scientific reports

Check for updates

OPEN

A lightweight hybrid model for scalable and robust plant leaf disease classification

Muhammad Asghar[1], Zahid Farooq Khan[2], Muhammad Ramzan[3], Muhammad Attique Khan[4]✉, Jamel Baili[5], Yudong Zhang[6], Yunyoung Nam[7]✉ & Yun-Cheol Nam[8]

Plant leaf diseases significantly impact crop yield and quality, causing substantial economic loss and risking food security. Despite significant progress in the field of automated plant disease diagnosis, there are still several challenges that need to be addressed. Accurate classification of plant leaf diseases at an early stage is crucial for diagnosis and effective treatment of these plant diseases. As the agricultural industry faces growing challenges from plant diseases, quickly identifying these diseases in a field environment while considering the computational resource limitations is more important than ever. To overcome these challenges, this study proposed a lightweight and compact convolutional neural network model, HPDC-Net (Hybrid Plant Disease Classification Network). The network used a block architecture with three blocks termed as Depth-wise Separable Convolution Block (DSCB), Dual-Path Adaptive Pooling Block (DAPB), and Channel-Wise Attention Refinement Block (CARB). The model extracts a robust but limited number of features due to the use of depth-wise separable convolutions in DSCB, making it accurate but lightweight. The proposed model has been trained to classify potato and tomato leaf diseases on three datasets. The model achieves a high accuracy score > 99% on all three datasets while keeping GFLOPs limited to 0.06 and the number of parameters to 0.52 M (for 10 classes) and 0.17 M (for 03 classes), yielding 19.82 FPS on CPU and 408.25 FPS on GPU in our setup. The code for implementation of proposed model is available on GitHub: https://github.com/ZahidFarooqKhan/HPDC-Net.

Keywords Precision agriculture, Edge AI model, Compact neural network, Crop health monitoring, Leaf disease classification

The Food and Agriculture Organization (FAO) of the United Nations states that the global food demand is increasing with the world population growth rate, which is projected to reach approximately 9.7 billion by 2050, requiring a 70% increase in food production[1]. Agriculture is the backbone of food production. The crop yield is significantly affected by plant pathogens. These pathogens are responsible for up to 40% crop loss annually on a global scale, threatening food security and livelihoods worldwide[2]. Plant diseases not only reduce crop yield but also compromise its quality. Accurate identification and classification of pathogens in their early stages is crucial for mitigating the food security risk and reducing the use of pesticides, which subsequently poses an environmental risk. Traditionally, farmers visually inspect the crop to detect and perceive plant diseases, but this approach has several limitations, including a lack of scalability, non-uniformity, low accuracy, time loss, and labor intensity[3]. As a result of its low diagnostic accuracy, visual inspection frequently fails to identify crop diseases in their early stages, leading to missed opportunities for the best control period[4]. To overcome these limitations, recent technological applications of machine learning, artificial intelligence, and deep learning frameworks in the agriculture sector have contributed promising results to perceive and classify plant diseases.

[1]Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan. [2]Department of Computer Science and IT, University of Lahore, Sargodha, Pakistan. [3]Department of Computer Science, Gujrat Institute of Management Sciences, Gujrat, Pakistan. [4]Center of Artificial Intelligence, Prince Mohammad Bin Fahd University, AlKhobar, Saudi Arabia. [5]Department of Computer Engineering, College of Computer Science, King Khalid University, 61413 Abha, Saudi Arabia. [6]Department of Informatics, University of Leicester, Leicester, UK. [7]Department of ICT Convergence, Soonchunhyang University, Asan, Republic of Korea. [8]Department of Architecture, Joongbu University, Goyang 10279, Republic of Korea. ✉email: attique.khan@ieee.org; ynam@sch.ac.kr

These architectures can quickly analyze the complex patterns and subtle symptoms that may not be apparent to the human eye.

Deep Learning (DL) is a subfield of machine learning (ML). It employs neural networks for data analysis, feature extraction, and feature selection automatically, utilizing multiple hidden layers to extract low-level features, then combining them to perceive high-level features. Traditional ML-based image processing algorithms face limitations in artificially extracting these features. They adopt manually designed feature extraction methods and cannot learn these features automatically. While deep learning overcomes this limitation by automatically extracting low-level, intermediate-level, and high-level semantic features from a high-dimensional feature space to perform detection and classification tasks[5]. Recent studies have explored DL-based frameworks across various fields such as robots, enterprises, cybersecurity, virtual assistants, computer vision, healthcare, etc[6]. The smart agriculture industry has also been revolutionized by employing DL-based frameworks such as water management, soil analysis, plant disease detection, and crop yield detection[7]. Tomato fruit has huge importance to fulfill the need for global food because it is the second most cultivated crop after the potato[8]. However, tomato crop yield and quality are significantly affected by diseases caused by fungi, bacteria, viruses, and pests. The accurate and timely classification of tomato diseases is crucial to control and effectively manage the pathogens. Symptoms of the disease often appear on tomato plant leaves. The possibility of disease and its types can be identified by observing structural changes in tomato plant leaves. This study specifically addresses the challenges pertaining to the tomato leaf disease classification task. Figure 1 visually represents the plant leaf disease classification.

Convolutional neural networks (CNNs) are the most important architectures that are extensively being used in computer vision tasks for object detection and classification. It has numerous effective deployments, including AlexNet[9], GoogLeNet[10], ResNet[11], VGG[12] Etc.

In recent years, many studies have employed a CNN-based framework for the classification of different plant leaf diseases. Due to their robust feature extraction capabilities from high-dimensional feature spaces, these networks can be used for various classification tasks. However, limitations still exist specific to their efficiency and real scenario deployment in a field environment. The existing models are computationally intensive and require GPU-based processing for efficient working. But in a field environment, deployment of GPUs in edge devices is rarely possible in the current scenario. Therefore, the practical deployment of these models with high computational complexity is not possible at a wide scale, especially in developing countries. These limitations hinder the real-world deployment of these models on resource-constrained devices. To address these potential challenges, this study proposes a novel, lightweight, and compact framework feasible for low-power device deployment. The proposed model is trained on a GPU and evaluated on both GPU and CPU using challenging datasets[13]. The key contributions of this study are as follows:

1. We proposed a lightweight and compact neural network model, HPDC-Net, which is optimized for deployment on low-computational edge devices, enabling on-the-go plant leaf disease classification.
2. We proposed three lightweight blocks named as Depth-wise Separable Convolution Block (DSCB), Dual-Path Adaptive Pooling Block (DAPB), and Channel-Wise Attention Refinement Block (CARB) for optimal feature extraction, fusion, and enhancement.
3. We in-depth evaluated the model's performance on GPU and CPU. Our model achieved an excellent classification speed of 19.82 FPS on CPU with an accuracy of > 99%.

The rest of the sections of this paper are organized as follows: A literature review is given in Sect. "Related work", featuring the previous related work and identified research gap after assessing the limitations of existing methods. Section "Proposed methodology" explains the methodology of the proposed HPDC-Net model. Experimental

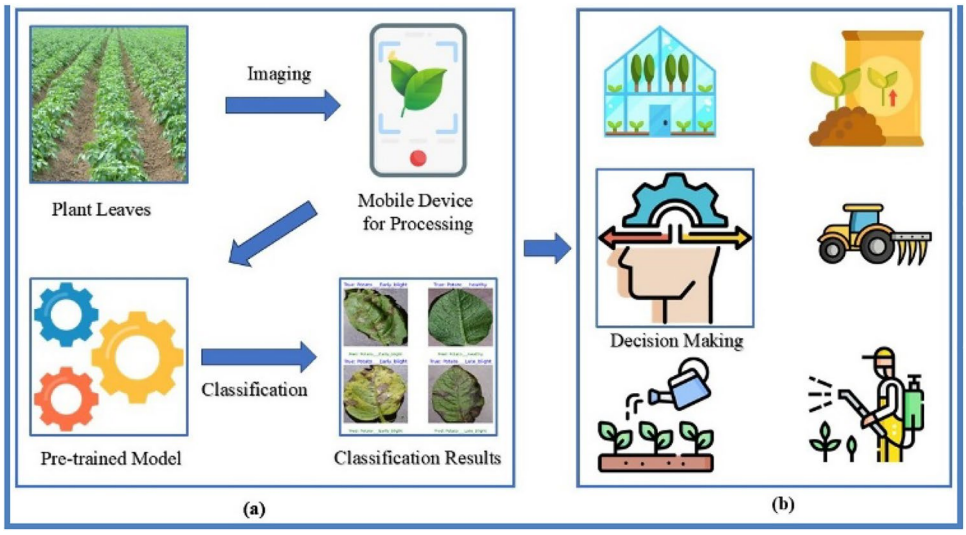

Fig. 1. Plant leaf disease classification framework process flow (**a**) Disease classification (**b**) Decision making for remedial actions like water and environment control, using pesticides, and fertilizers.

results and analysis are discussed in Sect. "Experimemtal results". Section "Discussion" includes a discussion of the findings, and Sect. "Conclusion" contains the conclusion.

Related work

In recent years, researchers have proposed diverse techniques and procedures for the automatic and efficient identification of plant leaf diseases. These studies have deployed various machine learning and deep learning-based frameworks to perform the plant disease classification task. In this section, we have conducted a comprehensive overview and critical analysis of such work from the literature, with a particular emphasis on tomato and potato leaf disease classification, and identified research gaps.

Pandiyaraju V et al. introduced an ensemble model with an exponential moving average function with temporal constraints and an enhanced weighted gradient optimizer that is integrated with two CNN models. The model comprised fine-tuned VGG-16 and NASNet mobile training methods to accurately perform in the early-stage classification task of tomato leaf diseases[14]. After applying the preprocessing techniques such as median filter, image cropping, brightness with contrast adjustments, and normalization, the model was trained and evaluated on the publicly available Plant Village dataset with an accuracy of 98.7%. However, its practical utility may be influenced by the potential limitation of high-level complexity. Such complexity may require significant computational resources and may not be feasible for implementation on resource-constrained devices. Showmick et al. proposed a lightweight custom CNN model and utilized transfer learning (TL)-based models VGG-16 and VGG-19 for the classification of tomato leaf diseases[15]. The model utilized a "tomato disease multiple sources" named dataset for training and evaluation of 10 tomato disease classes and achieved 95% accuracy. Finally, the model has been deployed in a Web-based and Android-based end-to-end (E2E) system to assist farmers in tomato plant disease classification. This approach shows better tomato plant disease classification. However, the classification accuracy needs more improvement.

Naeem et al. introduced a lightweight novel Deep Tomato Detection Network (DTomatoDNet) DL-based model for classification of tomato leaf diseases and achieved a classification accuracy of 98.95%[16]. This work is computationally efficient. However, it exhibits a limitation that it cannot detect multiple diseases on the same leaf image, and its reported precision, recall, and F1-score are around 95% on the test dataset. Similarly, Al-Gaashani et al. proposed a TL-based hybrid deep learning framework that extracts the features by employing pretrained models, MobileNetV2 and ResNet50V2[17]. Then, a Gravitational Search Algorithm (GSA) is applied for feature optimization. Finally, optimized features are passed to Multinomial Logistic Regression (MLR) for final classification. The proposed approach achieved high classification accuracy above 99% while simultaneously reducing the feature dimensionality by over 50%. However, its practical adoptability may be limited due to the use of dual CNN backbones and GSA-based feature selection, which hinders deployment on resource-constrained devices.

Saleh et al. introduced a customized deep learning-based framework CornerNet, and used DenseNet-77 as the backbone for accurate classification of tomato leaf diseases[18]. The feature vector computed by the DenseNet-77 is subsequently passed into the one-stage detector of the CornerNet model to both localize and classify the affected regions. This approach exhibited robust tomato plant disease classification results but shows detection degradation for images with angular variations. Al-Gaashani et al. introduced a novel strategy, utilizing a stack of four MultiRes blocks, each of which gradually increases the number of filters to control the growth of memory and computational requirements in deeper layers[19]. To collect more spatial information, it also implemented a residual link and a 1×1 convolutional layer. After each MultiRes block, it employed a Convolutional Block Attention Module (CBAM) to emphasize vital information and decrease redundant noise by inferring attention mappings along the channel and spatial dimensions. Finally, the EAMultiRes-DSPP model integrates a Dilated Spatial Pyramid Pooling (DSPP) module, which is designed to extract multi-scale information from the input image. The primary focus of this study was striking the balance between accuracy and efficiency; it achieved above 99% accuracy and also maintains a low parameter-count of 0.83 M. The on-the-go classification task requires further investigation to maintain high accuracy with low computational power. Marriam et al. presented a deep learning (DL)-based approach, namely ResNet-34-based Faster-RCNN, for tomato plant leaf disease classification[20]. In the first step, it generates the annotations of the suspected images. In the next step, the base network ResNet-34 feature extractor module extracts the deep features automatically; at last, the calculated feature vector is utilized for the Faster-RCNN model training to locate and categorize the tomato plant diseases. This approach achieved 99.97% accuracy in classifying tomato leaf disease images. However, the model exhibits a high level of complexity with 0.23 s of image inference time.

Lie et al. proposed a lightweight deep learning method based on the MobileNetv2-YOLOv3 model to early recognize tomato gray leaf spot disease. It used MobileNetv2 as the backbone network for feature extraction of MobileNetv2-YOLOv3[4]. This study demonstrates real-time accuracy, however, there is a need to generalize the model for the detection and classification of other kinds of tomato diseases. In research conducted by Abdulridha et al., a novel technique was developed that can be used in the laboratory and field using an unmanned aerial vehicle (UAV-based) for the identification and classification of three types of tomato diseases, bacterial spot (BS), target spot (TS), and tomato yellow leaf curl (TYLC), by utilizing machine learning and hyperspectral sensing[21]. The study performed well in recognizing the three categories of plant diseases, but suffered from extensive processing and the need to evaluate the model for other tomato diseases. Albattah et al. proposed a plant disease classification model by introducing a Custom CenterNet framework with DenseNet-77 as a base network. DenseNet-77 extracts the representative set of features from the input sample[22]. Then, the computed key points are used to train the CenterNet classifier to recognize and classify plant diseases of 14 species of the crop, including tomato crop. This model shows promising results but cannot be deployed in mobile devices due to its high level of computational complexity.

Md Shofiqul et al. utilized a combination of deep learning (DL) and machine learning (ML) methods for the detection and classification of tomato leaf diseases[23]. In this approach, at first, the input images are preprocessed using filtering and segmentation methods. Subsequently, a Conditional Generative Adversarial Network (CGAN) model is employed to generate synthetic images, aimed at addressing imbalanced and noisy or mislabeled data to enhance prediction accuracy. These synthetic images are then utilized in an attention-based dilated Convolutional Neural Network (CNN) layer to extract advanced features. Finally, a logistic regression model is applied to learn the extracted features and classify the tomato leaf disease images. A study shows robust results with 96.6% validation accuracy, leaving room for improvement in the model's accuracy. In Alzahrani et al., three different deep learning networks, i.e., DenseNet169, ResNet50V2, and a transformer model, namely ViT, were tested for tomato plant disease classification[24]. The DenseNet121 approach outperformed with a classification accuracy of 98.45%, followed by VIT and then ResNet50V2. A potential limitation of this study is the need for many parameters and a slow calculation. Such models are not suitable for deployment over devices with limited computational power. Hosny et al. proposed a novel lightweight integrated model that concatenates deep features obtained from a deep convolutional neural network (DCNN) with traditional handcrafted features (LBP) to compute the feature vector[25]. Subsequently, this computed feature vector is fed into the first fully connected layer, followed by the application of softmax for classification. This technique was trained and evaluated on three different plant leaf datasets: Apple leaf, Grape leaf, and Tomato leaf. It demonstrated promising results for plant leaf disease classification, achieving a testing accuracy of 96.5% for tomato leaf disease; however, there is a need for further improvement in the model's accuracy.

Sunil C.K. et al. introduced a deep learning-based framework for tomato leaf disease classification by combining the ResNet50, Multilevel Feature Fusion Network (MFFN), and an adaptive attention mechanism[26]. Adaptive attention mechanism with channel attention emphasizes each channel's information, which minimizes perception loss. Spatial attention is employed to focus on the location of the informative features by using the inter-spatial relationship of features. Pixel attention is employed to extract the informative features that enhance potential learning. The study reported impressive classification performance with 99.83% testing accuracy. However, the proposed technique requires high computational resources and is not feasible for resource-constrained devices. To maintain high classification accuracy with low computational overhead, Al-Gaashani et al. proposed a novel architecture, MSCPNet, which combines a truncated MobileNetV2 backbone with a Multi-Scale Convolutional PoolFormer block[27]. The truncated backbone retains only essential layers for feature extraction, and PoolFormer module performs feature aggregation with low computational utilization. MSCPNet demonstrated strong performance by achieving 97.44% accuracy on maize leaf disease classification and 99.32% accuracy on the PlantVillage dataset for tomato leaf disease classification while maintaining the low parameter-count of 0.998 M. However, the trade-off between efficiency and accuracy still requires further refinement for practical deployment over resource-constrained devices.

From the literature review, it is evident that earlier studies have proposed impressive frameworks for plant leaf disease classification and achieved promising results. However, several research challenges remain unaddressed. One of the primary issues is that many researchers have introduced deep learning-based CNN frameworks with numerous deep layers and parameters. Although these architectures often deliver high accuracy but they require substantial computational power, rendering them impractical for deployment on low-computation devices that can be utilized in the field. In response to the low computational demands, some researchers have proposed lightweight frameworks for deployment on resource-constrained devices. However, these models have been trained on only a limited number of tomato leaf disease datasets and need to be generalized for other diseases, and the evaluation of these models solely on CPU is not mentioned. Additionally, these works often fall short in terms of accuracy. To address these challenges, this study proposed a novel, lightweight deep learning architecture that can perform classification tasks efficiently on low computational devices like a smartphone. The proposed framework contains innovative blocks including optimized for feature selectivity while keeping their robustness intact. By striking a balance between efficiency and accuracy, this study aims to contribute a more versatile and practical tool for the agricultural community.

Proposed methodology

This section discusses the novel deep learning model architecture, which will be implemented to classify plant leaf images for various diseases. Figure 2 demonstrates the overall architecture and complete workflow of the proposed model.

HPDC-net

This study introduces a novel deep learning-based framework for identifying and classifying multiple plant diseases from leaf images. The proposed model, HPDC-Net, integrates several innovative blocks of components such as Depth-wise Separable Convolution Block (DSCB), Dual-Path Adaptive Pooling Block (DAPB), and Channel-Wise Attention Refinement Block (CARB) as shown in Fig 3. We used a combination of depth-wise separable convolution followed by point-wise convolutions to forward only the most robust features and minimize the computational overhead. Special residual connections are added to maintain the gradient flow. The model also employs parallel pooling strategies followed by channel reduction after concatenating features from different pooling layers. This parallelism improves the model's ability to capture multi-scale features, enabling it to effectively learn both fine-grained and large-scale patterns. The use of attention mechanisms ensures that the model can focus on the most important features, enhancing its robustness and accuracy. CARB refines the feature vector before passing it to a fully connected layer, which produces the final classification output. Complete architecture of the classification model is shown in Fig. 2. Each component of the architecture is discussed below in detail. Overall, the HPDC-Net combines simple techniques with efficient convolutional architectures,

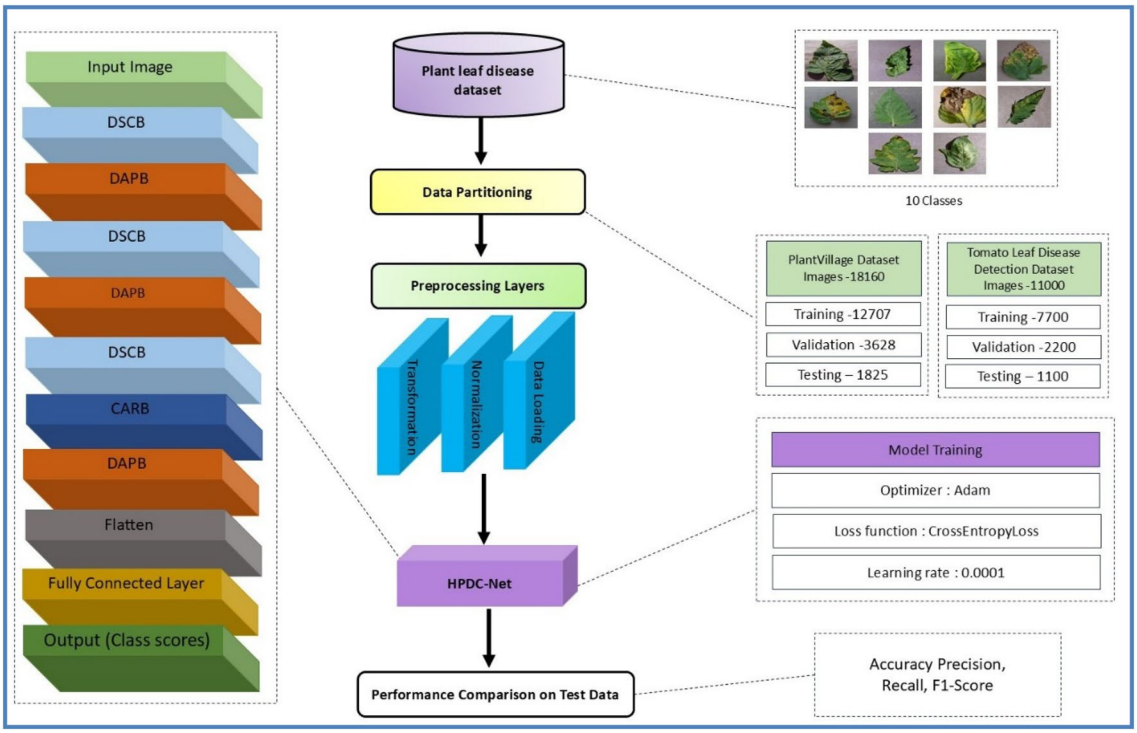

Fig. 2. Workflow and layer by layer representation of HPDC-Net.

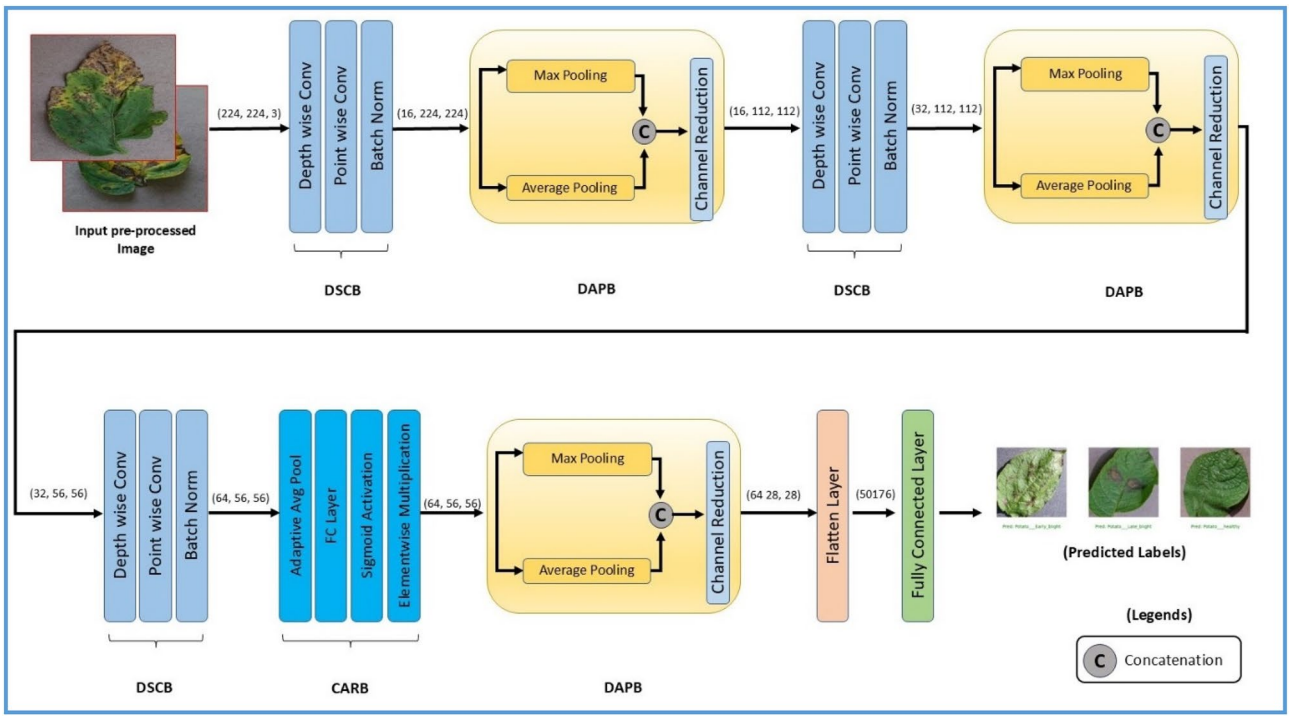

Fig. 3. HPDC Block diagram

attention mechanisms, and novel pooling strategies, resulting in a powerful and efficient model that can handle complex classification tasks while maintaining low computational costs.

Depth-wise Separable Convolution Block (DSCB)

The Depth-wise Separable Convolution Block (DSCB) is the foundational building block of the proposed model HPDC-Net. It extracts complex features from input sample images by incorporating advanced techniques like depth-wise separable convolutions, pointwise convolutions, and batch normalization. The DSCB, being the foundational block of the proposed model, captures both spatial and hierarchical features using depth-wise separable convolutions. This operation drastically reduces the computational load and enhances gradient flow. Batch normalization stabilizes the learning process and enables the model to converge faster. Each layer operation in the architecture contributes in a robust deep feature extraction process and maintains the computational efficiency.

The detail of these layered operations and their significance in the DSCB is elaborated below.

(a) Depth-wise separable convolution

This is the core operation of DSCB, which is decomposed into two separate operations: Depth-wise Convolution and Pointwise Convolution discussed below.

Depth-wise Convolution: In this operation, each filter is applied to each input channel independently. As tomato leaf input image has three RGB channels (Red, Green, Blue), three independent convolution filters are applied to each channel separately. It generates feature map for each filter. At this stage no combination of operation is applied, which reduces the number of operations as compared to conventional convolutional operation where each filter applied to all the input channels simultaneously and produces single feature map as output.Fig. 3HPDC Block diagramFig. 3HPDC Block diagramFig. 3HPDC Block diagram

Mathematical representation of the depth-wise convolutional operation is described by Eq. (1).

$$Output_{depthwise} = \sum_{k=1}^{K} W_k * X_k \tag{1}$$

Here W_k represents the filter applied, X_k represents the input channels and denotes the convolutional operation.

Pointwise Convolution: Pointwise convolution is a key component in depth-wise separable convolution. It is used, followed by a depth-wise convolutional operation, to combine the extracted feature map of each input channel. In this work, pointwise convolution (1×1 convolution) is used to enable the model in learning and to find the relationship between each input channel extracted feature map. The mathematical representation of the pointwise convolutional operation is described by Eq. (2).

$$Output_{pointwise} = W_{pointwise} * Output_{depthwise} \tag{2}$$

(b) Batch normalization

The batch normalization operation is applied immediately after the pointwise convolution operation to stabilize the learning process and enhance the model's generalization ability. This batch normalization operation takes a feature map as input from a pointwise convolution operation and normalizes it by adjusting and scaling the activations to ensure a consistent input distribution for subsequent layers. This process helps to reduce internal covariate shift, enables the model to converge faster, and achieve a higher learning rate. The batch normalization process is mathematically expressed as shown in Eq. (3).

$$Y = \gamma \frac{\mathrm{X} - \mu_B}{\sqrt{\sigma_B^2 - e}} + \beta \tag{3}$$

where X represents the output of pointwise convolution and input to the normalization process, μ_B represents the mean of the batch, σ_B^2 is the variance of the batch, e is a small constant for numerical stability, γ and β are learned scaling and shifting parameters, and finally, the Y is the output tensor after the batch normalization operation.

Dual-Path Adaptive Pooling Block (DAPB)

The pooling mechanism is used for dimensionality reduction in DL-based architectures. Conventional pooling strategies, such as Max Pooling and Average Pooling, are straightforward and computationally efficient, but face limitations like limited feature diversity, inflexibility to varying feature scales, and loss of fine-grained information. In this study, we proposed a Dual-Path Adaptive Pooling Block (DAPB), an innovative pooling strategy that not only performs the dimensionality reduction operation efficiently but also addresses the inherent shortcomings of conventional pooling mechanisms. The DAPB mechanism combines the strength of multiple pooling strategies. It operates by employing two parallel sub-blocks: a dynamic Dual-Path Pooling Block and a static Average Pooling Block, each performing a unique pooling operation. The Dual-Path Pooling block chooses the Average Pooling or Max Pooling operation randomly. This dynamic choice of pooling strategy preserves most of the information during dimensionality reduction. The Average Pooling Block consistently captures smooth and global features through averaging. After the pooling operations, their outputs are concatenated along the channel dimension to merge the complementary strengths of the two approaches. Since this concatenation increases the number of feature channels, a channel reduction layer, implemented using 1×1 convolution, is applied to compress the

feature map to a manageable size, ensuring computational efficiency. The detailed workflow of DAPB is provided in pseudo-code Algorithm 1, and its mathematical representation is given in Eq. (4).

$$Y = \begin{cases} MaxPool\,(X)\,withprobabilityp \\ AvgPool\,(X)\,withprobability\,(1-p) \end{cases} \tag{4}$$

```
1: Input: Feature map x (output from previous layer)
2: Output: Pooled feature map
3: START
4: // Random Value Generation
5: RandomValue = RandomGenerator() // Generates a random number between 0 and 1
6: Pooling Operation Selection:
7:    If (RandomValue > 0.5):
8:        PooledOutput = MaxPooling(x) // Applies MaxPooling operation
9:    else:
10:       PooledOutput = AveragePooling(x) // Applies AveragePooling operation
11: // Return Pooled Feature Map
12: Output = PooledOutput
13: STOP
```

Algorithm 1: Workflow of Dual-Path Adaptive Pooling Block.

Channel-Wise Attention Refinement Block (CARB)

The CARB is a powerful block within the model architecture to enhance the feature extraction process with low computational complexity. It dynamically adjusts the model's focus on the most important features in the plant leaf images, which improves the classification performance of the model. In the first step, it reduces the spatial dimensions of the input feature map using global average pooling. Next, this pooled vector is passed through a sequence of fully connected layers. First layer captures the salient features from the feature vector, second fully connected layer restores the original dimensions of the feature map, and it generates the attention weights for each channel. Finally, these attention weights are applied to the original feature map through channel-wise multiplication. This reweights the feature map, enhancing the most important channels and suppressing the less important ones. The mathematical formulation of CARB is presented in Eq. (5).

$$Attention = \sigma\left(W_2.ReLU\left(W_1.GlobalAvgPool\left(x\right)\right)\right) \tag{5}$$

where, W_1 and W_2 represents the weights of fully connected layers and σ represents the sigmoid activation function.

Preprocessing of input samples

To ensure optimal performance of our proposed model, a series of pre-processing steps are applied to the input images. Two different preprocessing pipelines are used; one is for the training dataset while the other is for the validation and testing datasets. These pipelines perform different preprocessing operations randomly on feed data images, which ensures random data input to the model in each epoch to justify the model's ability to generalize across different samples. Overall preprocessing task representation is given in Algorithm 2.

Training data preprocessing

Before the training phase of the model, each image in the training dataset undergoes some preprocessing operations. At first, each image was randomly cropped and resized to a fixed size of (224 × 224). Next, to induce angular and positional variation in the dataset set each image was randomly flipped in the horizontal direction. These random operations create different versions of the same image in each epoch, which improves the model's generalizability to unseen data samples and prevents the model from overfitting. Then, the data is normalized by calculating the mean difference between each pixel and dividing it by the standard deviation. During the normalization operation each pixel value x in the image is calculated by Eq. (6). Finally, training data samples are randomly shuffled to introduce order variation.

$$x_{normalized} = \frac{x - mean}{std} \tag{6}$$

Validation and test data preprocessing

Images from validation and test datasets are resized to (256 × 256) pixels to ensure consistent evaluation. These resized images are then center-cropped to the dimensions (224 × 224) pixels, capturing the most relevant features for validation and testing. Finally, the dataset was normalized by using the same mean and standard deviation values as used during training to ensure the model performance is evaluated under the same conditions.

```
1: Input: Tomato leaf disease dataset
2: Output: Preprocessed image sample of the dataset
3: START
4: Set the random seed for reproducibility
5: // Training dataset preprocessing
6: for each image in the Training dataset:
7:         Random crop and resize to dimensions (224x224)
8:         Random horizontal flip with a probability of 0.5
9:         Convert the image to a tensor and scale pixel values to the range [0, 1]
10:        Normalize the tensor using mean and standard deviation
11: end for
12: //Validation and test data preprocessing
13: for each image in the validation and test datasets:
14:        Resize the image to dimension (256 × 256)
15:        Crop the center of the resized image to dimensions (224x224)
16:        Convert the image to a tensor and scale pixel values to the range [0, 1]
17:        Normalize the tensor using mean and standard deviation
18: end for
19: Shuffle the training data
20: Generation of preprocessed image
21:  STOP
```

Algorithm 2: Preprocessing algorithm for HPDC-Net model.

This study aims to propose a solution to a critical challenge in tomato disease classification models with better performance for deployment over resource-constrained devices. Our experimental results confirm that the proposed model successfully achieved the balance of low computational requirements with high classification performance. The study encompasses the various data preprocessing techniques to ensure robustness and generalization of the model. The proposed model architecture contains sequentially connected multiple innovative blocks for effective feature extraction and the classification task. This section provides details about the extensive assessment of the outcomes of various experiments intended to evaluate the performance of the proposed model. It includes a description of the experimental setup, details about datasets used during training and evaluation, along with an ablation study that was performed to select the best-performing hyperparameters and architecture at last briefly discusses quantitative performance evaluation with state-of-the-art methods.

Benchmark datasets

In this work, experiments are performed on two tomato leaf disease datasets, which include tomato plant leaf images from a well-known PlantVillage dataset[13] and the Tomato Leaf Disease Detection dataset[28] for training and performance evaluation of our proposed framework. These datasets were chosen because of their public access and wide range of disease classes. Moreover, these datasets reflect real-world diversity, with both balanced and imbalanced class distributions. The PlantVillage dataset is known for its challenging nature due to the imbalanced distribution of images across classes. It comprises leaf images from 14 different crops. However, for our research, we narrowed our focus to tomato leaf disease images, which consist of 18,160 images categorized into 10 distinct classes. Nine of these classes correspond to various tomato leaf diseases, and the remaining one class represents the healthy tomato leaves. A key challenge with this dataset is the class imbalance, which can assess the model's ability to handle biased distributions effectively. For instance, the 'Yellow_Leaf_Curl_Virus' class contains 5357 images, while 'Mosaic_virus' includes only 373 samples. Table 1 illustrates the detailed

Class	Training samples	Validation samples	Test samples	Total samples	Imbalance Level
Bacterial_Spot	1488	425	214	2127	Medium
Early_Blight	700	200	100	1000	High
Late _Blight	1336	381	192	1909	Medium
Leaf_Mold	666	190	96	952	High
Septoria_Leaf_Spot	1239	354	178	1771	Medium
Spider_Mites	1173	335	158	1666	Medium
Target_Spot	982	280	142	1404	Medium
Yellow_Leaf_Curl_Virus	3749	1071	537	5357	Very High
Mosaic_virus	261	74	38	373	Very High
Healthy	1113	318	160	1591	Medium

Table 1. Detailed breakdown of tomato leaf samples from the PlantVillage dataset.

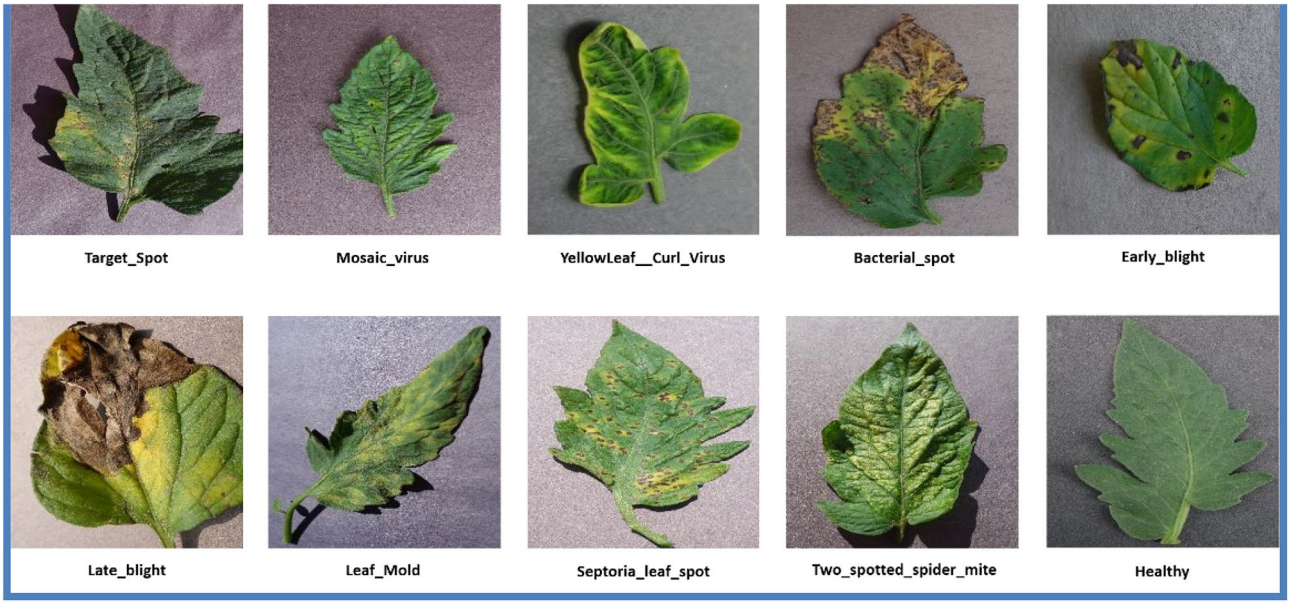

Fig. 4. Representative class samples of tomato leaves from PlantVillage dataset.

Class	Training samples	Validation samples	Test samples	Total samples	Balance
Bacterial_Spot	770	220	110	1100	Balanced
Early_Blight	770	220	110	1100	Balanced
Late _Blight	770	220	110	1100	Balanced
Leaf_Mold	770	220	110	1100	Balanced
Septoria_LeafSpot	770	220	110	1100	Balanced
Spider_Mites	770	220	110	1100	Balanced
Target_Spot	770	220	110	1100	Balanced
Yellow_Leaf_Curl_Virus	770	220	110	1100	Balanced
Mosaic_virus	770	220	110	1100	Balanced
Healthy	770	220	110	1100	Balanced

Table 2. Detailed breakdown of the Tomato Leaf Disease Detection dataset.

description of the dataset, including sample counts per class, their distribution across training, validation, and test sets, as well as the corresponding imbalance levels. A representative sample from each class is shown in Fig. 4. On the other hand, the Tomato Leaf Disease Detection dataset[28] is balanced, with 1100 images per class across the same 10 categories, making it ideal for benchmarking in a uniform training environment. Table 2 displays the number of samples from each class, including the sample distribution in the training, validation, and test samples.

Alongside tomato leaves, we considered it necessary to validate the performance of the proposed model in this study on different domain datasets to assess the adoptability and robustness. To conduct this experiment, we selected smaller and slightly imbalanced potato leaf disease images from PlantVillage dataset. This dataset contains a total of 2152 Potato leaf images, distributed into three classes: Potato_Early_Blight with 1000 images, Potato_Late_Blight with 1000 images, and Potato_healthy with 152 images. The further distribution of this dataset for training and evaluation is shown in Fig. 5. We used the same experimental setup as used for training and evaluation of tomato leaf disease earlier for this performance evaluation. HPDC-Net exhibited satisfactory performance on the potato leaf disease classification task, achieving average test accuracy, precision, recall, and F1-score of 99.54%, 99.67%, 99.67%, 99.67%, respectively. The sample predictions of HPDC-Net on potato leaf images are shown in Fig. 6.

Evaluation metrics

To evaluate the performance of our proposed model, we used various standard evaluation metrics like accuracy, precision, recall, and F1-score. This study addresses a multiclass classification problem. The rate of the correctly predicted positive class is known as True Positive (TP). The rate of correctly identified as the negative class is attributed as True Negative (TN). The wrong predictions for the positive class are termed False Positives (FP).

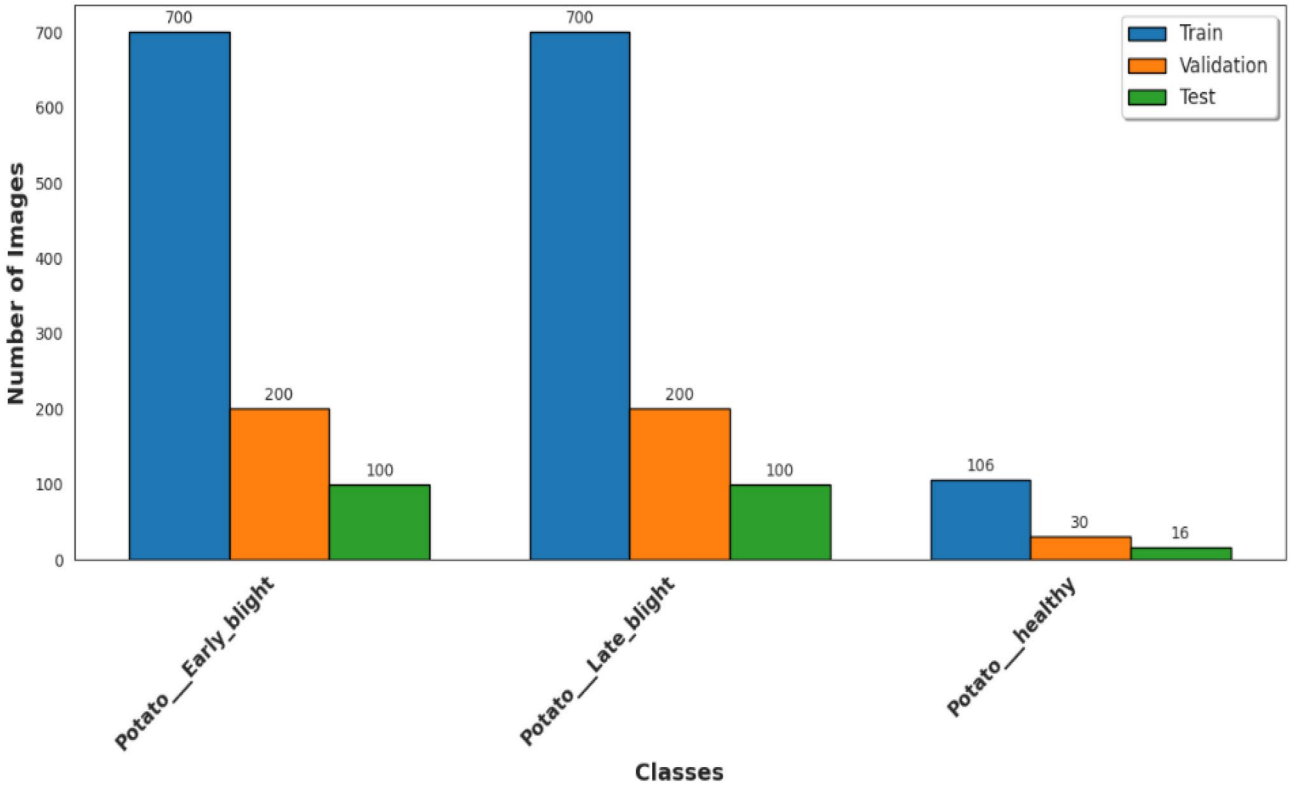

Fig. 5. Potato Leaf Disease Dataset Distribution.

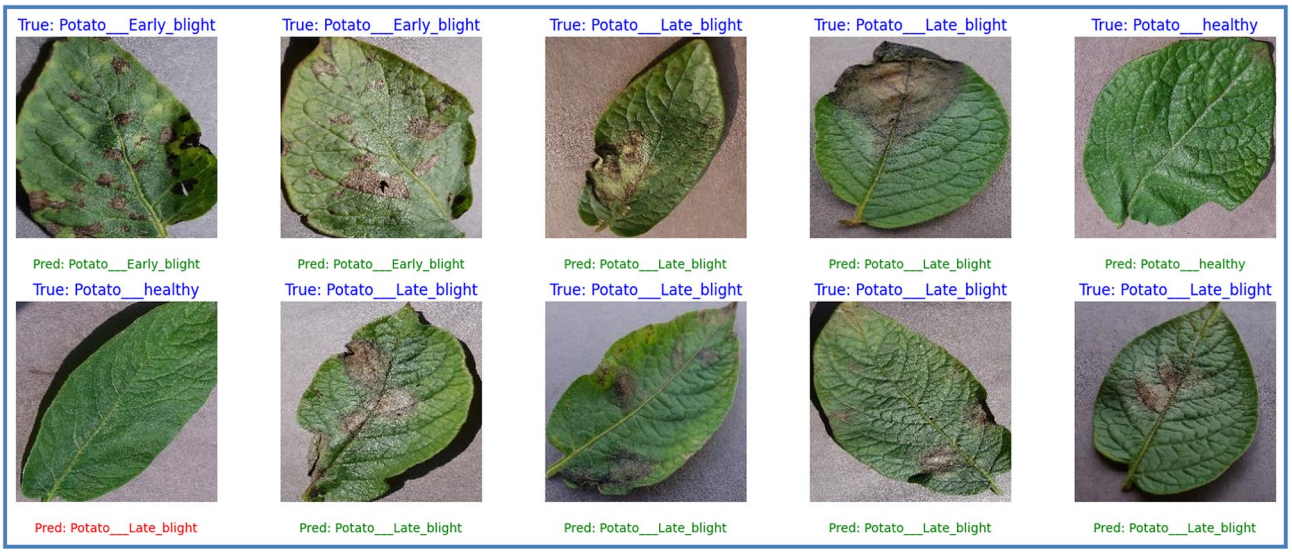

Fig. 6. Sample predictions, original label: Blue, Predicted label (Correct: green, Wrong: Red).

The inaccurate estimation probability of the negative class is False Negative (FN). These standard abbreviations are used to calculate the following evaluation metrics.

(1) Accuracy

Accuracy can also be used to evaluate the effectiveness of the model. Accuracy measures the proportion of correctly classified images out of the total number of images in a dataset. The mathematical description of accuracy is stated in Eq. (7).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

The accuracy score measures the number of correct predictions $(TP + TN)$ made by a model to the total number of predictions $(TP + TN + FP + FN)$ made.

(2) Precision

Precision tells us how many of the predicted positive cases were correct. It measures the proportion of true positive predictions out of all the predictions that were classified as positive by the model and determined by Eq. (8).

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

(3) Recall

Recall tells us the proportion of actual positive cases that were correctly identified by the model. It is measured as Eq. (9).

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

(4) F1-Score

Precision and recall are both used in combination to evaluate model performance in the F1 evaluation process. It is the harmonic mean of recall and precision and is expressed mathematically as in Eq. (10).

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{10}$$

Harmonic mean is used to evaluate F1-Score because harmonic mean is more conservative than arithmetic and geometric mean. Harmonic mean only shows a high value in the result when both precision and recall have high values. If one of the values among precision and recall is too low, then the harmonic mean will be near a low value.

Experimental setup

All experimental studies were conducted on Google Colab, a cloud-based platform reinforced by the CUDA GPU acceleration. This programming environment is equipped with 2 cores, Intel (R) Xeon (R) @ 2.20 GHz CPU, 12.7 GB of system RAM, 112.6 GB of storage, and NVIDIA Tesla T4 having 15 GB RAM. Python 3.10.12 is used as the core of our software infrastructure, and the proposed classification model is built with the open-source deep learning framework PyTorch 2.4.0.

Loss function

In all our experiments, we adopt the Cross-Entropy loss function for classification tasks during the training process. The dataset comprises ten distinct classes, including one class representing healthy leaves and nine classes corresponding to various tomato leaf diseases. This scenario presents a multi-class classification problem. The Cross-Entropy Loss is an essential metric for such tasks, as it effectively quantifies the dissimilarity between the predicted probability distributions and the true class labels. The mathematical formulation of the Cross-Entropy Loss is presented in Eq. (11).

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_i, clog\left(p\left(c|x_i\right)\right) \tag{11}$$

where N denotes the number of samples in the dataset, C represents the total number of classes (in our case, 10 classes corresponding to different tomato leaf diseases), y_i, c is a binary indicator that is 1 if the class label c is the correct classification for the input x_i, and $p\left(c|x_i\right)$ is the predicted probability of the class c for the input x_i.

Ablation study and hyperparameter impact analysis

This study performed a thorough quantitative analysis to find the optimal model architecture and best-performing hyperparameters. This process, known as an ablation study in deep learning, involves systematically testing variations in model architecture and hyperparameters to evaluate how each element affects overall performance. This ablation study was conducted in two parts: (1) architecture-level experiments and (2) hyperparameter optimization. Both types of experiments were aimed at achieving a balance between high classification accuracy and low computational complexity. The architecture-focused experiments involved five different tests. The

Experiment	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	GFLOPs	No of parameters (in millions)
With only average pooling	98.98	98.83	98.66	98.74	0.04	0.51
With only max pooling	98.68	98.38	98.38	98.38	0.04	0.51
With standard convolution	98.13	97.75	97.62	97.67	0.16	0.55
Without second and third pooling block layers	94.63	93.23	93.54	93.16	0.09	8.03
Without third DSCB	96.39	95.59	95.56	95.55	0.03	0.25

Table 3. Architecture-focused experiments (epochs = 200).

Learning rate /optimizer settings	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	GFLOPs	Parameters (millions)
LR = 0.00001 Optimizer = Adam	99.17	99.11	98.90	99.00	0.06	0.52
LR = 0.001 Optimizer = Adam	98.95	98.61	98.63	98.62	0.06	0.52
LR = 0.01 Optimizer = Adam	93.63	92.79	91.50	91.90	0.06	0.52
LR = 0.001 Optimizer = SGD	91.10	88.24	87.71	87.82	0.06	0.52
LR = 0.0001 Optimizer = Adam	**99.40**	**99.34**	**99.19**	**99.26**	**0.06**	**0.52**

Table 4. Parameter-based experiments detail (Epochs = 200). Significant values are in bold.

first two experiments examined the effect of the Hybrid Parallel Pooling mechanism in the proposed HPDC-Net model by deploying either only average pooling or only max pooling, instead of the innovative DAPB. The third experiment evaluated the model behaviour by eliminating the depth-wise separable convolution operation by a standard convolution operation in DSCB. The fourth and fifth experiments varied the number of layers in the model architecture to analyse their impact on performance. Table 3 summarizes how removing or modifying specific blocks affects classification performance and computational complexity. Table 3 results, carefully verified using the ptflops profiler. While it may initially appear counterintuitive that removing only pooling layers increases the parameter count significantly, this is due to the resulting higher-dimensional feature maps, which lead to a greater computational load in the following convolutional layers. In contrast, removing an entire DSCB reduces both the parameter count and GFLOPs, as expected, due to the exclusion of depth-wise and pointwise convolutions. These trends are consistent with the architectural structure of HPDC-Net. The results highlight the critical role of DAPB and DSCB in maintaining both high accuracy and lightweight design. Hyperparameter-based experiments involved variations in model parameters. The best-performing parameters were selected to enhance the model's performance while maintaining computational efficiency. Table 4 presents the hyperparameter scaling analysis to find the optimal balance between the proposed model's performance and its associated parameters. The last experiment in Table 4 results expresses optimal balance between accuracy and resource utilization, these are the selected parameters for our model.

Quantitative performance evaluation with state-of-the-art methods

We conducted a direct, side-by-side comparison of the proposed model for tomato leaf disease classification with the most recent state-of-the-art approaches reported in the literature, which include the[14,16,18,20,24] and[25]. Our proposed model, HPDC-Net outperforms these recent methods, achieving optimal balance of better classification accuracy with limited computational resources. To perform a direct performance evaluation analysis, the proposed model HPDC-Net is trained and evaluated on two different datasets: the PlantVillage tomato leaf disease dataset[13] and the Tomato Leaf Disease Detection dataset[28]. To perform a robust analysis, it is necessary to evaluate the performance of the proposed model for its effectiveness in the classification task and utilization of computational resources. Table 5 present a comparative analysis between the proposed model and several state-of-the-art approaches. This comparison is based on multiple evaluation metrics, including accuracy, recall, precision, F1-score, inference time, number of parameters, and practical suitability. The practical suitability metric is categorized as low, moderate, or high, depending on each model's parameter count and inference time, reflecting its feasibility for deployment in resource-constrained environments.

Authors in[14,18,20], and[25] benchmarked on tomato leaf disease images from PlantVillage Dataset[13], while[16] and[24] the study utilized the Tomato Leaf Disease Detection dataset[28] for training and evaluation of reported methods. Pandiyaraju V et al.[14] reports an accuracy of 98.7%. However, the proposed approach is computationally inefficient due to the use of a transfer learning algorithm, VGG-16, which employs complex standard convolution operations as compared to advanced techniques like depth-wise separable convolutions and group convolution operations. Saleh et al.[18] reported an impressive accuracy of 99.97%. Despite its high accuracy, the practical applicability of this method is limited by its computational complexity, with 6.2 million model parameters. The approach in[18] is computationally inefficient because it uses complex standard convolution operations based DenseNet-77 framework as a backbone network for the CornerNet model. Similarly, Marriam et al.[20] also reported testing accuracy of 99.97% for the proposed classification model. However, like the method in[18] it is computationally

Model	Dataset used	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Inference Time (seconds)	No of parameters (in millions)	Practical Suitability
Ensemble model [14]	PlantVillage	98.07	97.09	98.06	98.07	–	–	Moderate
DenseNet-77-based CornerNet[18]	PlantVillage	99.97	99.62	99.53	99.57	0.22	6.2	Low
ResNet-34-based Faster-RCNN[20]	PlantVillage	99.97	99.48	99.38	99.42	0.23	–	Low
DCNN model[25]	PlantVillage	96.0	96.0	96.0	96.0	–	0.4	Moderate
DenseNet121[24]	Tomato Leaf Disease Detection	98.45	94.32	98.45	98.45	–	–	Low
DTomatoDNet[16]	Tomato Leaf Disease Detection	99.34	96.5	96.6	96.55	–	0.6	Moderate
Purposed method	PlantVillage / Tomato Leaf Disease Detection	99.40 / 99.36	99.34 / 99.37	99.19 / 99.36	99.26 / 99.36	0.0025	0.52	High

Table 5. Quantitative performance comparison with state-of-the-art approaches on Plant Village dataset.

inefficient, requiring 0.23 s of image inference time. The approach in[20] require more computational resources because it also relies on standard convolution layers throughout the ResNet-34 architecture. Hosny et al.[25] proposed a novel lightweight plant leaf disease classification model and with a parameter count of only 0.4 million. Although approach in[25] is computationally efficient but falls short in reported testing accuracy for tomato leaf disease classification, with only 96%.

Naeem et al.[16] introduced a lightweight DL-based classification model which achieved test accuracy of 98.95% with only 0.6 million parameters. The method in[16] Deep Tomato Detection Network (DTomatoDNet) achieved a required balance between better accuracy with minimal resource utilization. However, its reported precision, recall, and F1-score values are too low, achieving a 95% score on the test dataset. Alzahrani et al.[24] employed three deep learning methods and showed the best performing architecture, DenseNet121, with a test accuracy of 98.45%. Reported method in[24] is computationally complex due to its dense connectivity pattern as compared to the model utilizing advanced techniques such as depth-wise separable convolutions. The proposed model for tomato leaf disease classification demonstrates a significant advancement in terms of both classification performance and computational efficiency when compared to recent state-of-the-art methods. Through benchmarking on two widely used datasets, the PlantVillage and the Tomato Leaf Disease Detection dataset, a detailed performance evaluation was conducted against the proposed models[14,18,20,24,25] and[16]. As shown in Table 5, our proposed model HPDC-Net achieved a superior accuracy of 99.40% on the PlantVillage dataset and 99.36% on the Tomato Leaf Disease Detection dataset while maintaining a minimal image inference time and reported parameter count of 0.52 million only. These results highlight the state-of-the-art classification accuracy of our proposed model with computational efficiency, making it a highly effective and practical solution for tomato leaf disease classification tasks on resource-constrained devices.

Confusion matrix

Figure 7 illustrates the confusion matrix to analyze the classification performance of our proposed model. The confusion matrix is an $n \times n$ matrix, which compares the model's predictions with the actual class labels. The rows in a confusion matrix represent the true classes, and the columns indicate the predicted classes.

The confusion matrix in Fig. 6 demonstrates high classification performance of our proposed approach because most of the predictions align along the diagonal, which represents the correctly classified samples. Off-diagonal elements of the matrix represent that the misclassified samples are minimal, which also showcases the model's high classification accuracy.

Discussion

Tomato and Potato plants, being largely cultivated crops worldwide, have huge importance for global food security. Plant pathogens severely impact crop yield and the quality of food. Most farmers rely on visual inspection and expert opinions to identify and manage these diseases. These traditional methods are error-prone and time-consuming. To cope with this limitation, it was essential to automate the plant disease detection and classification task. Numerous studies in the literature have proposed various high-performing methods to detect and classify these diseases. Most of the studies used CNN frameworks, which demand substantial computational resources. To cope with this limitation, we proposed a CPU-friendly DL-based approach, HPDC-Net, which delivers high classification performance utilizing limited computational resources. The experimental results in this study demonstrated superior performance of the proposed HPDC-Net model in terms of classification accuracy and computational efficiency compared to the state-of-the-art approaches utilizing two different tomato leaf disease datasets. As shown in Table 5, our proposed model achieved an average test accuracy of 99.40% on tomato leaf disease images of the PlantVillage dataset while maintaining the model parameter count to only 0.52 million and image inference time of 0.0025 s. Similarly, our proposed model achieved an average test accuracy of 99.36%, model's parameter counts of 0.52 million, and image inference time of 0.0025 s on the tomato leaf disease detection dataset. Furthermore, the proposed model was also trained and evaluated on potato leaf disease images from the PlantVillage dataset to assess its adaptability. The superior performance of HPDC-Net can be attributed to its innovative architectural components, optimal hyperparameter selection through ablation studies, and preprocessing layers that ensure random image sampling in each epoch. Each innovative block of operation in the model architecture is systematically designed to maximize the classification performance while minimizing the computational overhead. HPDC-Net makes significant strides toward addressing the

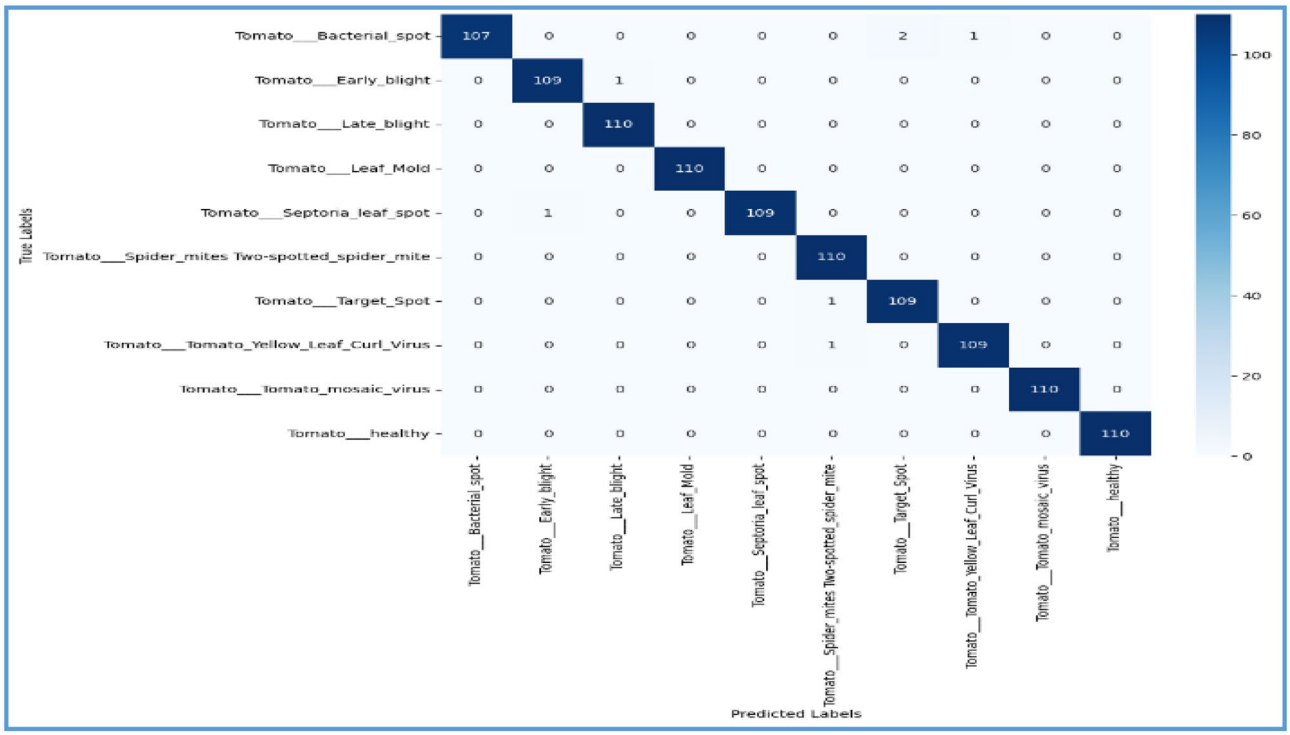

Fig. 7. Confusion matrix results of proposed model.

computational challenges associated with deploying deep learning models in resource-constrained environments. Its lightweight architecture and high accuracy make it a practical solution for resource-constrained devices like smartphones and FPGAs.

Conclusion

It is a tedious job to perceive the plant pathogens by visual inspection. This study proposed an automated tool, HPDC-Net, a novel lightweight deep learning framework designed for scalable and accurate plant leaf disease classification. Our proposed model achieved a superior balance between high classification accuracy and low computational complexity by employing lightweight blocks like DSCBs, DAPBs, and CARBs. HPDC-Net demonstrated remarkable performance on both the Tomato leaf imagery and Potato Leaf image datasets, achieving over 99% accuracy with minimal computational overhead. The adaptability and robustness of the proposed model for tomato disease classification have also been proven by computational complexity testing over both CPU and GPU. However, a key limitation of this study is that the datasets used comprise laboratory-controlled images, which may not fully capture the variability, noise, and environmental complexity encountered in real-world agricultural conditions. Future research should aim to extend the model's capabilities to training on more challenging datasets and testing of various image enhancement effects on the model's behavior. As the model has already achieved 19 FPS classification speed on CPU, effort should be made to make it a bit lighter to achieve a real-time speed of 30 FPS on a standard CPU.

Data availability

The dataset used in this work is publically available. Original Dataset (https://data.mendeley.com/datasets/tywbtsjrjv/1) and Selected Tomato classes for this paper (https://www.kaggle.com/datasets/charuchaudhry/plantvillage-tomato-leaf-dataset).

Received: 4 February 2025; Accepted: 24 June 2025
Published online: 03 September 2025

References

1. J. Bruinsma, "Expert meeting on how to feed the world in 2050 The resource outlook to 2050: 1 By how much do land, water and crop yields need to increase by 2050?".
2. Adigun, O. A. et al. Recent advances in bio-chemical, molecular and physiological aspects of membrane lipid derivatives in plant pathology. *Plant Cell. Environ.* **44**(1), 1–16. https://doi.org/10.1111/PCE.13904 (2021).
3. Shoaib, M. et al. An advanced deep learning models-based plant disease detection: A review of recent research. *Front. Plant Sci.* **14**, 1158933. https://doi.org/10.3389/FPLS.2023.1158933/BIBTEX (2023).
4. Liu, J. & Wang, X. Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model. *Plant Methods* **16**(1), 1–16. https://doi.org/10.1186/S13007-020-00624-2/FIGURES/6 (2020).

5. Liu, J. & Wang, X. Plant diseases and pests detection based on deep learning: a review. *Plant Methods* **17**(1), 1–18. https://doi.org/10.1186/S13007-021-00722-9/TABLES/4 (2021).
6. Taye, M. M. Understanding of machine learning with deep learning: architectures workflow applications and future directions. *Computers* **12**(5), 91. https://doi.org/10.3390/COMPUTERS12050091 (2023).
7. Attri, I., Awasthi, L. K., Sharma, T. P. & Rathee, P. A review of deep learning techniques used in agriculture. *Ecol. Inform* **77**, 102217. https://doi.org/10.1016/J.ECOINF.2023.102217 (2023).
8. Panno, S. et al. A review of the most common and economically important diseases that undermine the cultivation of tomato crop in the mediterranean basin. *Agronomy* **11**(11), 2188. https://doi.org/10.3390/AGRONOMY11112188/S1 (2021).
9. Z.-W. Yuan and J. Zhang, "Feature extraction and image retrieval based on AlexNet," *Eighth International Conference on Digital Image Processing (ICDIP 2016)* vol. 10033 100330E https://doi.org/10.1117/12.2243849 2016
10. C. Szegedy *et al.*, "Going deeper with convolutions," *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 07–12-June-2015 1–9 https://doi.org/10.1109/CVPR.2015.7298594. 2015
11. Kumar, S., Pal, S., Singh, V. P. & Jaiswal, P. Performance evaluation of ResNet model for classification of tomato plant disease. *Epidemiol. Methods* https://doi.org/10.1515/EM-2021-0044/MACHINEREADABLECITATION/RIS (2023).
12. S. R. N. Appe, G. Arulselvi, and G. N. Balaji, "Tomato ripeness detection and classification using VGG based CNN Models, *International Journal of Intelligent Systems and Applications in Engineering* 11 1 296–302, Feb. 2023, Accessed: May 31, 2024. [Online]. Available: https://www.ijisae.org/index.php/IJISAE/article/view/2538
13. "PlantVillage." Accessed: May 25, 2024. [Online]. Available: https://www.kaggle.com/datasets/mohitsingh1804/plantvillage
14. P. V. *et al.*, Improved tomato leaf disease classification through adaptive ensemble models with exponential moving average fusion and enhanced weighted gradient optimization *Front Plant Sci* 15 1382416 https://doi.org/10.3389/FPLS.2024.1382416 (2024)
15. Paul, S. G. et al. A real-time application-based convolutional neural network approach for tomato leaf disease classification. *Array* **19**, 100313. https://doi.org/10.1016/J.ARRAY.2023.100313 (2023).
16. Ullah, N. et al. A lightweight deep learning-based model for tomato leaf disease classification. *Comput. Mater. & Continua* **77**(3), 3969–3992. https://doi.org/10.32604/CMC.2023.041819 (2023).
17. Al-Gaashani, M. S. A. M. et al. Deep transfer learning with gravitational search algorithm for enhanced plant disease classification. *Heliyon* https://doi.org/10.1016/J.HELIYON.2024.E28967/ASSET/626D9E6A-FE55-4382-92EF-3944F4717E81/MAIN.ASSETS/GR11.JPG (2024).
18. Albahli, S. & Nawaz, M. DCNet: DenseNet-77-based cornernet model for the tomato plant leaf disease detection and classification. *Front. Plant Sci.* **13**, 957961. https://doi.org/10.3389/FPLS.2022.957961/BIBTEX (2022).
19. Al-Gaashani, M. S. A. M., Muthanna, A., Chelloug, S. A. & Kumar, N. EAMultiRes-DSPP: An efficient attention-based multi-residual network with dilated spatial pyramid pooling for identifying plant disease. *Neural. Comput. Appl.* **36**(26), 16141–16161. https://doi.org/10.1007/S00521-024-09835-3/METRICS (2024).
20. Nawaz, M. et al. A robust deep learning approach for tomato plant leaf disease localization and classification. *Sci.Rep.* **12**(1), 1–18. https://doi.org/10.1038/S41598-022-21498-5 (2022).
21. Abdulridha, J., Ampatzidis, Y., Qureshi, J. & Roberts, P. Laboratory and UAV-based identification and classification of tomato yellow leaf curl, bacterial spot, and target spot diseases in tomato utilizing hyperspectral imaging and machine learning. *Remote Sens.* **12**(17), 2732. https://doi.org/10.3390/RS12172732 (2020).
22. Albattah, W., Nawaz, M., Javed, A., Masood, M. & Albahli, S. A novel deep learning method for detection and classification of plant diseases. *Complex & Intell. Syst.* **8**(1), 507–524. https://doi.org/10.1007/S40747-021-00536-1/TABLES/9 (2022).
23. Islam, M. S. et al. Multimodal hybrid deep learning approach to detect tomato leaf disease using attention based dilated convolution feature extractor with logistic regression classification. *Sensors* **22**(16), 6079. https://doi.org/10.3390/S22166079 (2022).
24. Alzahrani, M. S. & Alsaade, F. W. Transform and deep learning algorithms for the early detection and recognition of tomato leaf disease. *Agronomy* **13**(5), 1184. https://doi.org/10.3390/AGRONOMY13051184/S1 (2023).
25. Hosny, K. M., El-Hady, W. M., Samy, F. M., Vrochidou, E. & Papakostas, G. A. Multi-class classification of plant leaf diseases using feature fusion of deep convolutional neural network and local binary pattern. *IEEE Access* **11**, 62307–62317. https://doi.org/10.1109/ACCESS.2023.3286730 (2023).
26. S. C.K., J. C.D., and N. Patil, Tomato plant disease classification using Multilevel Feature Fusion with adaptive channel spatial and pixel attention mechanism *Expert Syst Appl* 228 120381 https://doi.org/10.1016/J.ESWA.2023.120381. (2023)
27. Al-Gaashani, M. S. A. M. et al. MSCPNet: A multi-scale convolutional pooling network for maize disease classification. *IEEE Access* https://doi.org/10.1109/ACCESS.2024.3524729 (2024).
28. "Tomato leaf disease detection." Accessed: Dec. 03, 2024. [Online]. Available: https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf

Author contributions
All authors contributed equally in this work. Thank you.

Funding
This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023–00218176) and the Soonchunhyang University Research Fund. The authors extend their appreciation to the Deanship of Research and Graduate Studies at King Khalid University for funding this work through Large Research Project under grant number RGP.2/275/46.

Declarations

Competing interests
The authors declare that they have no competing interests to report regarding the present study.

Additional information
Correspondence and requests for materials should be addressed to M.A.K. or Y.N.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.