Pi, it demonstrates superior classification speed and throughput, making it well suited for real-time, edge-level agricultural deployments.

Importantly, the design methodology emphasizes a resource-efficient balance between accuracy and speed, enabled through HLS-based accelerator development, which enhances portability and accelerates hardware–software co-design. Although this work focuses on MobileNetV2, we believe the knowledge distillation and optimization strategies presented here could be extended to other lightweight models, such as MobileNetV1 or MobileNetV3-Small. Investigating the generalizability of DIST across architectures and deployment platforms remains a promising direction for future work.

From an agricultural perspective, the proposed FPGA-based system offers high adaptability to environmental constraints and represents a practical and energy-efficient solution for precision crop disease identification. This work reinforces the potential of deploying AI models on edge hardware to improve agricultural disease monitoring, prevention, and management in real-world scenarios.

Author Contributions: Conceptualization, J.Z. (Jingwen Zheng), X.H., J.H., D.C. and J.Z. (Jingcheng Zhang); Data curation, J.Z. (Jingwen Zheng), Z.L., D.L., C.L., Y.Z. and L.F.; Formal analysis, J.Z. (Jingwen Zheng), Z.L., Y.Z. and L.F.; Funding acquisition, X.H., D.C. and J.Z. (Jingcheng Zhang); Investigation, J.Z. (Jingwen Zheng), Z.L., D.L., C.L., Y.Z. and L.F.; Methodology, J.Z. (Jingwen Zheng), Z.L., X.H. and J.H.; Project administration, X.H. and J.Z. (Jingcheng Zhang); Resources, X.H., J.H. and J.Z. (Jingcheng Zhang); Software, J.Z. (Jingwen Zheng), Z.L. and D.C.; Supervision, X.H. and J.Z. (Jingcheng Zhang); Validation, J.Z. (Jingwen Zheng) and Z.L.; Visualization, J.Z. (Jingwen Zheng), X.H. and D.C.; Writing—original draft, J.Z. (Jingwen Zheng); Writing—review and editing, J.Z. (Jingwen Zheng), X.H., J.H., D.C. and J.Z. (Jingcheng Zhang). All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key Research and Development Program of China (Grant No. 2022YFD2000100), National Natural Science Foundation of China (Grant No. 62271184, 62276086), Fundamental Research Funds for the Provincial Universities of Zhejiang (Grant No. GK249909299001), and Zhejiang Provincial Natural Science Foundation of China (Grant No. LZ22F010007, ZCLZ24F0201).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mohanty, S.P.; Hughes, D.P.; Salathé, M. Using Deep Learning for Image-Based Plant Disease Detection. *Front. Plant Sci.* **2016**, *7*, 1419. [CrossRef] [PubMed]
2. Chen, J.; Teo, T.H.; Kok, C.L.; Koh, Y.Y. A Novel Single-Word Speech Recognition on Embedded Systems Using a Convolution Neuron Network with Improved Out-of-Distribution Detection. *Electronics* **2024**, *13*, 530. [CrossRef]
3. Jiang, P.; Chen, Y.; Liu, B.; He, D.; Liang, C. Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks. *IEEE Access* **2019**, *7*, 59069–59080. [CrossRef]
4. Pathan, M.; Patel, N.; Yagnik, H.; Shah, M. Artificial Cognition for Applications in Smart Agriculture: A Comprehensive Review. *Artif. Intell. Agric.* **2020**, *4*, 81–95. [CrossRef]
5. Rupanagudi, S.R.; Ranjani, B.S.; Nagaraj, P.; Bhat, V.G.; Thippeswamy, G. A Novel Cloud Computing Based Smart Farming System for Early Detection of Borer Insects in Tomatoes. In Proceedings of the 2015 International Conference on Communication, Information & Computing Technology (ICCICT), Mumbai, India, 15–17 January 2015.
6. Wagle, S.A. A Deep Learning-Based Approach in Classification and Validation of Tomato Leaf Disease. *Trait. Signal* **2021**, *38*, 140. [CrossRef]
7. Luo, Y.; Cai, X.; Qi, J.; Guo, D.; Che, W. FPGA–Accelerated CNN for Real-Time Plant Disease Identification. *Comput. Electron. Agric.* **2023**, *207*, 107715. [CrossRef]
8. Chang, X.; Pan, H.; Lin, W.; Gao, H. A Mixed-Pruning Based Framework for Embedded Convolutional Neural Network Acceleration. *IEEE Trans. Circuits Syst. I* **2021**, *68*, 1706–1715. [CrossRef]
9. Dai, G.; Fan, J. An Industrial-Grade Solution for Crop Disease Image Detection Tasks. *Front. Plant Sci.* **2022**, *13*, 921057. [CrossRef]
10. Hsieh, T.-H.; Kiang, J.-F. Comparison of CNN Algorithms on Hyperspectral Image Classification in Agricultural Lands. *Sensors* **2020**, *20*, 1734. [CrossRef]
11. Mohammed; Swapnil, A.L.; Peris, M.D.; Nihal, I.H.; Khan, R.; Matin, M.A. Multimodal Deep Learning for Violence Detection: VGGish and MobileViT Integration with Knowledge Distillation on Jetson Nano. *IEEE Open J. Commun. Soc.* **2024**, *6*, 2907–2925. [CrossRef]
12. Kok, C.L.; Heng, J.B.; Koh, Y.Y.; Teo, T.H. Energy-, Cost-, and Resource-Efficient IoT Hazard Detection System with Adaptive Monitoring. *Sensors* **2025**, *25*, 1761. [CrossRef]
13. Sterpone, L.; Azimi, S.; De Sio, C. CNN-Oriented Placement Algorithm for High-Performance Accelerators on Rad-Hard FPGAs. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *43*, 1079–1092. [CrossRef]
14. Wang, Y.; Liao, Y.; Yang, J.; Wang, H.; Zhao, Y.; Zhang, C.; Xiao, B.; Xu, F.; Gao, Y.; Xu, M.; et al. An FPGA-Based Online Reconfigurable CNN Edge Computing Device for Object Detection. *Microelectron. J.* **2023**, *137*, 105805. [CrossRef]
15. Kim, Y.; Kim, H.; Yadav, N.; Li, S.; Choi, K.K. Low-Power RTL Code Generation for Advanced CNN Algorithms Toward Object Detection in Autonomous Vehicles. *Electronics* **2020**, *9*, 478. [CrossRef]
16. Liao, J.-X.; Wei, S.-L.; Xie, C.-L.; Zeng, T.; Sun, J.; Zhang, S.; Zhang, X.; Fan, F.-L. BearingPGA-Net: A Lightweight and Deployable Bearing Fault Diagnosis Network via Decoupled Knowledge Distillation and FPGA Acceleration. *IEEE Trans. Instrum. Meas.* **2023**, *73*, 1–14. [CrossRef]
17. Zhang, H.; Liu, W.; Guo, Q.; Shi, J.; Chang, S.; Wang, H.; He, J.; Huang, Q. DDDG: A Dual Bi-Directional Knowledge Distillation Method with Generative Self-Supervised Pre-Training and Its Hardware Implementation on SoC for ECG. *Expert Syst. Appl.* **2024**, *244*, 122969. [CrossRef]
18. Stewart, R.; Nowlan, A.; Bacchus, P.; Ducasse, Q.; Komendantskaya, E. Optimising Hardware Accelerated Neural Networks with Quantisation and a Knowledge Distillation Evolutionary Algorithm. *Electronics* **2021**, *10*, 396. [CrossRef]
19. Ahmed, H.O.; Ghoneima, M.; Dessouky, M. Systolic-Based Pyramidal Neuron Accelerator Blocks for Convolutional Neural Network. *Microelectron. J.* **2019**, *89*, 16–22. [CrossRef]
20. Saidi, A. FPGA-Based Implementation of Classification Techniques: A Survey. *Integration* **2021**, *81*, 280–299. [CrossRef]
21. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.

22. Benesty, J.; Chen, J.; Huang, Y.; Cohen, I. Pearson Correlation Coefficient. In *Noise Reduction in Speech Processing*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–4.
23. Wu, D.; Zhang, Y.; Jia, X.; Tian, L.; Li, T.; Sui, L.; Xie, D.; Shan, Y. A High-Performance CNN Processor Based on FPGA for MobileNets. In Proceedings of the 2019 29th International Conference on Field Programmable Logic and Applications (FPL), Barcelona, Spain, 8–12 September 2019; p. 136.
24. Li, H.; Fan, X.; Jiao, L.; Cao, W.; Zhou, X.; Wang, L. A High Performance FPGA-Based Accelerator for Large-Scale Convolutional Neural Networks. In Proceedings of the 2016 26th International Conference on Field Programmable Logic and Applications (FPL), Lausanne, Switzerland, 29 August–2 September 2016; pp. 1–9.
25. Ma, Y.; Cao, Y.; Vrudhula, S.; Seo, J.-S. Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural Networks. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017; pp. 45–54.
26. Zhang, C.; Li, P.; Sun, G.; Guan, Y.; Xiao, B.; Cong, J. Optimizing FPGA-Based Accelerator Design for Deep Convolutional Neural Networks. In Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2015; pp. 161–170.
27. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
28. Petrović, B.; Bumbálek, R.; Zoubek, T.; Kuneš, R.; Smutný, L.; Bartoš, P. Application of Precision Agriculture Technologies in Central Europe-Review. *J. Agric. Food Res.* **2024**, *15*, 101048. [CrossRef]
29. Tarek, H.; Aly, H.; Eisa, S.; Abul-Soud, M. Optimized Deep Learning Algorithms for Tomato Leaf Disease Detection with Hardware Deployment. *Electronics* **2022**, *11*, 140. [CrossRef]
30. Suharjito; Elwirehardja, G.N.; Prayoga, J.S. Oil Palm Fresh Fruit Bunch Ripeness Classification on Mobile Devices Using Deep Learning Approaches. *Comput. Electron. Agric.* **2021**, *188*, 106359. [CrossRef]
31. Zhao, G.; Quan, L.; Li, H.; Feng, H.; Li, S.; Zhang, S.; Liu, R. Real-Time Recognition System of Soybean Seed Full-Surface Defects Based on Deep Learning. *Comput. Electron. Agric.* **2021**, *187*, 106230. [CrossRef]
32. Razfar, N.; True, J.; Bassiouny, R.; Venkatesh, V.; Kashef, R. Weed Detection in Soybean Crops Using Custom Lightweight Deep Learning Models. *J. Agric. Food Res.* **2022**, *8*, 100308. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.