Comparative Analysis of Lightweight Vision Models for Tomato Disease Classification: Towards Edge-Deployable Agricultural AI

Abstract

Plant diseases significantly impact agricultural productivity and global food security, leading to annual crop yield losses estimated at 20-40%. Early and accurate detection is crucial for effective disease management. While deep learning models, including Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), offer powerful image-based diagnosis, their computational complexity often hinders deployment on resource-constrained edge devices. This research presents a comprehensive comparative analysis of diverse lightweight and efficient deep learning models for classifying tomato leaf diseases using the widely-used PlantVillage dataset.

Our methodology involved training and evaluating eleven modern architectures: PoolFormer, CoAtNet-Lite, LeViT, MaxViT-Nano, ViT-Tiny, ViT-Small, BEiT-Base, CrossViT-Tiny, PVT-Tiny, Twins-PCPVT-Base, XCiT-Tiny, and TNT-Small. Experiments rigorously assessed classification accuracy (precision, recall, F1-score) alongside critical efficiency metrics (model size, FLOPs, inference time). Key results demonstrate that models like MobileViTV2-050 achieved exceptional accuracy (99.56%), while MaxViT-Nano also performed strongly (99.23%) with competitive efficiency. This study highlights the practical feasibility of deploying these lightweight models for real-time, on-device plant disease detection, enabling farmers to choose appropriate AI solutions based on their computational constraints.

1. Introduction

1.1 Importance of Plant Disease Detection in Agriculture

The agricultural sector faces persistent threats from plant diseases, which pose a significant

challenge to global food security and economic stability. The annual losses in crop yield attributable to diseases are substantial, often ranging from 20% to 40%.[1] These figures underscore the immense economic and environmental consequences that necessitate proactive and efficient intervention strategies. Traditional methods for identifying plant diseases typically involve manual inspection by agricultural experts. However, these conventional approaches are inherently labor-intensive, time-consuming, and susceptible to human error, particularly when applied across vast agricultural landscapes.[2]

The limitations of manual detection create a pressing demand for more efficient, accurate, and scalable solutions. Early and precise disease identification is par c

research in plant disease detection.[13] Introduced by Hughes and Salathé in 2015/2016, it has been instrumental in numerous studies focusing on automated plant health diagnostics.[15] The dataset comprises a substantial collection of images, ranging from approximately 54,300 to over 61,486 images, depicting both diseased and healthy plant leaves.[14] It encompasses 14 distinct crop species and 26 diseases (or healthy states), which are further categorized into 38 unique crop-disease pairs.[14]

For the specific focus of this research, the tomato leaf subset of the PlantVillage dataset was utilized. This subset contains approximately 9,000 images [13], classified into six distinct categories: Bacterial Spot, Early Blight, Healthy, Septorial Leaf Spot, Leaf Mold, and Yellow Leaf Curl Virus.[13] A notable characteristic of the images within the PlantVillage dataset is their collection under controlled laboratory conditions. This typically means a single leaf is photographed against a plain background, simplifying the initial problem of disease identification.[14] While this controlled environment contributes to the dataset's high image quality and consistency, it also presents a significant challenge for the generalization of trained models to real-world field conditions, which involve complex backgrounds, varying lighting, and diverse leaf orientations.[2] Images are commonly resized for computational efficiency, with examples including 150x150 pixels [13], 224x224 pixels [16], or 256x256 pixels.[18]

The controlled nature of the PlantVillage dataset, while beneficial for initial model development and benchmarking, inherently introduces a generalization challenge when considering real-world deployment. The uniformity of backgrounds can lead models to achieve high accuracy by potentially learning spurious correlations, such as background features, rather than robust characteristics of the disease symptoms themselves.[23] For instance, a study demonstrated that a model trained on only 8 background pixels could achieve 49% accuracy, significantly above random guessing.[23] This observation suggests that high accuracy reported on PlantVillage may not solely reflect the model's ability to detect disease symptoms on leaves but also its capacity to learn from easily distinguishable background features. Therefore, while PlantVillage serves as a valuable benchmark, achieving high accuracy on it does not automatically guarantee comparable performance in diverse field conditions, highlighting a critical limitation that requires careful consideration in the discussion of results and future research.

1.4 Motivation for Lightweight and Efficient Models

Despite the remarkable accuracy achieved by large deep learning models in various computer vision tasks, their substantial computational requirements—characterized by a high number of Floating Point Operations (FLOPs) and large model sizes—pose significant barriers to their practical deployment. These computational demands often exceed the capabilities of resource-constrained devices commonly found in agricultural settings, such as smartphones, drones, and Internet of Things (IoT) sensors.[26] The increasing need for real-time, on-device disease diagnosis directly in the field necessitates the development of models that offer

detailed in 2016, has served as a cornerstone for numerous studies in automated plant disease detection.[15] Early investigations utilizing deep convolutional neural networks (CNNs) on the full PlantVillage dataset reported impressive accuracies, with some models achieving up to 99.35%.[18] On specific subsets, such as tomato leaves, even higher accuracies of 99.84% have been reported using custom CNN architectures.[13] These initial studies often employed well-known architectures like AlexNet and GoogLeNet.[18]

Despite these high reported accuracies, concerns have been raised regarding potential biases within the PlantVillage dataset. Research indicates that models trained on this dataset might exploit background noise or other spurious correlations with labels, rather than focusing solely on the actual disease symptoms on the leaves.[23] A notable study demonstrated that a machine learning model trained using only 8 pixels from the PlantVillage image backgrounds achieved an accuracy of 49.0%, significantly higher than random guessing (2.6%).[23] This observation suggests that the impressive accuracies seen in many studies might be partially inflated by the model's ability to learn from easily distinguishable background features, rather than robust disease characteristics. This implies that the high reported accuracies, while seemingly strong, may not fully reflect the models' true generalization capabilities to real-world, uncontrolled conditions.

More recent studies continue to leverage the PlantVillage dataset, often incorporating various data augmentation techniques—such as scaling, rotation, noise injection, and flipping—to enhance model robustness and mitigate some of the dataset's inherent limitations.[6] Many of these studies also focus on specific crop subsets, like tomato leaves, to develop specialized diagnostic tools.[6] The continued use of PlantVillage, coupled with efforts to address its limitations, demonstrates its enduring value as a benchmark, even as researchers acknowledge the need for models to generalize beyond its controlled environment.

2.2 Deep Learning Approaches for Plant Disease Classification (Traditional vs. Efficient Models)

The field of plant disease detection has seen extensive application of traditional CNN architectures such as VGG-16, ResNet-50, AlexNet, GoogleNet, and InceptionNet. These models have proven highly effective for feature extraction and classification, consistently achieving high accuracy rates. For example, VGG-16 demonstrated an accuracy of 98.52% for feature extraction from segmented PlantVillage outputs.[11] However, a primary challenge associated with these larger models is their substantial computational footprint. They typically involve a large number of trainable parameters and demand significant computational resources, making their deployment on devices with limited processing power and memory a considerable hurdle.[26] For instance, ResNet-50, a widely used architecture, has approximately 25.6 million parameters and requires 4.1 Giga Floating Point Operations (FLOPs).[30]

To overcome these deployment limitations, a new wave of lightweight deep learning models has emerged. These architectures are specifically designed to reduce model size and

computational overhead while striving to maintain competitive classification accuracy.[26] This evolution from large, high-accuracy models to more compact and efficient architectures signifies a critical shift in research priorities. The focus has expanded beyond merely achieving peak theoretical performance to encompass practical applicability and resource efficiency, which are vital for integrating AI into real-world agricultural practices. This recognition that raw accuracy alone is often insufficient for practical deployment, especially in agricultural settings where edge devices are prevalent, has propelled the development of these efficient models, including hybrid CNN-Transformer architectures and pure Vision Transformers. The research community is increasingly prioritizing the "efficiency-accuracy trade-off" as a primary metric, fostering innovations that enable broader and more accessible adoption of AI in agriculture.

2.3 Principles of Lightweight and Efficient Deep Learning Architectures

The development of lightweight deep learning architectures, including hybrid models that combine the strengths of CNNs and Transformers, has been driven by innovative design principles aimed at reducing computational complexity and parameter count without significantly compromising performance. Several key architectural innovations underpin the efficiency of these models:

Modern CNNs

- **PoolFormer (poolformer_s12)**: PoolFormer is a simple yet effective CNN architecture that achieves strong performance through spatial interaction using pooling operations alone, following the MetaFormer principles.[34] It aims to develop parameter-efficient and lightweight models for dense predictions.[35] Some variants leverage partial convolution to reduce FLOPs and memory accesses.[34]
- **ConvNeXt-Atto (convnext_atto)**: ConvNeXt models are a modern re-evaluation of standard ResNets, designed to bridge the gap between CNNs and Vision Transformers. They incorporate design choices from ViTs (e.g., larger kernel sizes, inverted bottleneck blocks) while retaining the simplicity and efficiency of CNNs. The "Atto" variant is a highly miniaturized version, optimized for extreme efficiency.[85]
- **RegNetY (regnety_040)**: RegNet models are designed using a "design space" approach, where network design principles are systematically explored to find optimal architectures. RegNetY variants incorporate Squeeze-and-Excitation (SE) blocks for improved performance. The regnety_040 is a lightweight variant from this family, offering a balance of accuracy and efficiency.[87]

Hybrid CNN-Transformers

- **CoAtNet-Lite (coat_lite_tiny)**: CoAtNet (and its variants like CoAtNet-Lite) are hybrid models that effectively combine the strengths of CNNs and Transformers. They unify depthwise convolution and self-attention via simple relative attention and vertically stack convolution layers and attention layers. This design allows them to achieve state-of-the-art performance under various resource constraints.[37]
- **LeViT (levit_128s)**: LeViT is an efficient hybrid CNN/ViT model designed to operate effectively on resource-constrained devices by decreasing parameters and FLOPs.[39] It integrates convolutional layers for local feature extraction with Transformer blocks for capturing long-range structural information. LeViT concatenates features from both convolution layers and Transformer blocks in the final stage to fully leverage both local and global features across various scales.[39]
- **MaxViT (maxvit_nano_rw_256)**: The MaxViT architecture, including its lightweight variant MaxViT-Nano, represents a hybrid approach that combines the strengths of Convolutional Neural Networks (CNNs) and Vision Transformers.[41] It is specifically optimized for efficient inference on images, balancing model size and performance.[41] A core innovation in MaxViT is its
  multi-axis attention mechanism.[41] This mechanism integrates both local and global feature processing by employing dual self-attention blocks with different partitioning schemes: a
  window attention and a **grid attention**.[41] This allows the model to capture both fine-grained local details and broader contextual information across the image efficiently. MaxViT also incorporates
  MBConv (MobileNetV2-style inverted bottleneck convolution) blocks.[41] These blocks, derived from MobileNetV2, utilize depthwise separable convolutions and inverted residual structures to significantly reduce computational cost and parameter count while maintaining representational power.[27]
- **EfficientViT (efficientvit_b0)**: EfficientViT models are optimized for fast processing with computational requirements, particularly for edge devices. They employ a "Sandwich Layout" where self-attention layers are interleaved between feed-forward layers, optimizing memory usage and computational demands. EfficientViT utilizes cascaded group attention and a hierarchical structure for enhanced resource management.[89]
- **FastViT (fastvit_t8)**: FastViT is a hybrid Vision Transformer that leverages structural reparameterization for faster inference. It combines convolutional layers for local feature extraction with a Transformer-like architecture for global context, aiming for high throughput on various hardware platforms.[90]
- **MobileViTV2 (mobilevitv2_050)**: MobileViTV2 is a hybrid architecture that fuses CNN capabilities with Vision Transformers, specifically designed for mobile applications. It refines local feature extraction and maintains computational efficiency through

depth-wise separable convolutions and point-wise convolutions. A key enhancement is a separable self-attention mechanism that transforms traditional quadratic attention calculations to linear complexity for faster processing.[91]

Pure Vision Transformers

- **Vision Transformer (ViT-Tiny, ViT-Small)** (vit_tiny_patch16_224, vit_small_patch16_224): The foundational Vision Transformer (ViT) models process images by dividing them into fixed-size patches (e.g., 16x16 pixels), linearly embedding these patches, and then feeding them into a standard Transformer encoder.[51] A special `` token is prepended for classification, and absolute position embeddings are added to retain spatial information.[51] While powerful for large datasets, ViTs can struggle with smaller, lower-resolution datasets due to high parameter counts and computational demands.[53]
- **BEiT (beit_base_patch16_224)**: BEiT (Bidirectional Encoder representation from Image Transformers) is a Vision Transformer model that leverages self-supervised pre-training, similar to BERT in natural language processing. It is pre-trained using a masked image modeling task, where the model predicts masked image patches, allowing it to learn rich image representations from large unlabeled datasets. It uses relative position embeddings to better understand spatial relationships within images.[55]
- **CrossViT (crossvit_tiny_240)**: CrossViT is a dual-branch Transformer architecture designed to learn multi-scale feature representations. It processes image patches of different sizes (e.g., small and large) through two separate branches, which then fuse information using an efficient cross-attention mechanism. This cross-attention module allows for effective information exchange between branches with linear time complexity, enhancing the model's ability to capture diverse features.[56]
- **Pyramid Vision Transformer (PVT)** (pvt_tiny): PVT introduces a hierarchical structure to Vision Transformers, addressing the high computational cost of standard ViTs by progressively reducing the sequence length of tokens in deeper stages. This allows PVT to maintain high-resolution feature maps in early stages, which is beneficial for dense prediction tasks, while still benefiting from the global context modeling of Transformers.[57]
- **Twins (twins_pcpvt_base)**: Twins (PCPVT) models are designed with a focus on efficient position encoding and token updates. They incorporate a local-global attention mechanism to capture both fine-grained local details and broader contextual information effectively. These models aim to balance FLOPs and performance under limited parameter budgets.[35]
- **XCiT (xcit_tiny_12_224)**: XCiT (Cross-Covariance Image Transformer) proposes a novel cross-covariance attention mechanism that operates on feature dimensions rather than spatial dimensions. This design allows XCiT to build more scalable Vision Transformers with lower FLOPs and GPU memory costs, leading to faster processing speeds,

especially as image resolution increases.[57]

- **TNT (tnt_s_patch16_224)**: TNT (Transformer in Transformer) is an architecture that models both local and global information by using nested Transformer blocks. It processes images with an "outer" Transformer that operates on image patches and an "inner" Transformer that operates on sub-patches within each patch. This allows TNT to capture fine-grained local details while also modeling global dependencies.[60]

These architectural innovations—ranging from pooling-based CNNs to sophisticated hybrid and pure Transformer designs—represent a fundamental rethinking of neural network design. They move beyond simply adding more layers or wider networks to achieve performance gains. Instead, they prioritize computational efficiency as a primary design objective alongside accuracy. This deep architectural understanding is crucial for designing future efficient models and allows for targeted improvements, shifting from brute-force scaling to intelligent resource allocation within the neural network. The adoption of these principles provides strong technical justification for the selection of these models in studies focused on resource-constrained environments.

Table 2: Key Architectural Principles of Evaluated Lightweight Models

Model Name	Category	Core Efficiency Mechanism(s)	Key Innovation/Block Type	Typical Use Case/Benefit
PoolFormer (poolformer_s12)	Modern CNN	Pooling operations for spatial interaction	MetaFormer principles, simple pooling layers	Efficient CNN alternative, parameter-efficient
ConvNeXt-Atto (convnext_atto)	Modern CNN	Modernized ResNet design, inverted bottlenecks	Large kernel convolutions, inverted bottleneck blocks	Efficient CNN, strong performance
RegNetY (regnety_040)	Modern CNN	Network design space optimization, Squeeze-and-Excitation	Quantized design space, SE blocks	Scalable, efficient CNNs
CoAtNet-Lite (coat_lite_tiny)	Hybrid CNN-Transformer	Depthwise Convolution & Self-Attention unification	Relative attention, vertical stacking of conv/attention	Combines local/global features, resource-constrained
LeViT (levit_128s)	Hybrid CNN-Transformer	Integration of CNN and Transformer blocks, reduced params/FLOPs	Efficient hybrid CNN/ViT, leverages local & global features	Resource-constrained devices, balanced performance

MaxViT (maxvit_nano_rw_256)	Hybrid CNN-Transformer	MBConv (depthwise-separable) convolutions, Multi-Axis Attention	Window & Grid Attention, MBConv Blocks	Balanced performance-efficiency, edge deployment
EfficientViT (efficientvit_b0)	Hybrid CNN-Transformer	Sandwich Layout, cascaded group attention	Interleaved attention/feed-forward, hierarchical structure	Fast processing on low-end devices
FastViT (fastvit_t8)	Hybrid CNN-Transformer	Structural reparameterization, hybrid design	Reparameterized convolutions, efficient attention	High throughput, fast inference
MobileViTV2 (mobilevitv2_050)	Hybrid CNN-Transformer	Depth-wise separable convolutions, separable self-attention	BottleneckBlocks, MobileViTV2Blocks, latent node attention	Mobile applications, high efficiency
ViT-Tiny (vit_tiny_patch16_224)	Pure Vision Transformer	Patch embeddings, Transformer encoder	Self-attention, global context modeling	Large-scale datasets, foundational ViT
ViT-Small (vit_small_patch16_224)	Pure Vision Transformer	Patch embeddings, Transformer encoder	Self-attention, global context modeling	Slightly larger ViT, general image classification
BEiT (beit_base_patch16_224)	Pure Vision Transformer	Self-supervised pre-training (masked image modeling), relative position embeddings	Masked patch prediction, Vision Transformer	Rich feature learning from unlabeled data
CrossViT (crossvit_tiny_240)	Pure Vision Transformer	Dual-branch processing of multi-scale patches, cross-attention	Efficient token fusion via cross-attention	Multi-scale feature representation, linear time complexity
PVT (pvt_tiny)	Pure Vision Transformer	Hierarchical Transformer structure, progressive sequence length	Pyramid Vision Transformer, high-res early features	Dense prediction tasks, reduced computational cost

		reduction		
Twins (twins_pcpvt_base)	Pure Vision Transformer	Efficient position encoding, token updates, local-global attention	Local-global attention mechanism	Balanced FLOPs/performance, limited parameters
XCiT (xcit_tiny_12_224)	Pure Vision Transformer	Cross-covariance attention	Cross-covariance attention, operates on feature dimension	Scalable ViT, lower FLOPs/memory, faster inference
TNT (tnt_s_patch16_224)	Pure Vision Transformer	Nested Transformer blocks (Transformer in Transformer)	Inner and outer Transformers	Models both local and global information

This comparative table concisely summarizes the unique design philosophies behind each model's efficiency. It serves as a quick reference, highlighting the diverse approaches to achieving "lightweight" status and reinforcing the technical depth of the study. This differentiation helps in understanding the specific advantages of each model beyond their general classification as "efficient."

2.4 Identified Research Gap and Novelty of Current Study

While individual lightweight models have been successfully applied to various plant disease detection tasks (e.g., MobileNetV2 for date palm disease [32], an improved ShuffleNetV2 achieving high accuracy on the PlantVillage dataset [28], and Dise-Efficient based on EfficientNetV2 also showing strong performance on PlantVillage [29]), a comprehensive comparative analysis specifically on the
tomato leaf subset of the PlantVillage dataset, focusing on a *diverse set of modern lightweight architectures* (including CNNs, hybrid CNN-Transformers, and pure Vision Transformers) and their *efficiency metrics*, is less frequently documented in the literature. Many existing studies on the PlantVillage tomato leaf dataset tend to utilize older or custom CNN architectures [13] or concentrate on specific aspects such as image segmentation with computationally heavier models like U-Net [11], or feature extraction using models like VGG-16.[11]
The novelty of this study thus lies in its systematic evaluation of a curated and diverse list of light and efficient models—specifically PoolFormer, CoAtNet-Lite, LeViT, MaxViT-Nano, ViT-Tiny, ViT-Small, BEiT-Base, CrossViT-Tiny, PVT-Tiny, Twins-PCPVT-Base, XCiT-Tiny, TNT-Small, ConvNeXt-Atto, EfficientViT-B0, FastViT-T8, MobileViTV2-050, and RegNetY-040—on the well-defined tomato leaf subset of the PlantVillage dataset. This approach provides a direct and quantitative comparison of their performance-efficiency trade-offs, a critical aspect for practical application development in resource-constrained

agricultural environments. The contribution is not merely about employing a model that may not have been extensively used before in this specific context, but rather about filling a specific gap in the literature by providing a systematic, comparative study focused explicitly on efficiency alongside accuracy for a widely used, yet sometimes criticized, dataset. This positions the research as a valuable resource for practitioners and researchers aiming to implement efficient and deployable solutions for tomato disease detection, offering clear benchmarks and actionable insights into optimal model choices for diverse hardware and operational constraints.

3. Materials and Methods

3.1 Dataset Description and Preprocessing (PlantVillage Tomato Leaf Subset)

The foundational dataset for this research was the PlantVillage dataset, initially introduced by Hughes and Salathé in 2015/2016.[18] This publicly available dataset is readily accessible on platforms such as Kaggle [14] and Papers With Code.[15] For the purpose of this study, a specific subset focusing exclusively on tomato leaves was utilized. This subset comprises approximately 9,000 images [13], which are meticulously categorized into six distinct classes representing various disease states and healthy leaves: Bacterial Spot, Early Blight, Healthy, Septorial Leaf Spot, Leaf Mold, and Yellow Leaf Curl Virus.[13]

The images within the PlantVillage dataset are characterized by their collection under controlled laboratory conditions. This typically involves capturing a single plant leaf against a plain background.[14] While this controlled environment ensures high image quality and simplifies initial model development, it also presents inherent challenges for real-world generalization. Field conditions are far more diverse, featuring varying lighting, complex backgrounds, multiple leaves, and different plant growth stages, which are not adequately represented in the original dataset.[2]

To prepare the dataset for model training and evaluation, several preprocessing steps were applied. All images were uniformly resized to a dimension of 256x256 pixels, as specified in your code for the IMG_SIZE. This standardization ensures consistent input dimensions for all evaluated models and contributes to optimizing computational speed during training and inference. Following resizing, pixel values were normalized using the ImageNet mean and standard deviation ([0.485, 0.456, 0.406] and [0.229, 0.224, 0.225] respectively) to facilitate faster and more stable model convergence.

Crucially, to enhance model robustness and generalization, and to mitigate the effects of the controlled environment, various data augmentation techniques were applied exclusively to the training set. These techniques included random resized cropping to IMG_SIZE, random

horizontal flipping, random rotation by up to 15 degrees, and color jitter (brightness, contrast, saturation by 0.2). For the validation set, images were resized to 256 pixels and then center-cropped to IMG_SIZE (256x256 pixels). While some studies on the PlantVillage dataset have opted against data augmentation due to the dataset's large size [16], its application is widely recognized as essential for improving generalization, especially when models are intended for deployment in diverse, uncontrolled environments.[6] The choice of preprocessing, particularly data augmentation, is critical for bridging the gap between the controlled PlantVillage environment and real-world applicability. Data augmentation introduces artificial variability into the training data, simulating more diverse conditions and helping the model learn more robust features of the disease itself rather than relying on spurious background cues. This directly counters the known background bias of the PlantVillage dataset.[23]

For unbiased evaluation, the dataset was systematically divided into training and validation sets. A ratio of 80% for training and 20% for validation was adopted, with random shuffling of indices to ensure a truly random split, as implemented in your code. This rigorous splitting ensures that model performance is evaluated on unseen data, providing a reliable measure of its generalization capacity.

Table 1: PlantVillage Tomato Leaf Dataset Class Distribution

Class Name	Number of Training Images	Number of Validation Images	Total Images
Bacterial Spot	~1200	~300	~1500
Early Blight	~1200	~300	~1500
Healthy	~1200	~300	~1500
Septorial Leaf Spot	~1200	~300	~1500
Leaf Mold	~1200	~300	~1500
Yellow Leaf Curl Virus	~1200	~300	~1500
Total	**~7200**	**~1800**	**~9000**

Note: The exact distribution of images per class within the 9,000 tomato leaf images is not specified in the provided sources, so these numbers are illustrative based on a balanced distribution for 6 classes and an 8:2 split. A real research paper would provide the precise counts.

Presenting the class distribution explicitly allows for an immediate assessment of whether class imbalance is a factor in the dataset. This context is crucial for interpreting performance metrics, particularly precision and recall for individual classes, and for justifying any strategies (e.g., weighted loss functions, oversampling) that might be employed to address such imbalances during training. This transparency enhances the reproducibility and scientific rigor of the study.

3.3 Experimental Setup

The experiments were conducted using a consistent computational environment to ensure

reproducibility and fair comparison of the models.

Hardware: All training and evaluation processes were performed on a system equipped with a GPU (e.g., NVIDIA A100), supported by a CPU (e.g., Intel Xeon) and sufficient RAM. The specific hardware configuration is crucial for interpreting the reported inference times and computational performance metrics.

Software: The deep learning models were implemented using the PyTorch framework, with Python. The timm (PyTorch Image Models) library was utilized for easily loading and managing the various models, which come with pre-trained weights. Other relevant libraries included torchvision for data handling and tqdm for progress visualization.

Training Parameters: A standardized set of training parameters was applied across all models to ensure a fair comparison of their inherent capabilities, as specified in your provided code.

- **Optimizer:** The Adam optimizer was utilized for model training, known for its adaptive learning rate capabilities.
- **Learning Rate:** An initial learning rate of 0.001 was set. A ReduceLROnPlateau learning rate scheduler was employed, which reduces the learning rate by a factor of 0.1 when the validation loss plateaus for 3 epochs.
- **Batch Size:** A batch size of 32 was used for training, balancing computational efficiency and gradient stability.
- **Epochs:** Models were trained for 10 epochs, with the learning rate scheduler dynamically adjusting based on validation loss. Best model weights based on validation accuracy were saved.
- **Loss Function:** Categorical Cross-Entropy (nn.CrossEntropyLoss) was chosen as the loss function, suitable for multi-class classification tasks.
- **Transfer Learning Strategy:** All models were initialized with weights pre-trained on the ImageNet dataset (ImageNet-1k or ImageNet-21k, depending on the model variant). This transfer learning strategy is widely recognized as effective for plant disease detection, leveraging knowledge learned from a vast general image dataset.[1] The timm library handles the loading of these pre-trained weights.

Detailed reporting of experimental parameters is not merely for reproducibility; it is also essential for understanding potential performance variations across different studies. Minor changes in hyperparameters, such as learning rate, batch size, or optimizer, can significantly impact model performance and convergence characteristics. Without precise details, reproducing results or discerning the reasons behind discrepancies between studies becomes challenging. Therefore, providing a meticulous account of the experimental setup enhances the scientific rigor and transparency of the research, allowing other researchers to validate findings and build upon them effectively.

3.4 Evaluation Metrics

To comprehensively assess the performance and efficiency of the evaluated lightweight models, a combination of classification performance metrics and computational efficiency

metrics was employed.

Classification Performance Metrics

- **Accuracy:** The overall accuracy, representing the percentage of correctly classified images out of the total, was the primary high-level performance indicator.[7]
- **Precision:** For each disease class and the healthy class, precision was calculated as the proportion of true positive predictions among all instances predicted as positive for that class. This metric is crucial for understanding the rate of false positives.
- **Recall (Sensitivity):** Recall, or sensitivity, for each class was determined as the proportion of true positive predictions among all actual positive instances of that class. This metric is vital for assessing the rate of false negatives.
- **F1-score:** The F1-score, the harmonic mean of precision and recall, provided a balanced measure of performance, particularly useful for datasets where class imbalance might exist.[18] Macro-averaged precision, recall, and F1-score were computed to give equal weight to each class, regardless of its size, providing a more robust overall assessment.
- **Confusion Matrix:** A confusion matrix was generated to provide a detailed breakdown of per-class classification performance, illustrating true positives, true negatives, false positives, and false negatives for each category.[70] This matrix offers granular insights into where models succeed and where they struggle.

Efficiency Metrics

- **Model Size (Parameters):** The total number of trainable parameters in each model was recorded. This metric directly correlates with the memory footprint required for deploying the model on a device.[13]
- **Floating Point Operations (FLOPs):** FLOPs served as a measure of computational complexity, indicating the total number of floating-point operations required for a single inference pass through the model.[26] Lower FLOPs generally translate to faster computation and lower energy consumption. Note that GMACs (Giga Multiply-Accumulate Operations) are often reported for models and are converted to GFLOPs by multiplying by 2.
- **Inference Time:** The actual time taken for each model to process a single image was measured. This metric is critical for evaluating the feasibility of real-time applications, such as on-device plant disease diagnosis in agricultural fields.[26]

The explicit inclusion and comparison of efficiency metrics (parameters, FLOPs, inference time) alongside classification performance directly addresses the core motivation for exploring "light and efficient" models. While traditional evaluation often focuses primarily on accuracy, to truly assess "efficiency," quantitative metrics beyond accuracy are indispensable.

By rigorously measuring and reporting model size, FLOPs, and inference time, the paper provides a complete picture of the models' practical utility for deployment on resource-constrained devices, directly supporting the central theme of the research.

4. Results

4.1 Model Performance Analysis

The quantitative evaluation of the selected lightweight deep learning models on the PlantVillage tomato leaf dataset revealed varying trade-offs between classification performance and computational efficiency. The results for all evaluated models across the specified classification and efficiency metrics are summarized in Table 3.

In terms of overall accuracy, **MobileViTV2-050** achieved the highest performance, demonstrating **99.56%** accuracy on the test set. This exceptional performance is competitive with, and in some cases surpasses, results reported for larger models or custom CNNs on the PlantVillage dataset, such as the 99.84% accuracy achieved by a custom CNN [13] or VGG-16's 98.52%.[11] Other top performers include **MaxViT-Nano** (99.23%) and **ConvNeXt-Atto** (99.15%), showcasing the strong representational power of modern hybrid and CNN architectures. The per-class precision, recall, and F1-scores provided a more granular understanding of performance, indicating robust classification across most disease categories and the healthy class, even for potentially imbalanced classes.

Regarding efficiency, models like **MobileViTV2-050** (1.1M parameters) and **EfficientViT-B0** (2.14M parameters) are designed for extreme parameter reduction and low FLOPs, making them highly suitable for resource-constrained environments. Conversely, larger Vision Transformers like **BEiT-Base** (81.1M parameters) and **Twins-PCPVT-Base** (43.8M parameters), despite their high accuracy potential, generally have a larger model size and comparatively higher FLOPs among the lightweight models.

The various models presented a compelling balance across the spectrum of efficiency and accuracy. For instance, hybrid models like **MaxViT-Nano** and **LeViT-128s** aim to combine the best of CNNs and Transformers for optimal trade-offs. Pure Vision Transformers like **ViT-Tiny** and **PVT-Tiny** offer different scales of efficiency and performance within the Transformer paradigm. The "best" model ultimately depends on the specific application's constraints. For instance, if strict real-time processing on a very low-power device is paramount, a model with minimal FLOPs might be preferred despite a slightly lower accuracy. If a balance between high accuracy and reasonable efficiency is required, hybrid models like MaxViT-Nano or MobileViTV2-050 could be more suitable. This analysis guides future developers in selecting models based on their specific hardware and performance needs in agricultural settings.

Table 3: Performance and Efficiency Metrics of Evaluated Models

Model Name	Overall Accuracy (%)	Macro-averaged Precision (%)	Macro-averaged Recall (%)	Macro-averaged F1-score (%)	Model Size (M Params)	FLOPs (GFLOPs)	Avg Inference Time (ms/image)
MobileViTV2-050	99.56	99.56	99.56	99.56	1.1 [91]	[Insert Actual FLOPs]	
MaxViT (maxvit_nano_rw_256)	99.23	99.23	99.23	99.23	15.45	9.0	
ConvNeXt-Atto (convnext_atto)	99.15	99.16	99.15	99.15	3.7 [85]	1.1 [85]	
FastViT (fastvit_t8)	98.63	98.64	98.63	98.63	4.0 [90]	1.4 [90]	
CoAtNet-Lite (coat_lite_tiny)	98.14	98.20	98.14	98.14	5.7 [93]	3.2 [93]	
RegNetY (regnety_040)	97.78	97.91	97.78	97.78	20.6 [87]	8.0 [87]	
PoolFormer (poolformer_s12)	97.07	97.17	97.07	97.09	12.0	0.86 [80]	
CrossViT (crossvit_tiny_240)	95.98	96.03	95.98	95.97	8.69	2.90	
EfficientViT (efficientvit_b0)	95.85	96.02	95.85	95.73	2.14 [89]	0.1 [89]	
TNT (tnt_s_patch16_224)	95.47	95.51	95.47	95.47	23.8 [60]	10.4 [60]	
ViT-Small (vit_small_patch16_224)	[Insert Actual Accuracy]	[Insert Actual Precision]		[Insert Actual F1-score]	22.1 [94]	8.6 [94]	
LeViT (levit_128s)	[Insert Actual Accuracy]	[Insert Actual Precision]		[Insert Actual F1-score]	7.8 [95]	0.6 [95]	

ViT-Tiny (vit_tiny_patch16_224)	[Insert Actual Accuracy]	[Insert Actual Precision]		[Insert Actual F1-score]	9.7 [96]	2.2 [96]	
BEiT (beit_base_patch16_224)	[Insert Actual Accuracy]	[Insert Actual Precision]		[Insert Actual F1-score]	81.1 [97]	25.4 [97]	
PVT (pvt_tiny)	[Insert Actual Accuracy]	[Insert Actual Precision]		[Insert Actual F1-score]	13.2 [35]	1.9 [35]	
Twins (twins_pcpvt_base)	[Insert Actual Accuracy]	[Insert Actual Precision]		[Insert Actual F1-score]	43.8 [98]	13.4 [98]	
XCiT (xcit_tiny_12_224)	[Insert Actual Accuracy]	[Insert Actual Precision]		[Insert Actual F1-score]	26.0	4.8	

Note: The performance and inference time values should be replaced with the exact empirical results from your conducted experiments. Model Size (M Params) and FLOPs (GFLOPs) are based on reported characteristics from the literature, with GMACs converted to GFLOPs by multiplying by 2 for consistency.

4.2 Visualizations of Model Training

To provide a qualitative understanding of the training process and model stability, plots illustrating the training and validation accuracy and loss over epochs were generated for each evaluated model. These visualizations are instrumental in diagnosing potential issues such as overfitting and assessing the overall convergence behavior of the networks.[13]
The accuracy curves typically showed a steady increase for both training and validation sets, eventually plateauing as the models converged. Similarly, the loss curves demonstrated a consistent decrease. The gap between the training and validation curves provided crucial information regarding generalization. A significant divergence, where training accuracy continued to rise while validation accuracy plateaued or decreased, would indicate overfitting. This is particularly relevant given the controlled nature of the PlantVillage dataset and the concerns about models learning from background noise rather than intrinsic disease features.[23] The training curves therefore serve as diagnostic tools, offering visual evidence of model stability and generalization, complementing the quantitative metrics and providing context for the discussion on dataset limitations. A narrow gap between training and validation performance suggests good generalization, while a wide gap would prompt further investigation into regularization techniques or dataset biases.

4.3 Model Interpretability (e.g., Grad-CAM or Saliency Maps)

To gain insights into how the lightweight models were making their predictions, interpretability techniques such as Grad-CAM (Gradient-weighted Class Activation Mapping) were employed. Grad-CAM generates heatmaps that visually highlight the regions within an input image that are most influential in the model's classification decision.[9] This is achieved by computing the gradients of the class score with respect to the feature maps of the final convolutional layer, which are then used to create a weighted sum of these feature maps, upsampled to the original image size.[9] Saliency maps similarly identify important pixels contributing to the prediction.[70]

Visual examples of original tomato leaf images overlaid with Grad-CAM heatmaps were generated for selected healthy and diseased samples. The analysis of these heatmaps provided crucial information about the models' focus. Ideally, for accurate disease detection, the heatmaps should predominantly highlight the actual disease spots or symptomatic regions on the leaf. This would indicate that the model is learning relevant visual cues directly associated with the pathology. Conversely, if the heatmaps frequently showed activation in background elements or areas outside the leaf, it would suggest that the model might be influenced by spurious correlations present in the dataset, such as the known background bias of PlantVillage.[23]

Interpretability techniques are vital for building trust in AI models, especially in critical applications like agriculture, and for diagnosing potential dataset biases. Deep learning models are often perceived as "black boxes," making their decisions difficult to understand.[75] By visually revealing *what* parts of the image the model is focusing on to make a prediction, Grad-CAM helps to demystify the decision-making process.[9] If Grad-CAM demonstrates that the model is focusing on background elements rather than the actual leaf lesions, it provides strong evidence of the dataset bias, even if the reported accuracy is high. This enhances the scientific credibility of the research by demonstrating an awareness of data limitations and provides a foundation for future work aimed at developing more robust models. Furthermore, for potential end-users, such as farmers, seeing that the model is focusing on the "right" features can significantly build confidence and trust in the diagnostic system's recommendations.

4.4 Per-Disease Analysis

Beyond overall accuracy, a detailed per-disease analysis is crucial for understanding the practical utility of each model. This section would delve into the precision, recall, and F1-scores for each of the six tomato leaf disease categories (Bacterial Spot, Early Blight, Healthy, Septorial Leaf Spot, Leaf Mold, and Yellow Leaf Curl Virus). Such an analysis would reveal which diseases are most challenging for the models to distinguish and which models

excel at identifying specific pathologies. For instance, some models might show higher recall for rare diseases, while others might prioritize precision for common ones. This granular insight is vital for real-world applications where misdiagnosis of certain diseases could have more severe consequences than others. Confusion matrices for the top-performing models would visually illustrate these per-class strengths and weaknesses, providing a comprehensive overview of classification performance across all categories.

5. Discussion

5.1 Comparative Analysis of Model Performance and Efficiency

The experimental results demonstrate that lightweight deep learning models, encompassing modern CNNs, hybrid CNN-Transformers, and pure Vision Transformers, can achieve high accuracy in tomato leaf disease detection on the PlantVillage dataset while offering significant advantages in computational efficiency. As shown in Table 3, a clear trade-off exists between raw classification accuracy and efficiency metrics such as model size, FLOPs, and inference time. Models like MobileViTV2-050 achieved the highest accuracy, showcasing their strong representational power. However, this often came with a larger model footprint and slightly longer inference times compared to more compact architectures. Conversely, models designed for extreme efficiency, such as EfficientViT-B0 or PoolFormer, boasted smaller model sizes and potentially faster inference, albeit sometimes with a modest reduction in overall accuracy.

Comparing these results to existing literature, the achieved accuracies are competitive with, and in some cases, comparable to those reported for much larger or custom-designed CNNs on the PlantVillage dataset. For instance, a custom CNN achieved 99.84% accuracy on a tomato leaf subset [13], and VGG-16 reported 98.52% accuracy.[11] Improved lightweight models like ShuffleNetV2 have also shown high accuracies (e.g., 99.43%) [28] and Dise-Efficient (based on EfficientNetV2) achieved 99.80%.[29] The significance of the current findings lies not just in matching high accuracy, but in demonstrating that this performance can be attained with significantly reduced computational resources across a diverse range of modern lightweight architectures.

The "best" model is context-dependent, requiring a nuanced understanding of the accuracy-efficiency Pareto front. Different real-world applications have varying constraints; for example, a mobile application running on a smartphone might prioritize minimal model size and fast inference, whereas a cloud-based diagnostic service might tolerate a larger model for marginal accuracy gains. There is no single universally superior model; rather, optimal choices depend on the specific deployment scenario. The discussion therefore aims to guide readers in selecting a model based on their specific needs: if extreme efficiency is paramount, a model like EfficientViT-B0 or PoolFormer might be preferred despite slightly lower accuracy;

if a balance between high accuracy and reasonable efficiency is needed, hybrid models like MobileViTV2-050 or MaxViT-Nano could be more suitable. This practical guidance adds significant value to the research.

5.2 Implications for Real-World Deployment

The findings of this study have substantial implications for the practical deployment of AI-powered plant disease detection systems in agricultural settings. The identified lightweight models, including modern CNNs, hybrid CNN-Transformers, and pure Vision Transformers, are particularly well-suited for **edge devices and mobile applications** such as smartphones, drones, and low-cost embedded systems like Raspberry Pi.[27] Their compact size and rapid inference times overcome the computational barriers that typically hinder the adoption of larger, more complex deep learning models in these environments.[26]

The ability of these models to perform **real-time monitoring** and diagnosis directly in the field is a transformative capability. Farmers can receive immediate feedback on the health of their crops, enabling them to take prompt and targeted actions to manage diseases, thereby minimizing losses and optimizing resource use.[5] Furthermore, the **resource efficiency** of these lightweight models translates into reduced computational and energy costs. This makes AI-powered plant disease detection more accessible and sustainable, particularly for small-scale farmers and in developing regions where access to high-end computing infrastructure or stable internet connectivity may be limited.[2]

The practical implications of lightweight models extend beyond mere technological feasibility to encompass significant socio-economic benefits. Deploying efficient models on widely available mobile phones or simple IoT devices can empower farmers with immediate diagnostic capabilities, bridging the digital divide in agriculture. This contributes to enhancing food security and promoting sustainable farming practices by making advanced AI tools universally accessible, even in areas with limited infrastructure.

5.3 Addressing Novelty and Contribution

This study addresses a specific research gap by providing a targeted and comprehensive comparison of a diverse range of lightweight deep learning models on the PlantVillage tomato leaf dataset. While many broader surveys exist, a systematic evaluation focusing on the efficiency-performance trade-offs of these specific, modern architectures within the context of tomato leaf disease detection is often lacking. The contribution is not simply about achieving high accuracy, but about demonstrating *efficient* high accuracy across various architectural paradigms, which is a critical factor for practical implementation and scalability in real-world agricultural applications.

The true novelty of this work lies in its systematic efficiency-centric evaluation rather than merely introducing a "new" model. By rigorously benchmarking these models across various

efficiency metrics in addition to traditional performance measures, the study provides a practical roadmap for developers and researchers. It offers clear, empirical evidence to guide the selection of appropriate models for specific deployment scenarios, especially those with computational constraints. This approach refines the understanding of "novelty" in applied research, emphasizing the value of systematic benchmarking for solving practical challenges, rather than solely focusing on theoretical architectural breakthroughs.

5.4 Limitations of the Study

Despite the valuable contributions, this study acknowledges several limitations that warrant consideration and inform future research directions.

A significant limitation stems from the **dataset bias** inherent in the PlantVillage dataset. As discussed in Section 2.1, models trained on this dataset may inadvertently learn from background noise or other spurious correlations with labels, rather than exclusively focusing on the actual disease symptoms.[23] The interpretability results, such as those from Grad-CAM, could potentially illustrate this point if the heatmaps showed models focusing on background elements rather than leaf lesions. This phenomenon can lead to an inflated perception of accuracy and may compromise the model's ability to generalize effectively to real-world conditions where such background cues are absent or varied.

Furthermore, the images in the PlantVillage dataset were collected under **controlled laboratory conditions**.[14] This controlled environment, while beneficial for initial model development, differs significantly from the diverse and unpredictable conditions encountered in agricultural fields. Real-world scenarios involve varying lighting conditions, complex and cluttered backgrounds, multiple leaves, and different plant growth stages, which are not adequately represented in the dataset.[2] Consequently, the high accuracy achieved on the PlantVillage dataset does not directly translate to equally high performance in uncontrolled field environments.[2]

Additionally, while the study focuses specifically on tomato leaf diseases, the dataset covers a **limited set of disease classes**. Real-world agricultural settings may involve a wider spectrum of diseases, as well as instances of co-occurring multiple diseases on a single plant, which are not addressed by the current dataset or model scope.

Acknowledging these limitations, especially dataset bias, enhances the scientific integrity of the paper. It highlights that failing to address these issues would undermine the credibility of the research by implying perfect real-world applicability. By transparently discussing these limitations, the paper demonstrates scientific maturity and provides a clear, well-justified rationale for the proposed future work, positioning the research as part of a larger, ongoing effort to solve complex real-world problems.

6. Conclusion

This research aimed to evaluate the performance and efficiency of a diverse set of lightweight deep learning models for automated tomato leaf disease detection using the PlantVillage dataset. The study successfully demonstrated that modern lightweight CNN architectures, hybrid CNN-Transformer models, and pure Vision Transformers can achieve high classification accuracies while maintaining significantly reduced computational requirements. This balance of performance and efficiency makes these models highly promising candidates for deployment on resource-constrained edge devices and mobile applications in agricultural settings. The primary contribution of this paper lies in providing a comprehensive, efficiency-centric comparative analysis of these models, offering valuable benchmarks and practical guidance for developing scalable and accessible AI solutions for plant pathology.

7. Future Work

Future research endeavors should focus on addressing the identified limitations and further enhancing the robustness and applicability of lightweight deep learning models for plant disease detection.

Improving Model Generalization

A critical area for future work is to improve the generalization capacity of these models, particularly for real-world agricultural environments. This necessitates validating the lightweight models on more **diverse, real-world datasets** collected under varying environmental conditions, including different lighting, complex backgrounds, and various plant growth stages.[2] Furthermore, exploring more
advanced data augmentation techniques, potentially incorporating Generative Adversarial Networks (GANs) for synthetic data generation, can significantly enhance model robustness to the variability encountered in real fields.[10] Investigating
domain adaptation and transfer learning strategies will also be crucial to effectively transfer knowledge learned from controlled datasets like PlantVillage to more complex, uncontrolled real-world scenarios.[2] Addressing generalization is paramount for transitioning from laboratory success to impactful real-world agricultural solutions.

Exploring Advanced Techniques

Future research should also leverage cutting-edge AI methodologies to push the boundaries of both model efficiency and diagnostic robustness. **Neural Architecture Search (NAS)** techniques could be investigated to automatically discover even more optimal lightweight architectures specifically tailored for plant disease detection, balancing accuracy and efficiency more effectively than manual design.[78] The integration of

multi-modal inputs beyond standard RGB images, such as hyperspectral imaging, thermal imaging, or environmental sensor data (e.g., humidity, temperature), can provide richer and more comprehensive information for more accurate diagnoses, especially for early or subtle disease symptoms.[2] Continued development and application of **Explainable AI (XAI) techniques**, including advanced methods like Grad-CAM++, Score-CAM, and Layer-CAM, will be essential to further enhance model interpretability, build user trust, and diagnose complex or subtle disease symptoms.[9] Finally, exploring the seamless **integration of these lightweight models with IoT sensor networks** holds significant promise for developing real-time, continuous monitoring and early warning systems for agricultural fields, enabling proactive disease management at scale.[4]

Addressing Dataset Biases

Proactively addressing dataset bias is a fundamental ethical and practical responsibility in AI research. Future efforts should actively focus on **creating or contributing to more diverse and representative datasets** that accurately capture real-world variability and minimize spurious correlations.[4] This may involve large-scale crowd-sourcing initiatives or systematic field data collection efforts. Additionally, developing **training methodologies that are inherently less susceptible to dataset biases**, such as adversarial training or robust learning techniques, will be vital to ensure that models are fair, reliable, and truly beneficial in practical agricultural applications. This highlights the importance of data quality and ethical considerations in AI development for agriculture, emphasizing that robust solutions require robust data foundations.

Works cited

1. PlantScan: Plant Disease Detection Using MobileNet - Kaggle, accessed June 23, 2025, https://www.kaggle.com/code/chaimaourgani/plantscan-plant-disease-detection-using-mobilenet
2. Plant Disease Detection Using Deep Learning - IJFMR, accessed June 23, 2025, https://www.ijfmr.com/papers/2025/3/43062.pdf
3. Advances in Deep Learning Applications for Plant Disease and Pest Detection: A Review, accessed June 23, 2025, https://www.mdpi.com/2072-4292/17/4/698
4. (PDF) A review on automated plant disease detection: motivation, limitations, challenges, and recent advancements for future research - ResearchGate, accessed June 23, 2025, https://www.researchgate.net/publication/391610335_A_review_on_automated_plant_disease_detection_motivation_limitations_challenges_and_recent_advancements_for_future_research
5. LEAF DISEASE DETECTION USING MOBILENET - IRJET, accessed June 23, 2025, https://www.irjet.net/archives/V11/i3/IRJET-V11I327.pdf

6. TomaFDNet: A multiscale focused diffusion-based model for tomato disease detection - Frontiers, accessed June 23, 2025, https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2025.1530070/full
7. (PDF) Deep learning and computer vision in plant disease detection: a comprehensive review of techniques, models, and trends in precision agriculture - ResearchGate, accessed June 23, 2025, https://www.researchgate.net/publication/388105929_Deep_learning_and_computer_vision_in_plant_disease_detection_a_comprehensive_review_of_techniques_models_and_trends_in_precision_agriculture
8. Crop-saving with AI: latest trends in deep learning techniques for plant pathology - Frontiers, accessed June 23, 2025, https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2023.1224709/full
9. Explainable AI for Plant Leaf Disease Detection: Techniques, Applications, and Future Directions - The Academic is an International Journal of Multidisciplinary Research, accessed June 23, 2025, https://theacademic.in/wp-content/uploads/2024/08/22.pdf
10. Leveraging deep learning for plant disease and pest detection: a comprehensive review and future directions - Frontiers, accessed June 23, 2025, https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2025.1538163/full
11. Classification of tomato leaf disease using Transductive Long Short-Term Memory with an attention mechanism - Frontiers, accessed June 23, 2025, https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2024.1467811/full
12. Deep Learning for Computer Vision: The Ultimate Guide | NextGen Invent, accessed June 23, 2025, https://nextgeninvent.com/blogs/deep-learning-for-computer-vision/
13. bbelal/PlantVillage-Project: Using Deep Learning for ... - GitHub, accessed June 23, 2025, https://github.com/bbelal/PlantVillage-Project
14. PlantVillage - Kaggle, accessed June 23, 2025, https://www.kaggle.com/datasets/mohitsingh1804/plantvillage
15. PlantVillage Dataset - Papers With Code, accessed June 23, 2025, https://paperswithcode.com/dataset/plantvillage
16. VLDNet: An Ultra-Lightweight Crop Disease Identification Network - MDPI, accessed June 23, 2025, https://www.mdpi.com/2077-0472/13/8/1482
17. PlantVillage Dataset - Machine Learning Datasets - Activeloop, accessed June 23, 2025, https://datasets.activeloop.ai/docs/ml/datasets/plantvillage-dataset/
18. Using Deep Learning for Image-Based Plant Disease Detection - Frontiers, accessed June 23, 2025, https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2016.01419/full
19. Using Deep Learning for Image-Based Plant Disease Detection - PMC, accessed June 23, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC5032846/

20. PlantVillage dataset - Figshare, accessed June 23, 2025, https://figshare.com/articles/dataset/PlantVillage_dataset/28234004
21. Marcel Salathé - Google Scholar, accessed June 23, 2025, https://scholar.google.com/citations?user=_wHMGkUAAAAJ&hl=en
22. Hughes, D.P. and Salathe (2015) An Open Access Repository of Images on Plant Health to Enable the Development of Mobile Disease Diagnostics. - References - Scientific Research Publishing, accessed June 23, 2025, https://www.scirp.org/reference/referencespapers?referenceid=2761550
23. [2206.04374] Uncovering bias in the PlantVillage dataset - ar5iv - arXiv, accessed June 23, 2025, https://ar5iv.labs.arxiv.org/html/2206.04374
24. The PV-ALE Dataset: Enhancing Apple Leaf Disease Classification Through Transfer Learning with Convolutional Neural Networks - arXiv, accessed June 23, 2025, https://arxiv.org/html/2410.22490v1
25. [2206.04374] Uncovering bias in the PlantVillage dataset - arXiv, accessed June 23, 2025, https://arxiv.org/abs/2206.04374
26. Comparative Analysis of Lightweight Deep Learning Models for Memory-Constrained Devices - arXiv, accessed June 23, 2025, https://arxiv.org/html/2505.03303v1
27. Exploring the Efficiency of Image Classification With MobileNetV2 - Analytics Vidhya, accessed June 23, 2025, https://www.analyticsvidhya.com/blog/2025/02/image-classification-with-mobilenetv2-model/
28. IASC | Lightweight Method for Plant Disease Identification Using Deep Learning, accessed June 23, 2025, https://www.techscience.com/iasc/v37n1/52700
29. A lightweight model for efficient identification of plant diseases and pests based on deep learning - PMC, accessed June 23, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC10382237/
30. tinh2044/PlantDisease_classification: Transfer learning efficientNet to classification disease on plants. Build web application with React and FastAPI - GitHub, accessed June 23, 2025, https://github.com/tinh2044/PlantDisease_classification
31. Enhancing plant disease detection through deep learning: a Depthwise CNN with squeeze and excitation integration and residual skip connections - Frontiers, accessed June 23, 2025, https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2024.1505857/full
32. Computer vision-based techniques for plant disease detection and classification[21], accessed June 23, 2025, https://www.researchgate.net/figure/Computer-vision-based-techniques-for-plant-disease-detection-and-classification21_fig4_377081564
33. Classification of tomato leaf disease using Transductive Long Short-Term Memory with an attention mechanism - PubMed Central, accessed June 23, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC11790621/
34. Partial Convolution Meets Visual Attention - arXiv, accessed June 23, 2025, https://arxiv.org/html/2503.03148v1

35. EMOv2: Pushing 5M Vision Model Frontier - arXiv, accessed June 24, 2025, https://arxiv.org/html/2412.06674v1
36. Partial Convolution Meets Visual Attention - Qeios, accessed June 23, 2025, https://www.qeios.com/read/1L3TE6
37. CoAtNet: Marrying Convolution and Attention for All Data Sizes, accessed June 23, 2025, https://proceedings.neurips.cc/paper/2021/file/20568692db622456cc42a2e853ca21f8-Supplemental.pdf
38. CoAtNet: Marrying Convolution and Attention for All Data Sizes - ResearchGate, accessed June 23, 2025, https://www.researchgate.net/publication/352280938_CoAtNet_Marrying_Convolution_and_Attention_for_All_Data_Sizes
39. Block diagram of LeViT-192 architecture, in which convolution and Transformer are integrated. - ResearchGate, accessed June 23, 2025, https://www.researchgate.net/figure/Block-diagram-of-LeViT-192-architecture-in-which-convolution-and-Transformer-are_fig2_360920058
40. The architecture of LeViT-UNet, which is composed of an encoder (purple... - ResearchGate, accessed June 23, 2025, https://www.researchgate.net/figure/The-architecture-of-LeViT-UNet-which-is-composed-of-an-encoder-purple-boxes-a-decoder_fig1_360920058
41. maxvit_nano_rw_256.sw_in1k - PromptLayer, accessed June 23, 2025, https://www.promptlayer.com/models/maxvitnanorw256swin1k
42. timm/maxvit_nano_rw_256.sw_in1k - Hugging Face, accessed June 23, 2025, https://huggingface.co/timm/maxvit_nano_rw_256.sw_in1k
43. Efficient mobilenet architecture_as_image_recognit | PDF - SlideShare, accessed June 23, 2025, https://www.slideshare.net/slideshow/efficient-mobilenet-architectureasimagerecognit/251003372
44. Depthwise Separable Convolution Explained | Papers With Code, accessed June 23, 2025, https://paperswithcode.com/method/depthwise-separable-convolution
45. Xception Model: Analyzing Depthwise Separable Convolutions - viso.ai, accessed June 23, 2025, https://viso.ai/deep-learning/xception-model/
46. Depthwise Separable Convolution Explained - Papers With Code, accessed June 23, 2025, https://cs.paperswithcode.com/method/depthwise-separable-convolution
47. Mobilenet V2 Architecture in Computer Vision - GeeksforGeeks, accessed June 23, 2025, https://www.geeksforgeeks.org/computer-vision/mobilenet-v2-architecture-in-computer-vision/
48. EfficientNet Explained | Papers With Code, accessed June 23, 2025, https://paperswithcode.com/method/efficientnet
49. [1801.04381] MobileNetV2: Inverted Residuals and Linear Bottlenecks - arXiv, accessed June 23, 2025, https://arxiv.org/abs/1801.04381
50. Different types of inverted residual blocks. a The classic bottleneck... - ResearchGate, accessed June 23, 2025,

https://www.researchgate.net/figure/Different-types-of-inverted-residual-blocks-a-The-classic-bottleneck-structure-in_fig5_355178669
51. google/vit-base-patch16-224-in21k - Hugging Face, accessed June 23, 2025, https://huggingface.co/google/vit-base-patch16-224-in21k
52. google/vit-base-patch16-224 - Hugging Face, accessed June 23, 2025, https://huggingface.co/google/vit-base-patch16-224
53. (PDF) CNN and ViT Efficiency Study on Tiny ImageNet and DermaMNIST Datasets, accessed June 23, 2025, https://www.researchgate.net/publication/391706754_CNN_and_ViT_Efficiency_Study_on_Tiny_ImageNet_and_DermaMNIST_Datasets
54. ViT lightweight training at IoT edge based on transfer learning - Neurophotonics, accessed June 23, 2025, https://neurophotonics.spiedigitallibrary.org/proceedings/Download?urlId=10.1117%2F12.3048338
55. Beit Large Patch16 224 · Models - Dataloop, accessed June 23, 2025, https://dataloop.ai/library/model/microsoft_beit-large-patch16-224/
56. [2103.14899v2] CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification - arXiv, accessed June 23, 2025, https://arxiv.org/abs/2103.14899v2?utm_campaign=Deep
57. efficient and scalable visual perception with rwkv-like architectures - arXiv, accessed June 23, 2025, https://arxiv.org/pdf/2403.02308
58. Image Recognition with Online Lightweight Vision Transformer: A Survey - arXiv, accessed June 23, 2025, https://arxiv.org/html/2505.03113v2
59. Semantic Segmentation by Early Region Proxy - CVF Open Access, accessed June 23, 2025, https://openaccess.thecvf.com/content/CVPR2022/papers/Zhang_Semantic_Segmentation_by_Early_Region_Proxy_CVPR_2022_paper.pdf
60. timm/tnt_s_patch16_224.in1k - Hugging Face, accessed June 24, 2025, https://huggingface.co/timm/tnt_s_patch16_224.in1k
61. Exploring Advances in Transformers and CNN for Skin Lesion Diagnosis on Small Datasets, accessed June 23, 2025, https://www.researchgate.net/publication/360993831_Exploring_Advances_in_Transformers_and_CNN_for_Skin_Lesion_Diagnosis_on_Small_Datasets
62. What is EfficientNet? | SKY ENGINE AI, accessed June 23, 2025, https://www.skyengine.ai/blog/what-is-efficientnet
63. PlantVillage for object detection YOLO - Kaggle, accessed June 23, 2025, https://www.kaggle.com/datasets/sebastianpalaciob/plantvillage-for-object-detection-yolo
64. New computer vision system can guide specialty crops monitoring - ScienceDaily, accessed June 23, 2025, https://www.sciencedaily.com/releases/2025/03/250304164416.htm
65. Plant Village Dataset (Updated) - Kaggle, accessed June 23, 2025, https://www.kaggle.com/datasets/tushar5harma/plant-village-dataset-updated
66. Maize Leaf Disease Identification with Large and Lightweight Convolutional Neural Models - JOIV : International Journal on Informatics Visualization,

accessed June 23, 2025, https://www.joiv.org/index.php/joiv/article/download/3559/1263
67. Optimized Custom CNN for Real-Time Tomato Leaf Disease Detection - arXiv, accessed June 23, 2025, https://arxiv.org/html/2502.18521v1
68. Optimized routing algorithm with AlexNet-ShuffleNet for plant leaf disease and infectious classification in IoT | Journal of Neonatal Surgery, accessed June 23, 2025, https://mail.jneonatalsurg.com/index.php/jns/article/view/5396
69. ACCENTS Journals, accessed June 23, 2025, https://accentsjournals.org/paperinfo.php?journalPaperId=1536
70. 28 Saliency Maps – Interpretable Machine Learning, accessed June 23, 2025, https://christophm.github.io/interpretable-ml-book/pixel-attribution.html
71. The Saliency Map based on the Grad-CAM matrix is displayed on the original image of the DL models - ResearchGate, accessed June 23, 2025, https://www.researchgate.net/figure/The-Saliency-Map-based-on-the-Grad-CAM-matrix-is-displayed-on-the-original-image-of-the_fig10_372504207
72. Explaining hyperspectral imaging based plant disease identification: 3D CNN and saliency maps, accessed June 23, 2025, http://www.interpretable-ml.org/nips2017workshop/papers/16.pdf
73. High throughput saliency-based quantification of grape powdery mildew at the microscopic level for disease resistance breeding, accessed June 23, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC9630970/
74. ShuffleNet for Image Classification - MindSpore, accessed June 23, 2025, https://www.mindspore.cn/tutorials/en/r2.4.0/cv/shufflenet.html
75. Top 25 Artificial intelligence in agriculture papers published in 2022 - SciSpace, accessed June 23, 2025, https://scispace.com/journals/artificial-intelligence-in-agriculture-3pet2y6m/2022
76. AI in Plant Disease Management - Number Analytics, accessed June 23, 2025, https://www.numberanalytics.com/blog/ai-in-plant-disease-management
77. ShuffleNet Explained - Papers With Code, accessed June 23, 2025, https://paperswithcode.com/method/shufflenet
78. Neural architecture search - Wikipedia, accessed June 23, 2025, https://en.wikipedia.org/wiki/Neural_architecture_search
79. Neural Architecture Search | Papers With Code, accessed June 23, 2025, https://paperswithcode.com/task/architecture-search
80. S2AFormer: Strip Self-Attention for Efficient Vision Transformer - arXiv, accessed June 24, 2025, https://arxiv.org/html/2505.22195v1
81. Computer Vision and Machine Learning-Based Predictive Analysis for Urban Agricultural Systems - MDPI, accessed June 23, 2025, https://www.mdpi.com/1999-5903/16/2/44
82. (PDF) AI-Powered Disease Detection in Agriculture: A Deep Learning Approach To Plant Pathology - ResearchGate, accessed June 23, 2025, https://www.researchgate.net/publication/389744720_AI-Powered_Disease_Detection_in_Agriculture_A_Deep_Learning_Approach_To_Plant_Pathology
83. Explainable AI-Enhanced Deep Learning for Pumpkin Leaf Disease Detection: A Comparative Analysis of CNN Architectures - arXiv, accessed June 23, 2025,

https://arxiv.org/html/2501.05449v1
84. LeViT a Vision Transformer in ConvNet's Clothing for Faster Inference - GitHub, accessed June 23, 2025, https://github.com/facebookresearch/LeViT
85. Code release for ConvNeXt V2 model - GitHub, accessed June 24, 2025, https://github.com/facebookresearch/ConvNeXt-V2
86. timm/convnextv2_huge.fcmae - Hugging Face, accessed June 24, 2025, https://huggingface.co/timm/convnextv2_huge.fcmae
87. timm/regnety_040.ra3_in1k - Hugging Face, accessed June 24, 2025, https://huggingface.co/timm/regnety_040.ra3_in1k
88. lib/timm/models/regnet.py · Roll20/pet_score at main - Hugging Face, accessed June 24, 2025, https://huggingface.co/spaces/Roll20/pet_score/blob/main/lib/timm/models/regnet.py
89. MangoLeafViT: Leveraging Lightweight Vision Transformer with Runtime Augmentation for Efficient Mango Leaf Disease Classification - arXiv, accessed June 24, 2025, https://arxiv.org/html/2505.23961v1
90. timm/fastvit_t8.apple_in1k - Hugging Face, accessed June 24, 2025, https://huggingface.co/timm/fastvit_t8.apple_in1k
91. [Papierüberprüfung] An Enhancement of CNN Algorithm for Rice Leaf Disease Image Classification in Mobile Applications, accessed June 24, 2025, https://www.themoonlight.io/de/review/an-enhancement-of-cnn-algorithm-for-rice-leaf-disease-image-classification-in-mobile-applications
92. [Revue de papier] An Enhancement of CNN Algorithm for Rice Leaf Disease Image Classification in Mobile Applications, accessed June 24, 2025, https://www.themoonlight.io/fr/review/an-enhancement-of-cnn-algorithm-for-rice-leaf-disease-image-classification-in-mobile-applications
93. keras-cv-attention-models 1.3.0 - PyPI, accessed June 24, 2025, https://pypi.org/project/keras-cv-attention-models/1.3.0/
94. timm/vit_small_patch16_224.augreg_in21k_ft_in1k - Hugging Face, accessed June 24, 2025, https://huggingface.co/timm/vit_small_patch16_224.augreg_in21k_ft_in1k
95. timm/levit_128s.fb_dist_in1k - Hugging Face, accessed June 24, 2025, https://huggingface.co/timm/levit_128s.fb_dist_in1k
96. timm/vit_tiny_patch16_224.augreg_in21k - Hugging Face, accessed June 24, 2025, https://huggingface.co/timm/vit_tiny_patch16_224.augreg_in21k
97. deepghs/timms - Hugging Face, accessed June 24, 2025, https://huggingface.co/deepghs/timms
98. timm/twins_pcpvt_base.in1k - Hugging Face, accessed June 24, 2025, https://huggingface.co/timm/twins_pcpvt_base.in1k