Chapter 4

CHAPTER 4

PROPOSED MODEL: A POSITIONAL-AWARE DUAL-ATTENTION AND TOPOLOGY-FUSION WITH GENERATIVE ADVERSARIAL NETWORK-BASED HIGH-RESOLUTION FRAMEWORK FOR LEAF DISEASE IMAGE CLASSIFICATION

4.1 INTRODUCTION

This chapter presents a quick summary of the difficulties in classifying leaf disease images. It also discusses the role of leaf disease image restoration and classification techniques. Using the Plant Village Dataset (PVD), The state-of-the-art models are compared and contrasted, including DATFGAN using ShuffleNetV2, DenseNet121, and MobileNetV2 classifiers, against the proposed Positional-aware Dual-Attention and Topology-Fusion with Generative Adversarial Network (PDATFGAN).

4.2 OVERVIEW OF LEAF DISEASE IMAGE CLASSIFICATION

Nowadays, crop management put importance on leaf disease classification because of more frequency of different kinds of leaf diseases. Leaf disease detection is more difficult task because of diagnosis impacts and discrepancies in atmospheric conditions. The classification of many leaf diseases, which affect several plant species is problematic and manual detection is not appropriate for proper diagnosis. Automated leaf disease image classification has been developed to solve this problem. It's a subfield of plant pathology where computer vision is used to develop methods for the purpose of automatically classifying images of plant leaf diseases (Kaur et al., 2019). It intends to computerize the classification procedure by training different algorithms on huge datasets of labelled images, allowing them to capture different visual characteristics related to the leaf disease symptoms. It is essential for the agriculture field since cultivators may apply it to make precise decisions on preventing leaf diseases that enhances crop yield quality and productivity.

On the other hand, poor quality leaf images can pose significant challenges to leaf disease classification models (Wen et al. 2020). These challenges arise from the fact that

low-quality images might lack essential visual features and information needed for accurate disease detection. Some specific problems that can occur in leaf disease classification due to poor quality leaf images are the following:

- Lack of clear features: Image quality issues such as low resolution, blurriness, and noise might hinder the model's ability to differentiate between healthy and sick leaves. Clear and distinct visual features are crucial for accurate classification (Dhaka et al. 2021)
- Ambiguous visual cues: Some diseases might manifest as subtle colour changes or minor textural differences on leaves. Poor quality images can obscure these cues, leading to misclassifications or difficulties in distinguishing between classes
- Limited textural information: Texture plays a crucial role in leaf disease identification. Low-quality images might lack the necessary texture details, making it challenging for the model to differentiate between different disease types (Singh et al. 2020)
- Colour distortions: Inaccurate colour reproduction, lighting variations, or colour shifts in images can misrepresent the true colour of leaves. Since colour is a vital visual feature, these distortions can lead to incorrect disease classification
- Occlusions and background noise: Cluttered backgrounds, occluded leaves, or other objects in the image can interfere with leaf disease detection (Wang et al. 2021). Poor quality images might contain more noise and distractions, making it harder for the model to focus on the leaf itself
- Limited diversity: Poor quality images might not capture the full diversity of disease symptoms, leading to biased or incomplete training data. The model might struggle to generalize to unseen disease variations
- Loss of structural information: Poor quality images might lack the structural details of leaves such as vein patterns or leaf shape (Azlah et al. 2019). These details are important for distinguishing between different plant species and diseases

Pre-trained DCNNs like AlexNet, VGG, and GoogLeNet have been used in a number of automated leaf disease image detection systems reported in recent decades (Thenmozhi & Reddy 2019). Those DCNNs are helpful for identifying and categorizing leaf diseases. Instead, most farm-based leaf disease images are fuzzy, low-resolution affairs. Low-quality images significantly degrade the classification performance of pre-trained DCNN classifiers, which are typically trained on clear high-resolution datasets. In order to improve the accuracy of agricultural disease image classification, low-resolution images must be high-resolved to increase spatial resolution and replicate the high-frequency characteristics of sharp edges.

To better diagnose agricultural leaf diseases, a novel Dual-Attention and Topology-Fusion with Generative Adversarial Network (DATFGAN) (Dai et al., 2020) can up-resolution low-resolution leaf images. They used the weight allocation policy to train deeper structures that classify leaf diseases according to texture features while reducing the number of parameters. However, the goal of a GAN was to train a generator that adapts a prior latent distribution to actual data. By improving algorithms to coordinate the generator and discriminator networks, its training time was increased. Also, the diseases could have affected the leaf partially or completely.

There is a need for the Positional-aware Generative Adversarial Network (PGAN) model to learn the spatial relationships among multiple leaf image observations and increase the accuracy of classifying leaf diseases.

4.3 HIGH-RESOLUTION LEAF DISEASE IMAGE GENERATION CLASSIFICATION

4.3.1 High-Resolution Leaf Disease Image Generation

Generating high-resolution leaf disease image is a technique used to enhance the resolution of low-quality or low-resolution leaf images. This process involves generating high-resolution versions of images while preserving important details and features. It is useful for various purposes, such as creating realistic training datasets, generating synthetic data for research, or enhancing the visual quality of diagnostic tools (Abayomi-Alli et al. 2021). We'll go over a few methods for creating images of excellent quality of leaf diseases.

4.3.1.1 Data Augmentation

"data augmentation" describes an approach to expanding the variety of information available from a dataset by altering its current images. This is commonly done by introducing small variations to the images while preserving their semantic content. In the context of high-resolution leaf disease image creation, data augmentation is used to generate new images with the same disease characteristics but slightly different visual properties. This technique helps improvement in model generalization and performance may be achieved by expanding the amount and variety of the dataset (Liu et al. 2020).

Augmentations such as rotation, flipping, cropping, zooming, deblurring, denoising, colour variations (brightness, contrast, Hue and saturation), and resolution adjustments (Downscaling and upscaling) can simulate different viewing angles and lighting conditions, effectively increasing the effective resolution of leaf disease image dataset (Shorten & Khoshgoftaar 2019).

By applying these augmentations, variations of high-resolution leaf disease images can be generated that resemble real-world conditions. This augmented dataset can then be used for training DL models, improving their ability to generalize to different scenarios and variations. To keep the produced images realistic and useful for appropriate diagnosis, however, a balance must be struck between augmentation and keeping the integrity of the underlying illness patterns.

4.3.1.2 Texture Synthesis

Texture synthesis is a technique used to generate new images with textures that resemble those found in existing images. In the context of high-resolution leaf disease image generation, texture synthesis involves creating new images that exhibit the same or similar textural patterns as real disease-affected leaves (Hasan et al. 2022). This technique can be particularly useful for generating high-resolution images of leaf diseases with intricate and distinctive texture details.

Texture synthesis methods, like patch-based or example-based approaches, can generate high-resolution textures that mimic the appearance of real leaf diseases. These techniques extract texture patterns from existing images and apply them to

generate new high-resolution textures (Wei et al. 2009). But it might not capture the underlying structural and semantic information of the original disease. This could lead to generated images that lack accurate disease features. Also, these techniques can be computationally intensive, resulting in long processing time and less practical for large-scale image generation.

Texture synthesis might not consider the contextual information present in the original image, such as the spatial relationships between textures or their interactions with other image components. Some disease patterns are inherently complex and involve interactions between different textures and structures. Capturing such complex patterns accurately through texture synthesis can be challenging.

4.3.1.3 Data Fusion

Integrating information from several sources or forms of analysis, as is done in data fusion, provides a more comprehensive images of the data at hand. Data fusion may be utilized to improve the quality, accuracy, and realism of produced high-resolution leaf disease images by combining data from many sources (Ouhami et al. 2021). For instance, combining colour and multispectral data can result in high-resolution images that capture both colour and spectral information. Some data fusion techniques applied to create high-resolution leaf disease images include:

- Combining spatial and spectral information: Leaf diseases often manifest in both the spatial and spectral domains. Data fusion can leverage spatial and spectral information to generate high-resolution images that reflect the intricate spatial patterns of diseases and their spectral signatures
- Enhancing texture and detail: Fusion techniques can integrate texture information from one image source with structural details from another. This can result in generated images that exhibit more realistic and visually appealing textures, making them suitable for classification
- Super-resolution fusion: Super-resolution methods may be used with other data sources to boost the resolution of produced images. For instance, a low-resolution image of a diseased leaf can be fused with high-resolution texture patterns to create a more detailed and high-quality image

- Feature-level fusion: It involves extracting relevant features from different data sources and combining them to create a more informative representation. In the context of high-resolution leaf disease image generation, this can lead to images that accurately capture disease-specific features
- Neural network fusion: DL methods, such as neural networks, may be used to combine data from several sources. They can learn to extract and combine relevant features, resulting in high-resolution images that effectively leverage the strengths of each data source
- Semantic information fusion: Data fusion can incorporate semantic information, such as disease labels or regions of interest, to guide the generation process. This can lead to high-resolution images that accurately represent specific disease patterns

4.3.1.4 Image-to-Image Translation Using GAN Variants

The purpose of the image-to-image translation method is to transfer images from one domain to another without losing their significance. In the context of creating high-resolution images of leaves, image-to-image translation is used to convert images of healthy leaves (the source domain) into images of diseased leaves (the destination domain). This technique is often employed using DL models, such as Generative Adversarial Network (GAN) and its variants, to achieve realistic and accurate transformations (Lu et al. 2022).

A. **GAN**

High-resolution images of leaf diseases may be synthesized using GANs, and they will seem quite similar to the original data. To make synthetic images, GANs use a generator network, and to tell the difference between genuine and artificial images, GANs use a discriminator network (Liu et al. 2020). By iteratively training these networks, GANs can produce high-quality, high-resolution images.

Since its introduction by Goodfellow et al. (2020), GANs have been used by several scholars across many disciplines. The generator and discriminator of a GAN are two antagonistic components that engage in a minimax game to achieve their respective goals. The goal of the generator is to mimic the distribution of actual data and provide

convincing fake samples to the discriminator. Alternatively, discriminator goals are sample origin determination. The cost of each network is proportional to the success of the competing component, and this competition between the discriminator and the generator may be described using the following formulation (Eq. 4.1) with value function $V(G, D)$.

$$\min_G \min_D V(D, G) = E_x[log\, D(x)] + E_z[log(1 - D(G(z)))] \qquad (4.1)$$

$D(x)$ in Eq. (4.1) is the discriminator's assessment of the probability that data instance x is authentic, while the generator's response to noise z is denoted by G(z), The probability that a false instance is genuine, as determined by the discriminator, is represented by $D(G(z))$, E_x irepresents the average value of all occurrences of actual data, and E_z is the average value of the generator given a set of random inputs. Fig 4.1 depicts an overarching framework for the GAN training process. Digital Elevation Models (DEMs) of poor resolution may be upgraded using the Generator. To determine if high-resolution DEMs are genuine or false, the discriminator compares them to examples from the training set.

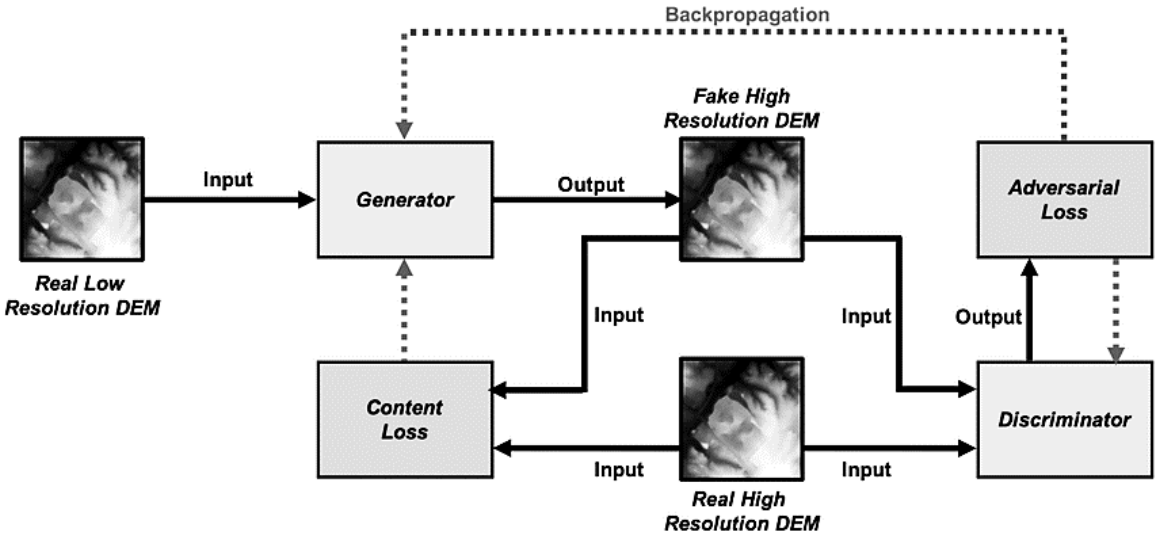

Fig. 4.1 Architecture of the GAN Training Procedure (Demiray et al. 2021)

Based on the discriminator's performance, the adversarial loss is computed, and the weights of the discriminator and the generator are adjusted. While feeding back to the generator, the content loss is determined by comparing the created DEMs to the original high-resolution ones pixel-by-pixel.

B. Super-Resolution GANs

Super-Resolution GAN (SRGAN) is a specialized type of GAN designed to enhance the resolution of images, including high-resolution leaf disease images. SRGANs use DL approaches to produce high-quality, high-resolution imagery from low-resolution inputs, essentially "super-resolving" the images while keeping realistic and visually correct features (Wang et al. 2022). Medical imaging and remote sensing are only two examples of the many applications of this method for creating and improving images.

A SRGAN consists of a generator and a discriminator, as shown in Fig. 4.2.

- Generator: In order to generate a high-resolution image, the generator must first receive a low-resolution image as input. It employs DCNNs to upscale the image while adding high-frequency details that are consistent with the high-resolution data distribution
- Discriminator: It is a method for verifying the authenticity of high-resolution images. It directs the generator to create images that seem just like actual high-resolution ones.

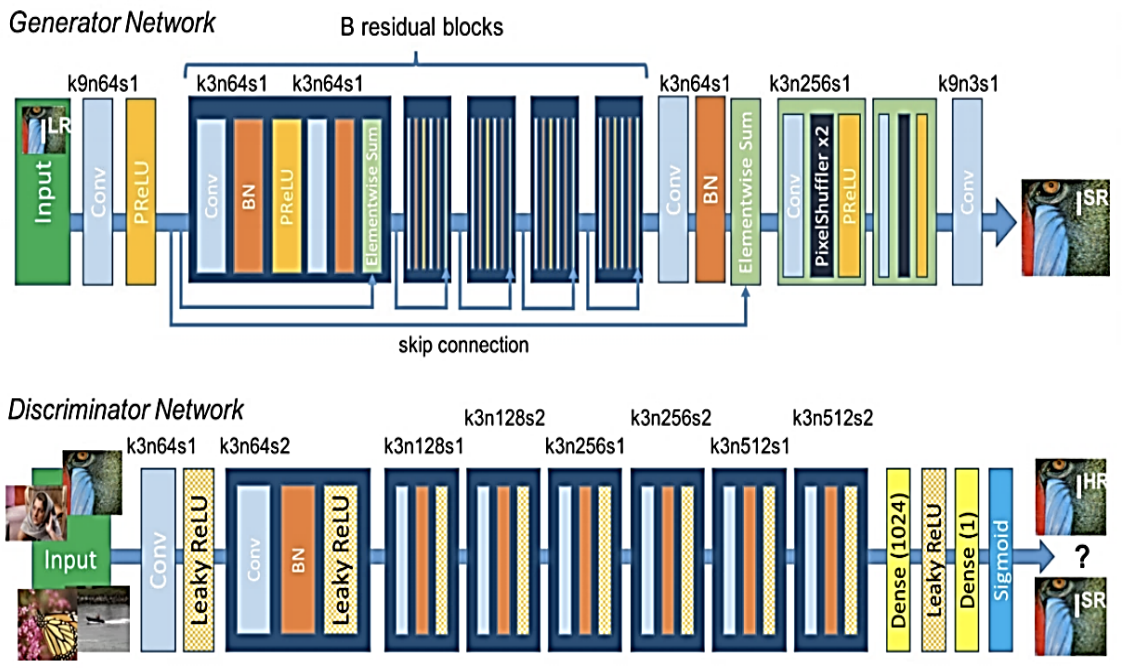

Fig. 4.2 Architecture of Generator and Discriminator Network (Ledig et al. 2017)

SRGAN training follows the standard GAN training process:

- The generator attempts to create high-resolution images that can "fool" the discriminator into categorizing them as actual images.
- The discriminator seeks to accurately categorize images of excellent quality as either authentic or fraudulent, given just their resolution.

As the generator learns to create more convincing high-resolution images, the discriminator becomes better at telling genuine from produced images, and the training loop continues. To help direct the training process and guarantee the quality of produced images, SRGAN makes use of a number of loss functions (Ledig et al., 2022).

- Adversarial loss: promotes the generation of images with excellent resolution that may deceive the discriminator into thinking they are genuine.
- Perceptual loss: Calculates the degree to which a pre-trained CNN can distinguish between genuine and synthetic visual features. Because of this, the creator is incentivized to protect crucial structural and visual aspects.
- Content loss: Similar to perceptual loss, content loss focuses on the preservation of image content and texture details

Disadvantages of SRGAN

SRGANs also have specific disadvantages and challenges when applied to high-resolution leaf disease image generation. The diagnostic, research, and training applications of the produced images may suffer as a result of these drawbacks.

- Loss of disease-specific details: SRGANs might prioritize general texture enhancement over disease-specific details. This could result in generated images that lack accurate disease patterns and characteristics, limiting their usefulness for accurate disease detection and analysis
- Unrealistic disease manifestations: Generated images might exaggerate disease patterns or create unrealistic manifestations that do not accurately represent the variations found in real-world leaf diseases

- Difficulty in capturing subtle features: Some leaf diseases exhibit subtle colour changes, minor texture variations, or small lesions. SRGANs might struggle to faithfully capture these fine-grained features, leading to inaccuracies in generated images
- Dependency on high-quality low-resolution inputs: SRGANs heavily rely on the quality of low-resolution inputs. If the low-resolution images are of poor quality or do not accurately represent the disease patterns, the generated high-resolution images might not be reliable
- Complex disease interactions: Some leaf diseases interact with each other or exhibit complex patterns that are challenging to model accurately using SRGANs. This can lead to inconsistencies or inaccuracies in generated images
- Evaluation challenges: Accurately evaluating the quality and realism of generated high-resolution leaf disease images can be difficult. Conventional metrics might not capture disease-specific features, and subjective evaluation by experts is often necessary
- Data artifacts: If the low-resolution images used for training contain artifacts or noise, the SRGAN might inadvertently amplify these artifacts when generating high-resolution images
- Limited interpretability: The generated images might lack interpretability, making it challenging to understand the underlying disease patterns that contributed to their creation

Combining SRGANs with other techniques, such as data fusion or attention, can also help address some of these limitations. Ultimately, understanding the potential disadvantages of using SRGANs for high-resolution leaf disease image generation is essential for making informed decisions about their deployment and interpreting their results.

C. Dual-Attention and Topology Fusion Generative Adversarial Network

The Deep Attention and Topology Fusion Generative Adversarial Network (DATFGAN) model was developed by Dai, et al., in 2020. It's made up of a network for extracting deep features, one for fusing attention to topologies, and another for

reconstructing images, all of which share parameters. To create high-resolution images, the reconstruction network makes advantage of global residual learning, whereas the generator network, as illustrated in Fig 4.3, comprises two convolutional layers for shallow features.

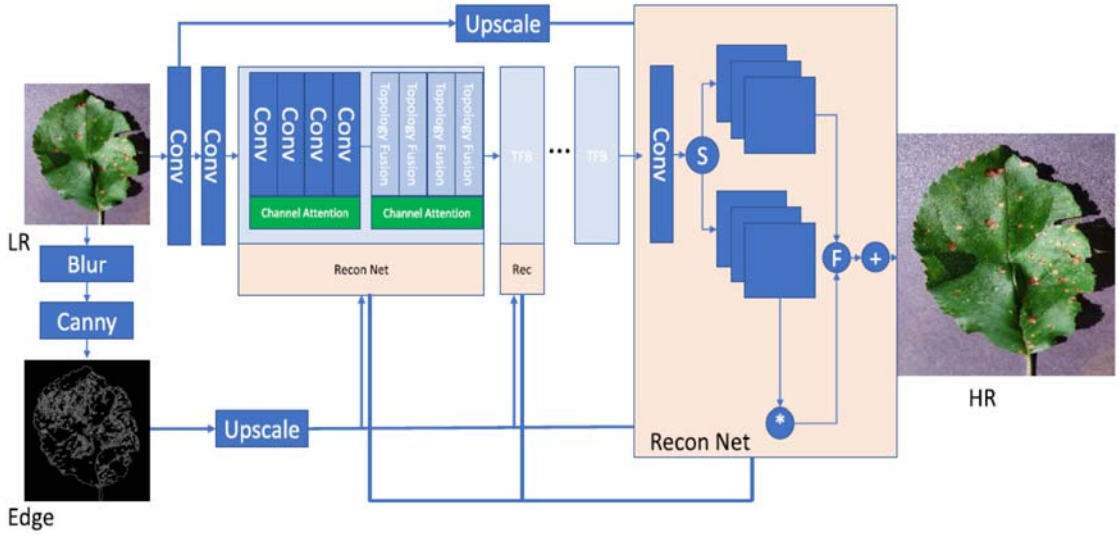

Fig. 4.3 Generator Network in DATFGAN

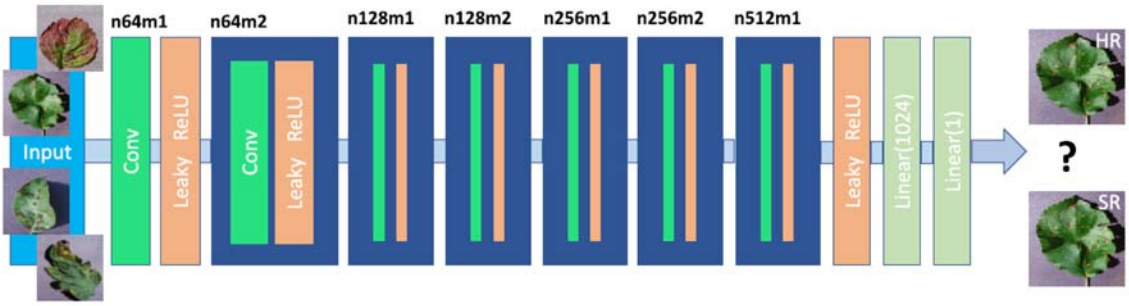

Fig. 4.4 Discriminator Network in DATFGAN

The discriminator network is seven convolutional layers deep, where the filter kernels increase in size with each layer. It is trained to maximise a target variable. To improve the probability of image classification, researchers first reduce the image resolution using striding convolutions, and then feed the resulting 512 feature maps into a LeakyReLU activation function and two linear layers (Fig. 4.4).

- **Parameter Sharing**

It is the local information extracted by convolution processes that may be utilized in various places throughout an image. The CNN uses a convolution kernel to extract features from the input data one at a time. However, parameter explosion in the convolution layer is a common occurrence if the input data contains numerous dimensions. Without taking into account local correlations, the convolution kernels each extract their own unique set of characteristics. Parameter sharing enables each feature to appear in many places in different data, hence reducing the number of parameters required in the convolutional layer (Dai et al., 2020).

The number of convolutional layer parameters may be reduced via weight sharing. Parameter-sharing attention may be used to reduce the number of network parameters, reduce the likelihood of overfitting, and improve the trainability of deeper structures."-improved topology"The DATFGAN generating network made use of fusion networks.

- **Topology Fusion**

ResNet (He et al. 2016) was developed to address the degradation problem in DL by reducing error rates and optimizing complex models. ResNet's residual block, implemented through residual connections, increases training speed without additional parameters or calculations. By building residual connections between the front and rear layers, this architecture enhances the overall training efficiency of deep CNNs.

Like ResNet, DenseNet (Huang et al. 2017) makes advantage of dense connections between layers to improve performance while decreasing the quantity of inputs and the amount of processing time. It also accomplishes direct fusion of feature maps, improving feature reuse.

Contiguous memory for DATFGAN is supplied by the generator's mix of residual and dense connections in a single layer, which halves the channel formation rate. It improves information flow and gradients while decreasing network parameters and making deeper structures trainable. Mixed-link connections' inner structure is seen in Fig 4.5.

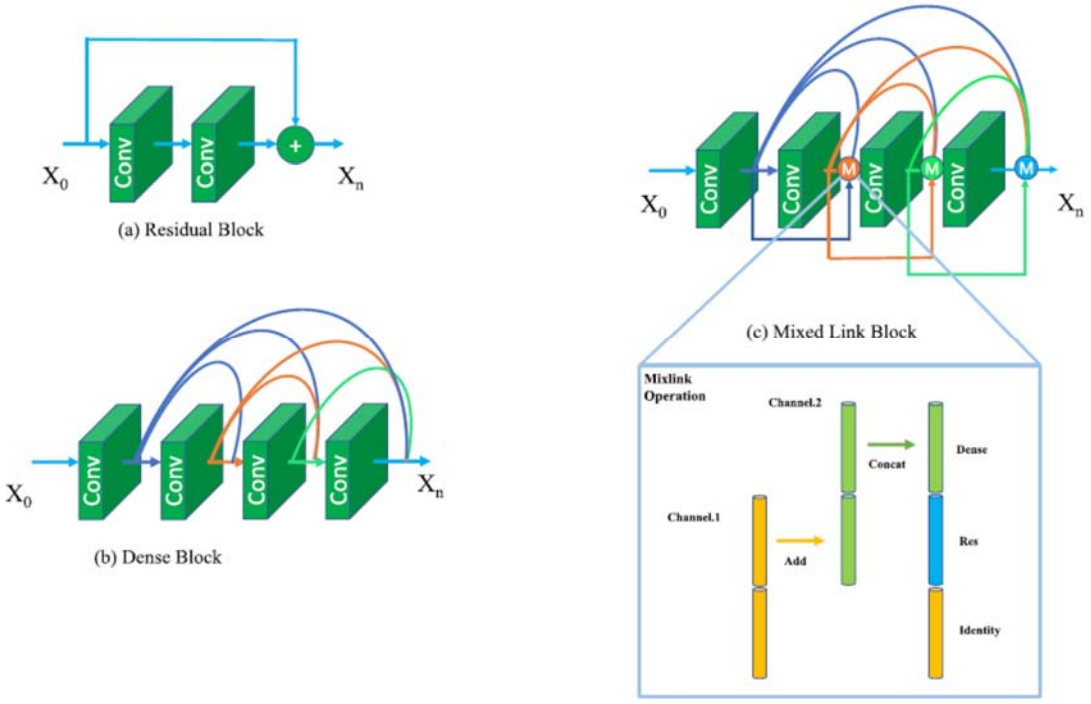

Fig. 4.5 Topology Fusion

Mixed-link operations are computed as follows:

$$F_{i-1}^1, F_{i-1}^2 = Slice(F_{i-1}) \tag{4.2}$$

$Slice(.)$ is a slicing operation, and it is used to split the input channels in half in Eq. (4.2). Since F_{i-1} is an N -channel feature map, F_{i-1}^1 and F_{i-1}^2 may each contain up to $\frac{N}{2}$ channels as a result of the slicing procedure.

$$F_i^1, F_i^2 = Slice(W(F_{i-1}) + b) \tag{4.3}$$

In Eq. (4.3), where W is a convolution layer's weight and b is the bias, the result of a single layer or unit is split in half along the channel dimension.

$$F_{i+1} = C(C(F_i^1 + F_{i-1}^2, F_i^2), F_{i-1}^1) \tag{4.4}$$

With respect to Eq. (4.4), $C(.)$ is a fusion operation, F_{i-1}^1 and F_{i-1}^2 represent subsets of the characteristics used in the previous layer, and F_i^1 and F_i^2 include characteristics extracted from the current layer in subsets. When F_i^1 and F_{i-1}^2 are added together, the resulting topology is residual, whereas when $F_i^1 + F_{i-1}^2, F_i^2$ and F_{i-1}^1 are fused, the resulting topology is dense.

As demonstrated in Eq. (4.5), the number of channels may be reduced by concatenating features across blocks using a transition convolution, where W_t is the weight of an 1×1 convolution for block-feature concatenation, F_{j-1} is the characteristics of the previous mixed-link section, and F_j shows the output characteristics of the active mixed-link block. This mixed-link strategy allows the DATFGAN to quickly build residual and dense connections, both of which control parameter expansion and improve network performance.

$$F_j = W_t\left(F_{j-1}\right) + b \tag{4.5}$$

- **Dual Attention**

Image transformation is made more effective by using channel and texture attention algorithms (Dai et al., 2020).

- Channel attention: Topology fusion is used to represent the interdependencies across convolution channels, learning independently how to emphasize relevant channels while downplaying noise. To rebalance the information and gradient flow across networks, it acts as a filter. To provide self-trained channel-wise attention, the module uses a global pooling layer, convolutions, and a sigmoid layer, as shown in Fig 4.6.

  Eqns. (4.6) and (4.7) are the basis for the operation of channel focus:

$$S(F) = \frac{1}{HW}\sum_i^H \sum_j^W F(i,j) \tag{4.6}$$

In Eq. (4.6), H and W stand for the width and height of the feature map used as input, respectively; $S(.)$ is a squeeze operation that pools the features in all channels into a global mean.

$$A(F) = \delta\left(W_u \sigma\left(W_d S(F)\right)\right) * F \tag{4.7}$$

Channel attention is denoted by $A(.)$ in Eq. (4.7), whereas ReLU is denoted by σ, and W_u and W_d are two 1×1 convolutions. First, the channels are shrunk by a factor of $\frac{1}{16}^{th}$ by W_d, and then a bottleneck is produced by the expansion of

the tensor to its true form by W_u. In addition, δ is the sigmoid function that scales all channel weights to a value between 0 and 1. These weights are used to highlight relevant data and hide irrelevant data.

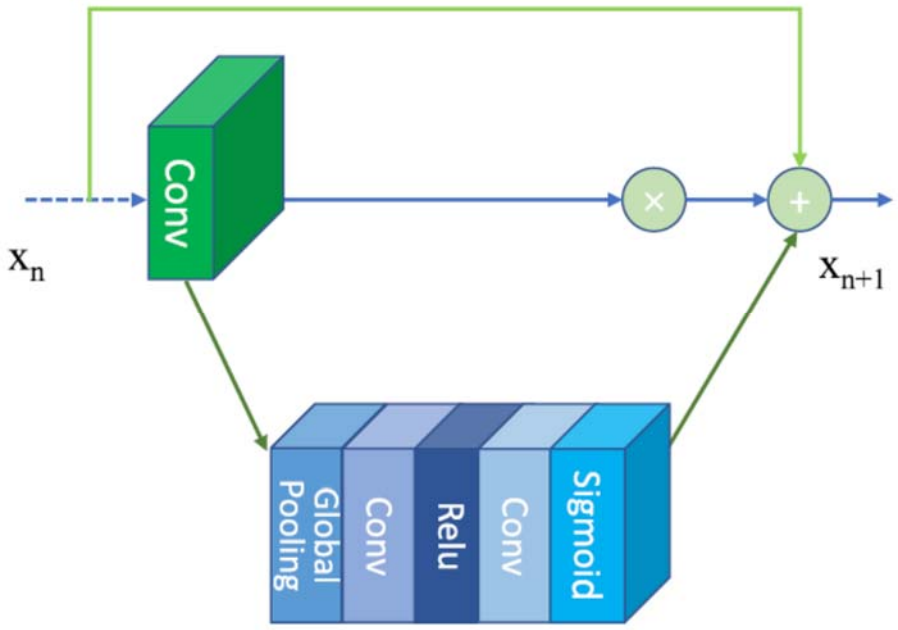

Fig. 4.6 Channel Attention

- Texture attention: High-frequency features in plant photos are often situated near the borders, making texture critical for image high-resolution applications. Eqns. (4.8) and (4.9), where W_{exp} signifies extending the actual number of channels, are utilized to design a reconstruction network that focuses on textures by paying attention to edges at the global spatial level. The number of global characteristics is doubled for this challenge. Only some of the channels have their weights adjusted based on global data; the others use only local data. As seen in Fig. 4.7, the two parts are combined by adding and averaging them.

$$F_i^1, F_i^2 = Slice\left(W_{exp}(F_{i-1})\right) \tag{4.8}$$

$$F_{i+1} = Up\big(Canny(F_0)\big) * F_i^1 + F_i^2 \tag{4.9}$$

Up is an upsampling operation, $Canny$ is an edge feature extraction operator, and F_0 is the original input features in Eqn. (4.9). divided by two utilising large-scale pixel maps for feature multiplication, and then combined with the other half of the input features.

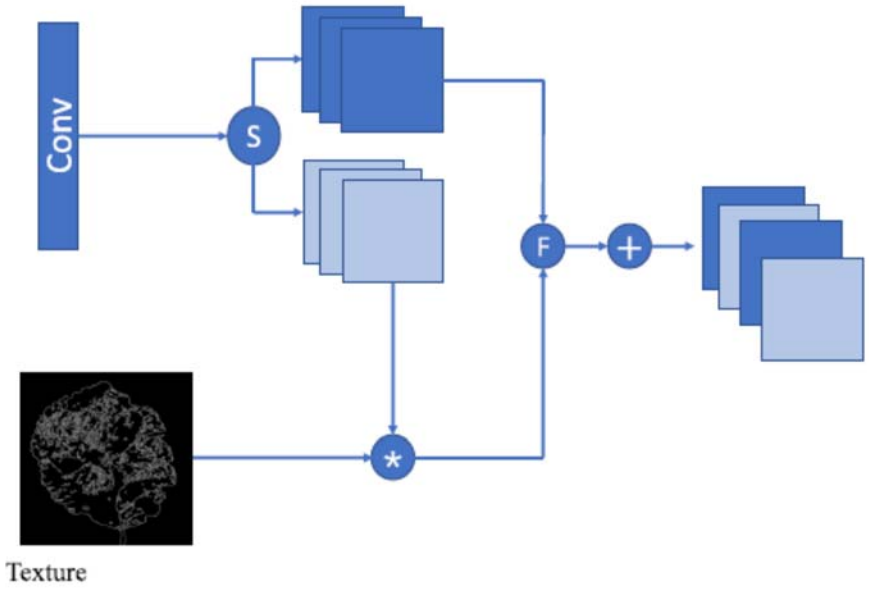

Fig. 4.7 Texture Attention

- **Adversarial Training**

Adversarial training is a kind of model improvement that uses competition to landscape loss function refinement for improved model performance. Although not expected to improve generalization ability, it can simplify prediction functions for real data, making them more smooth and simple.

Comparable inputs will provide comparable results in this strategy (Dai et al., 2020). For regularization to work, equivalent results must be achieved using a variety of inputs. In order to construct a confrontation sample and lower cross-entropy, it is essential to identify the most harmful disruption. To create more aesthetically pleasing pictures, adversarial training is used rather than minimising the MSE between input images and targets.

Adversarial loss is defined by

$$L_{GAN} = \mathbb{E}\left[D\left(G(I_{LR})\right)\right] - \mathbb{E}[D(I_{HR})] \tag{4.10}$$

Eq. (4.10) uses the DATFGAN discriminator $D(.)$, the generator $G(.)$, the produced pseudo-high-resolution pictures I_{LR}, and the real-time high-resolution images I_{HR}. The sum of loss and the adversarial loss is defined by

$$L = \alpha L_{GAN} + L_{content} \tag{4.11}$$

Eq. (4.11), where L is the overall loss, L_{GAN} is the adversarial loss, $L_{content}$ is the overall perceptual loss for the target content, and α is a constant value.

D. Positional-aware Dual-Attention and Topology Fusion Generative Adversarial Network

Because traditional GANs automatically produce entire images, memory and computing constraints dictate the maximum picture resolution that can be achieved. This work proposes the Positional-aware GAN (PGAN) as a solution to this problem by only producing a small region of an image based on its coordinates. The resulting pictures are then combined to form a single absolute global image. There are two networks that make up the PGAN: a generator (G) and a discriminator (D). The G integrates a micro-coordinate structure on a finer scale for G, a coarse-grained macro-coordinate structure for D, and images in three dimensions: full images (actual (a) and generated (x)), macro-patches (actual (a') and generated (x')), and micro-patches (generated (x'')). An enhanced topology-merging and re-creation network with a flexible attention distribution is also included. Using the coordinate structure of the shallow-feature selection, G is mined for its shallow features and the location of the image patch. The structure of G in PDATFGAN is shown in Fig 4.8.

In the first step, G is fed a collection of low-resolution picture patches that are then divided in half. An upscaling unit is given the first subset, then the first convolutional layer in G. After the second convolutional layer, a subset is sent to the topology merging unit for prediction. In order to construct high-resolution picture patches, the Restoration Network (RestoreNet) uses the global residual training to mix the upscaled patches with approximated information. The leaf image patches' feature vectors are then normalized using Conditional Batch Normalization (CBN).

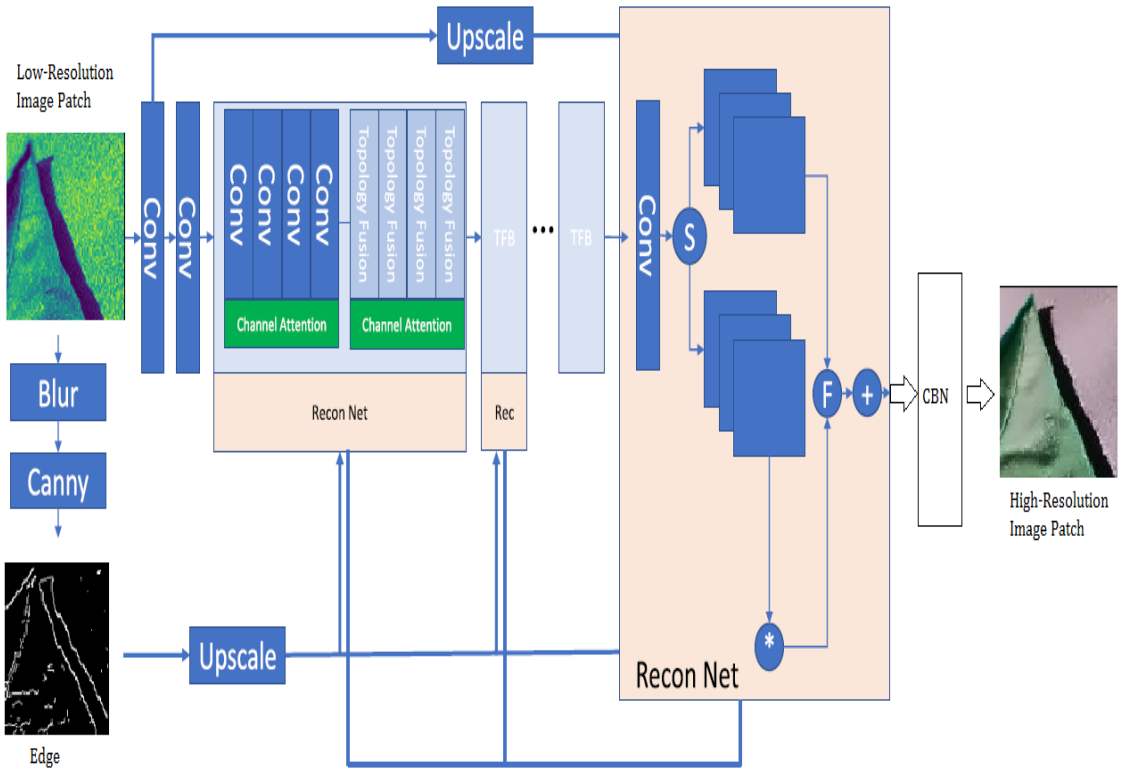

Fig. 4.8 Structure of G in PDATFGAN

$x'' = G(z, c'')$, where z is the latent vector and c'' is a micro-coordinate condition denoting the spatial location of x'' to be constructed, is a constrained framework inside PGAN that generates high-resolution micro-patches. The ultimate goal of G is to produce reasonable and faultless whole images by entirely gathering a set of x'' with a fusion factor φ. For high-resolution image patches, it is sufficient to configure φ as an aggregation factor with no overlap, since this is PGAN does automatically. To learn D, a partition conversion φ, in which a macro-patch a' is created by dividing an actual image a into smaller patches, $\varphi$is used to simulate real macro-patches.

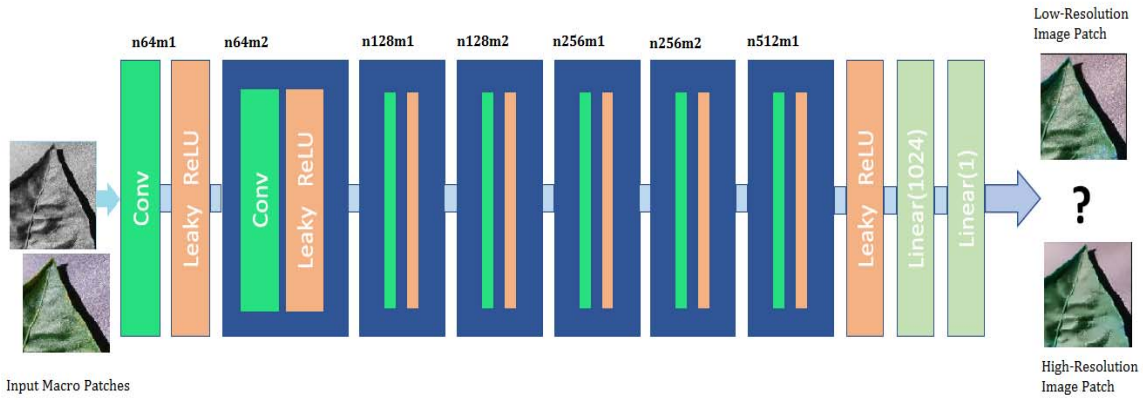

Fig. 4.9 Structure of D in PDATFGAN

In this layout, the ridges between consecutive patches are the main hindrance to positive identification. D is then taught with these larger macro-patches, which are compiled from many smaller micro-patches, to solve the issue. The goal of this model is to guarantee, relative to adversarial loss, the stability and consistency of a large number of consecutive or nearby micro-patches. For the G to manipulate D, it has to close the gaps between the patches it creates. The following loss factors are used to train this PGAN: adversarial loss (L_{PGAN}), total perceptual loss ($L_{Content}$), spatial consistency loss (L_S) and gradient penalty loss (L_{GP}). This PGAN only works with macro- and micro-patches for L_{PGAN} and L_{GP}, whereas traditional GAN makes use of whole pictures for both G and D training. L_S is a GAN loss factor that acts similarly to an auxiliary classifier. Fig. 4.9 depicts the organization of D inside PDATFGAN.

The macro-coordinate c' of the macro-patches a' is determined according to the set-up of φ. In addition, L_S and $L_{Content}$ want to lessen the disparity between c' and the D-approximated $\hat{c}'$ by closing the gap between the two values. Here are some of PGAN's losing factors:

$$\begin{cases} L_{PGAN} + L_{Content} + \lambda L_{GP} + \alpha L_S, & for\ D \\ -L_{PGAN} + \alpha L_S, & for\ G \end{cases} \tag{4.12}$$

(i) Spatial Coordinate System

Create a micro-coordinate framework for G and framework for large-scale coordination of D are the first steps. The matrix of microcoordinates is connected to each macrocoordinate $c'_{(i,j)}$ by the expression: $C''_{(i,j)} = \left[c''_{(i:i+N,j:j+M)}\right]$ whose absolute structure is as follows:

$$c''_{(i,j)} = \begin{bmatrix} c''_{(i,j)}\ c''_{(i,j+1)}\ \cdots\ c''_{(i,j+M-1)} \\ c''_{(i+1,j)}\ c''_{(i+1,j+1)}\ \cdots\ c''_{(i+1,j+M-1)} \\ \vdots\ \vdots\ \ddots\ \vdots \\ c''_{(i+N-1,j)}\ c''_{(i+N-1,j+1)}\ \cdots\ c''_{(i+N-1,j+M-1)} \end{bmatrix} \tag{4.13}$$

Equal time is spent sampling all possible combinations of $c''_{(i,j)}$ during PGAN training. By training using $G\left(z, c''_{(i,j)}\right)$, the G is constrained to produce high-resolution $x''_{(i,j)}$. By randomly scattering values of z over $C''_{(i,j)}$, the micro-patches matrix $X''_{(i,j)} = G\left(z, C''_{(i,j)}\right)$ is produced independently. In order to produce $c''_{(i,j)}$, it is assumed that the $X''_{(i,j)}$ are physically closer to one another. Following this, micro-patches are fused utilising φ to give an absolute $x'_{(i,j)} = \varphi\left(X''_{(i,j)}\right)$ as a coarser preview of the pictures complete-sight. Under the macro-coordinate structure for $c'_{(i,j)}$, $x'_{(i,j)}$ is also constructed with a produced $C''_{(i,j)}$. Actual $a'_{(i,j)} = \psi\left(a, c''_{(i,j)}\right)$ is formed based on the sampled macro-coordinates $c'_{(i,j)}$ in the real-time image scenario. Also, note that the selection of $C''_{(i,j)}$ is related with the topological aspect of the micro/macro-coordinate structures.When the micro-coordinate structure is solved for, the matching spatial coordinate matrix C''_{entire} may be found. This matrix is used to independently generate each micro-patch that makes up the final image. Several high-resolution micro-patches are generated and then stitched together to provide an entire image of a leaf.

(ii) Loss Functions

The adversarial loss L_{PGAN} is used in this model so that D can tell the difference between the true a' and the forged x'. Moreover, it helps G to deceive D with fake but functional micro-patches x''. It is defined as:

$$L_{PGAN} = \mathop{\mathbb{E}}_{a,c'}\left[D\left(\psi(a,c')\right)\right] - \mathop{\mathbb{E}}_{z,C''}\left[D\left(\varphi\left(G(z,C'')\right)\right)\right] \tag{4.14}$$

Coordinates c' and c'' in Eq. (4.14), representing macro- and micro-patches on D and G, respectively. Note that the micro-patches, denoted by the expression $G(z, C'')$, are the result of distinct processes. Macro-patches' differentiation is also subject to a gradient penalty:

$$L_{GP} = \mathop{\mathbb{E}}_{\hat{x}'}[(\|\nabla_{\hat{x}'} D(\hat{x}')\|_2 - 1)^2] \tag{4.15}$$

Using a random value in the range $\varepsilon \in [0,1]$, Eq. (4.15), $\hat{x}' = \varepsilon x' + (1 - \varepsilon)x'$ is calculated between the connected x' and a'. At end, the spatial uniformity loss L_S is comparable to a GAN loss with an auxiliary classifier. The D is performed with the help of a supplementary estimator A that attempts to gauge the macro-coordinate of a macro-patch along $A(a')$. In contrast to the discrete setup of an auxiliary classifier-like GAN, the continuous ranges in c'' and c' are rather large. Therefore, L_S is subjected to a measurement loss due to a gap. The goal is to master G such that similar micro-patches may be generated using $G(z, c'')$ in terms of the spatial criteria c''under consideration. In order to define the spatial uniformity loss, follows:

$$L_S = \mathop{\mathbb{E}}_{c'}\|c' - A(a')\|_2 \tag{4.16}$$

(iii) Conditional Batch Normalization

In order to regularize and modulate the features of this positionally-aware GAN, to use CBN, which applies the same principles γ and β as those of normal BN's and provisional generators. It determines $o_K = \left({}^{(i_K - \mu_K)}/_{\sigma_K}\right) * \gamma + \beta$ for the K^{th} input feature i_K, output feature o_K, feature mean μ_K and feature variance σ_K. But, positional-aware GAN gives spatial coordinate and latent vector as uncertain inputs. So, two MLPs: $MLP_\gamma(z, c)$ and $MLP_\beta(z, c)$ are created for every CBN layer that conditionally generates γ and β.

4.3.2 Leaf Disease Image Classification Techniques

Leaf disease image classification is a frequent computer vision task that involves sorting pictures of leaves into healthy and unhealthy groups. The purpose of leaf disease image classification is to create a computerized system that, given an image of a plant's leaves, can correctly identify and categorize the illness or condition shown in the image

(Pujari et al., 2015). By attaining high accuracy of classifying leaf diseases and reducing the necessity for improper prevention measures, pre-trained DL models have established remarkable success in the crop management.

4.3.2.1 AlexNet

When it comes to image categorization, DL's meteoric rise may be directly attributed to AlexNet, a groundbreaking DCNN architecture. In 2012, it was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, and it took first place in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The development of AlexNet was a major step forward in computer vision, and it paved the way for many future CNN designs, including those employed for leaf disease image classification (Krizhevsky et al., 2017).

In all, AlexNet has eight layers: three FC layers and five convolutional layers. Fig. 4.10 demonstrates how this early DCNN design was able to capture hierarchical features and complicated patterns in images. However, it is worth noting that since the development of AlexNet, newer and more advanced CNN architectures have been introduced, which might offer improved performance for leaf disease image classification. These architectures include VGG16, ResNet, Inception, and more.

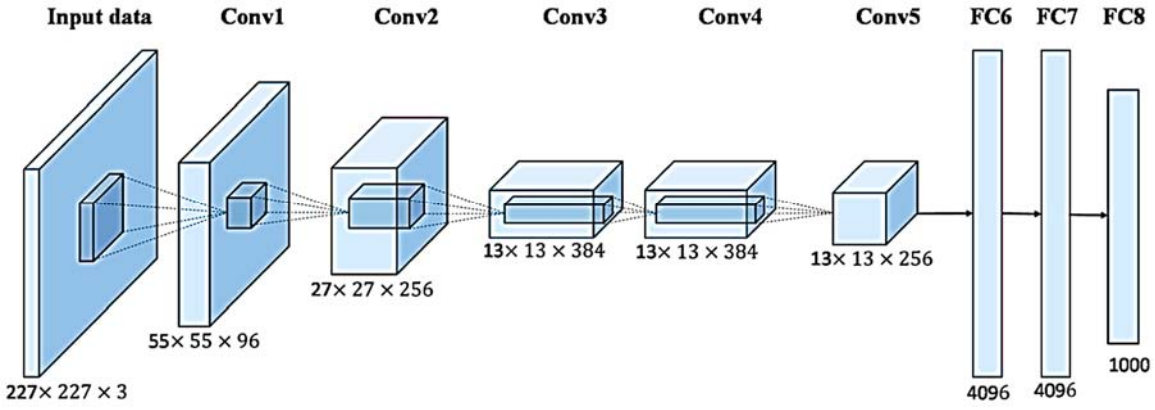

Fig. 4.10 Architecture of AlexNet

4.3.2.2 VGG16

Developed at Oxford University, the VGG16 is a DCNN design. It has had significant impact on computer vision and is largely regarded as a powerful model.

It has been employed for numerous image classification applications, including leaf disease image classification (Verma 2022). As shown in Fig 4.11, it has 16 layers, 13 of which are convolutional and 3 of which are completely linked.

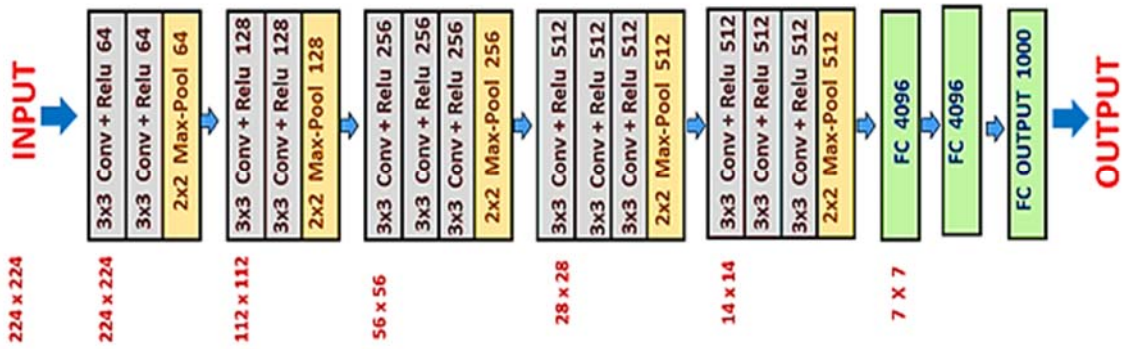

Fig. 4.11 Architecture of VGG16

4.3.2.3 InceptionV3

InceptionV3 is a DCNN architecture that was introduced as part of the Inception series of models. It was created to help with computer vision projects like image categorization. The purpose of InceptionV3, which is based on the same ideas as the original Inception architecture, is to be more effective in terms of computation and parameter consumption. (Xia et al. 2017). This architecture is well-suited for tasks like leaf disease image classification, where accuracy and efficiency are important factors.

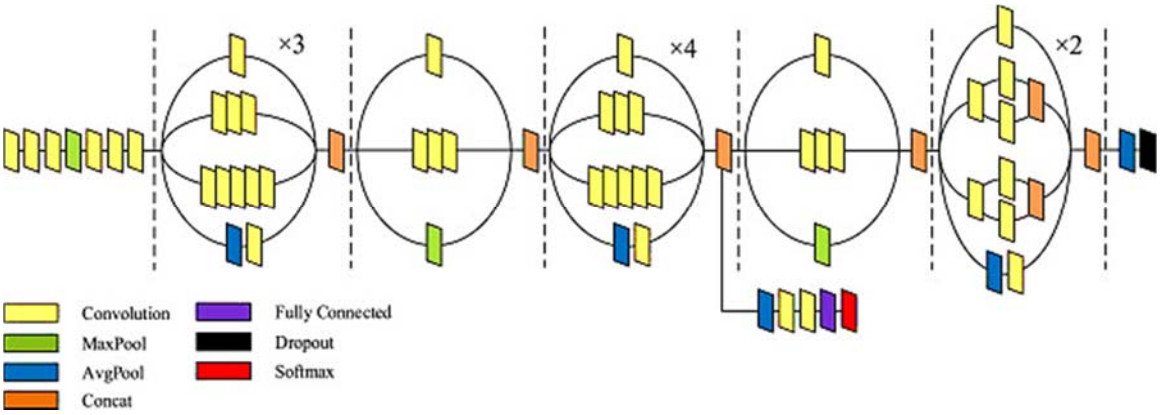

Fig. 4.12 Architecture of InceptionV3

The InceptionV3 architecture comprises of 42 convolutional layers and various auxiliary layers for training purposes. There are numerous Inception modules in the design,

each of which is a collection of convolutional layers with varying dimensionalities and kernel sizes. These modules help capture features at different scales and levels of abstraction. As can be seen in Fig. 4.12, it also has an FC layer, followed by a softmax layer, which generates the final classification result. In addition, it uses batch normalization and dropout regularization techniques to prevent overfitting and improve performance.

4.3.2.4 ResNet101

The ResNet (Residual Network) family of models includes the DCNN architecture known as ResNet101 (or "Residual Network with 101 layers"). ResNet101 is designed to overcome the challenges of training very deep neural networks by introducing the concept of residual connections, which alleviate the degradation problem caused by increasing network depth. He et al. (2016) found that this framework performed very well in classifying images of leaf diseases and other computer vision tasks. As can be seen in Fig 4.13, the ResNet-101 design is made up of 33 residual blocks. There are two convolutional layers in each remaining block, then a bypass link. Bypassing the convolutional layers, the shortcut connection links the residual block's input and output immediately.

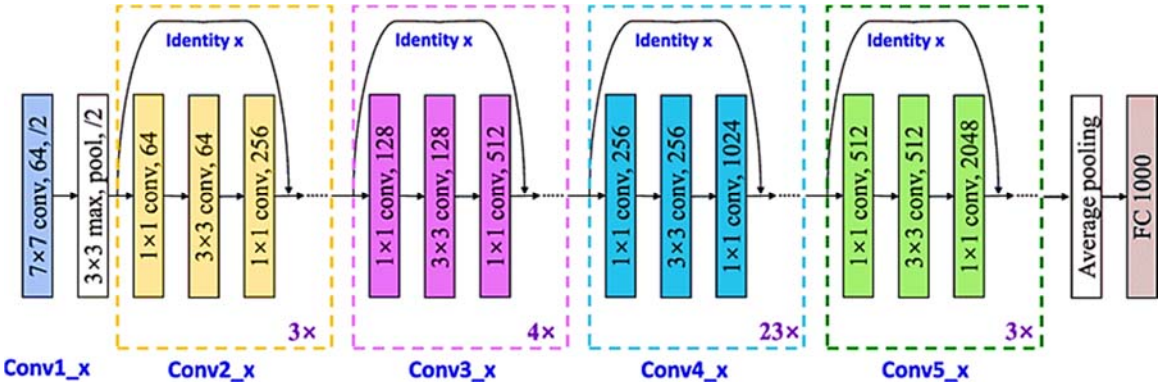

Fig. 4.13 Architecture of ResNet101

4.3.2.5 ResNeXt50

To boost its representational capacity and efficiency, ResNeXt50 is a variation of the ResNet architecture that incorporates the idea of cardinality. It builds on the idea of grouped convolutions, where multiple paths within a single layer learn different features in parallel. This design allows ResNeXt50 to achieve strong performance while being more memory and computationally efficient compared to other architectures (Xie et al. 2017).

ResNeXt50 uses a residual block with two convolutional layers and a bypass link. In the residual block, however, a bottleneck layer comes after the convolutional layers. The bottleneck layer is a small convolutional layer that reduces the number of channels in the output.

In addition, ResNeXt50 uses a technique called "cardinality" to increase the number of paths through the residual block. Cardinality is the number of parallel branches that are used in the residual block. ResNeXt50 uses a cardinality of 32, which means that there are 32 parallel branches in each residual block, as shown in Fig 4.14. ResNeXt50's performance is enhanced by the cardinality approach since it is then exposed to a more diverse set of characteristics from which to learn.

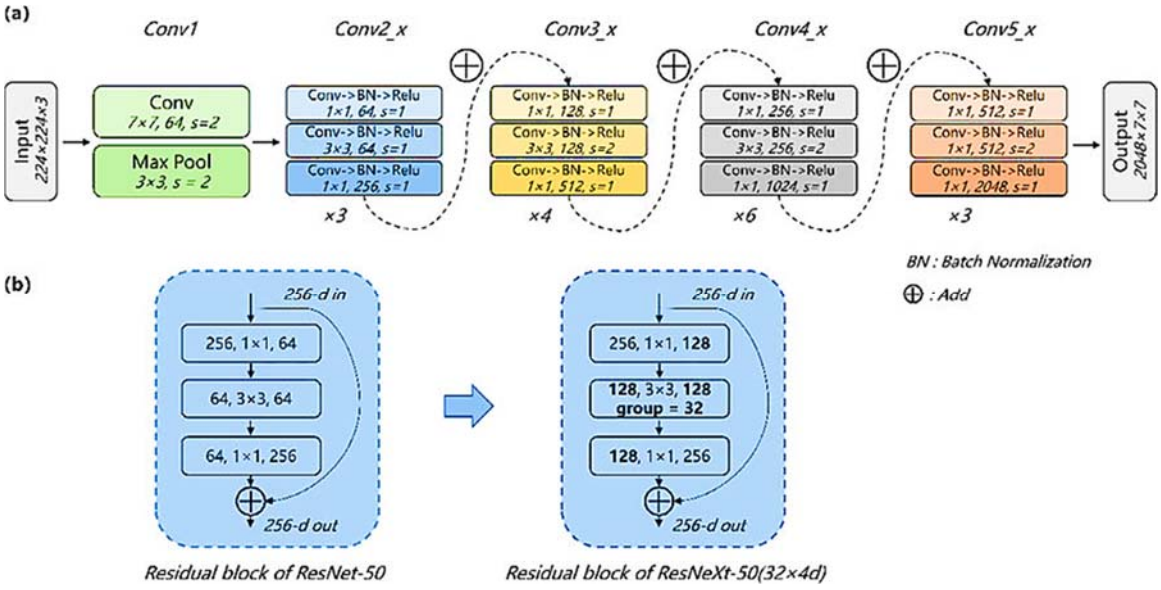

Fig. 4.14 Architecture of ResNeXt50. (a) ResNeXt Flow Diagram and (b) Residual Block Structure of ResNet50 and ResNeXt50

4.3.2.6 ShuffleNetV2

The channel split operator in ShuffleNetV2 divides the channels into two sets, keeping one set intact as the identity. Figure 4.15 and Table 4.1 depict the SuffleNetV2's topology and size, respectively. In contrast to 11 convolutions, which are not group-wise, the other branch's 3 convolutions have an equal number of input and output channels. ReLU, concat, and depth-wise convolutions are element-wise operations that can only be performed on one branch (Ma et al., 2018).

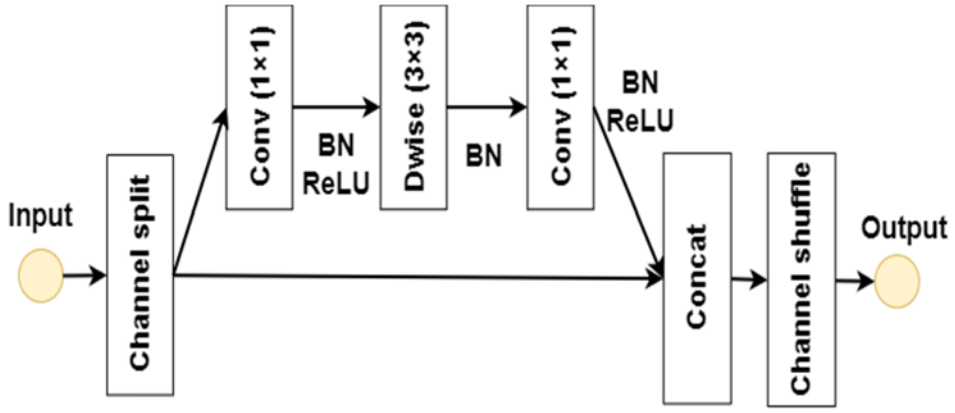

Fig 4.15 Structure of ShuffleNetV2 Classifier

Table 4.1 Design of ShuffleNetV2 Classifier

Layer	Outcome dimension	Kernel size/Stride
Input	224×224	
Convolution (Conv1)	112×112	3×3 $conv$ / 2
Pooling	56×56	3×3 maxpool / 2
Stage 2	28×28	-- / 2
	28×28	-- / 1
Stage 3	14×14	-- / 2
	14×14	-- / 1
Stage 4	7×7	-- / 2
	7×7	-- / 1
Conv5	7×7	1×1 $conv$ / 1, padding 0

4.3.2.7 DenseNet121

It certainly demands fewer variables than traditional architectures since its complicated connections prototype may not enable severe re-learning feature maps. The structure is partitioned into compact units, where the feature map dimensions are fixed within a given block but the number of filters is dynamic. It offers numerous advantages, including significantly reducing the number of variables, retaining the

features, and decreasing the vanishing gradient. The structure of DenseNet121 and their dimensions are presented in Fig 4.16 and Table 4.2, respectively. It has one convolution (112×112) and 4 dense units (Huang et al. 2017).

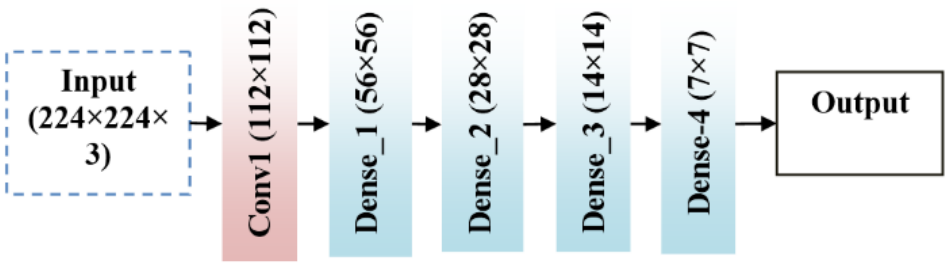

Fig. 4.16 Structure of DenseNet121 Classifier

Table 4.2 Design of DenseNet121 Classifier

Layer	Result dimension	Patch size/Stride
Conv	$112 \times 112 \times 64$	7×7 $conv$, stride 2, padding 3
Pooling	$56 \times 56 \times 64$	3×3 maxpool, stride 2, padding 1
Dense_1	$56 \times 56 \times 32$	$\begin{bmatrix} 1 \times 1\ conv, stride\ 1, padding\ 0 \\ 3 \times 3\ conv, stride\ 1, padding\ 1 \end{bmatrix} \times 6$
Transition_1	$56 \times 56 \times 128$ $28 \times 28 \times 128$	$1 \times 1 \times 128$ $conv$
Dense_2	$28 \times 28 \times 32$	$\begin{bmatrix} 1 \times 1\ conv, stride\ 1, padding\ 0 \\ 3 \times 3\ conv, stride\ 1, padding\ 1 \end{bmatrix} \times 12$
Transition_2	$28 \times 28 \times 256$ $14 \times 14 \times 256$	1×1 $conv$, stride 1, padding 0
Dense_3	$14 \times 14 \times 32$	$\begin{bmatrix} 1 \times 1\ conv, stride\ 1, padding\ 0 \\ 3 \times 3\ conv, stride\ 1, padding\ 1 \end{bmatrix} \times 24$
Transition_3	$14 \times 14 \times 896$ $7 \times 7 \times 896$	1×1 $conv$, stride 1, padding 0
Dense-4	$7 \times 7 \times 32$	$\begin{bmatrix} 1 \times 1\ conv, stride\ 1, padding\ 0 \\ 3 \times 3\ conv, stride\ 1, padding\ 1 \end{bmatrix} \times 16$

4.3.2.8 MobileNetV2

It progresses to MobileNetV1 and is used as the effective fundamental component by depthwise separable convolutions. To minimize dimensionality, two types of units are built in this structure: a residual unit with a stride 1 and another unit with a stride 2. Further, it includes linear blocks among the layers, which are required since nonlinearities prohibit more data from being affected. Such bottlenecks encode mid-level inputs and outputs (Sandler et al. 2018).

The internal layer can aid in the conversion of lower-level concepts such as pixels to higher-level descriptors. There are also shortcut links between bottlenecks. Figure 4.17 and Table 4.3 depict MobileNetV2's architecture and size, respectively.

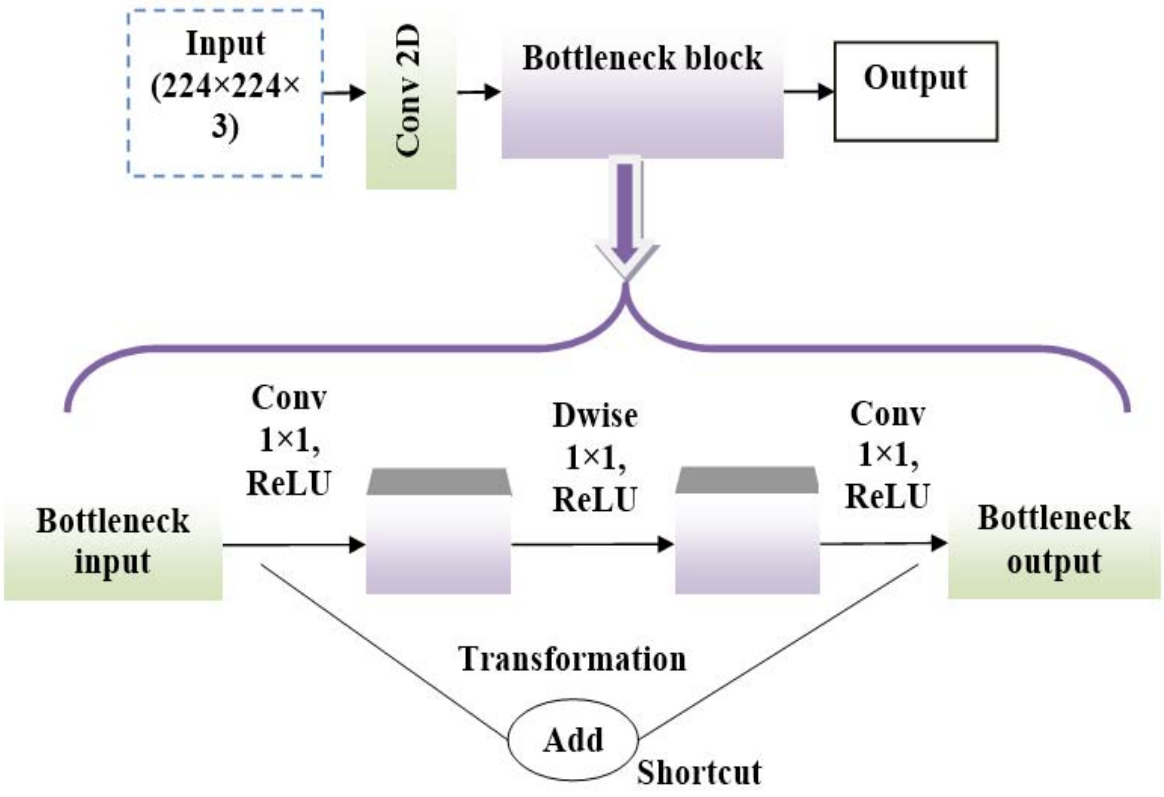

Fig. 4.17 Structure of MobileNetV2 Classifier

Table 4.3 Layout of MobileNetV2 Classifier

Layer	Outcome dimension	Patch size/Stride
Conv layer	$112 \times 112 \times 32$	$3 \times 3/2$
Bottleneck – 1	$112 \times 112 \times 16$	$-/1$
Bottleneck – 2	$56 \times 56 \times 24$	$-/2$
Bottleneck – 3	$28 \times 28 \times 32$	$-/2$
Bottleneck – 4	$14 \times 14 \times 64$	$-/2$
Bottleneck – 5	$14 \times 14 \times 96$	$-/1$
Bottleneck – 6	$7 \times 7 \times 160$	$-/2$
Bottleneck – 7	$7 \times 7 \times 320$	$-/1$
Conv layer	$7 \times 7 \times 1280$	$1 \times 1/1$
Mean pooling	$1 \times 1 \times 1280$	$7 \times 7/-$
Conv layer	k	$1 \times 1/1$

4.4 BUILDING THE PROPOSED MODEL

To address the issue of generating high-resolution leaf disease images for effective classification, the PDATFGAN model is proposed. The Positional-aware GAN (PGAN) model in high-resolution leaf disease image generation has been developed to build the model. The research works are carried out using the leaf disease image dataset PVD discussed in Chapter 3. As can be seen in Fig 4.18, the development of the Positional-aware Dual-Attention and Topology Fusion Generative Adversarial Network (PDATFGAN) that is the subject of this proposal goes through a number of stages. In addition, they are,

- Splitting the leaf disease image dataset into training and test sets
- High-resolution leaf disease image generation using PDATFGAN
- Leaf disease classification using pre-trained DCNN classification models

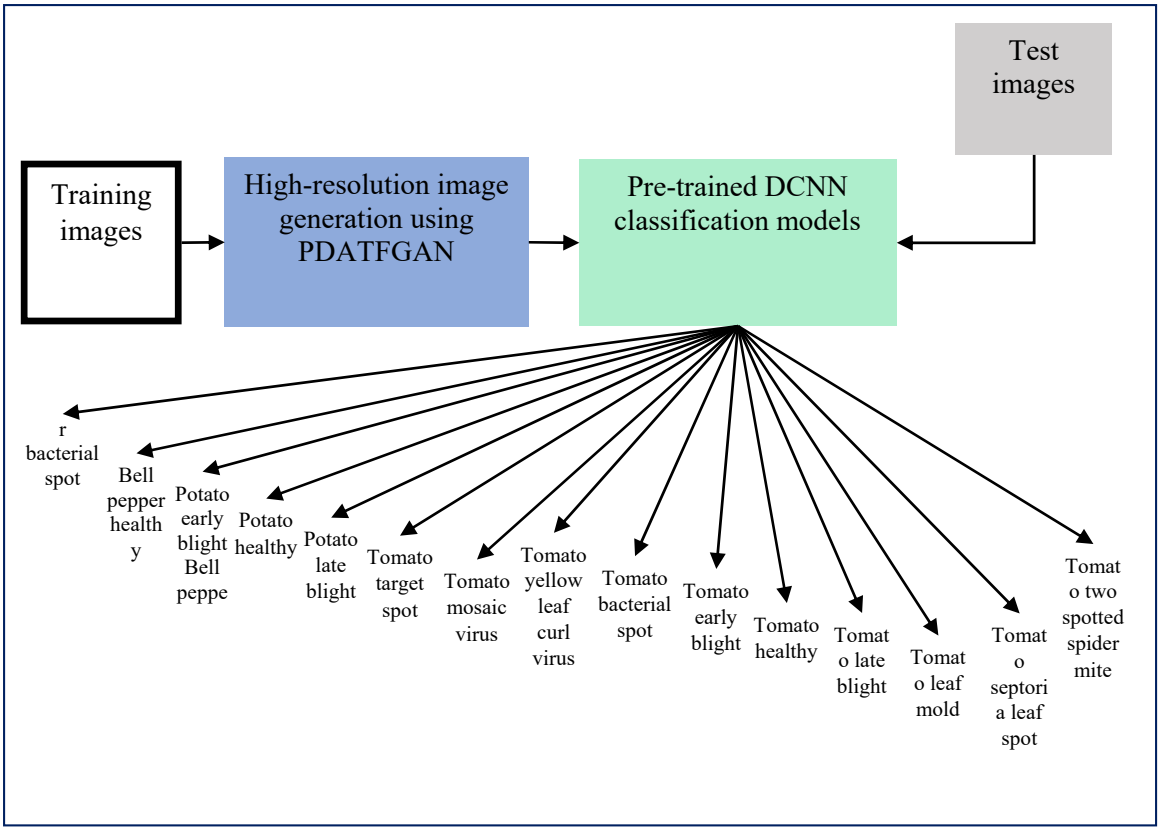

Fig. 4.18 Block Diagram of Proposed Model of PDATFGAN

4.4.1 Dataset Preparation

Leaf images from the Plant Village Dataset (PVD) are first divided into a training set and a test set, with both healthy and diseased examples included in each. The ratio between practice and evaluation is 70:30. The sample leaf disease images for various classes are shown in Chapter 3.

4.4.2 High-Resolution Leaf Disease Image Generation using PDATFGAN

In the next stage, as per the steps described in the PDATFGAN model given in section 4.3.1.4, the leaf images of distinct classes in the training image set are enhanced from low-resolution to high-resolution for classification. In the PDATFGAN, the network is built and trained using the hyperparameters listed in Table 4.4 with training images.

Table 4.4 Hyperparameters for PDATFGAN

Hyperparameters	Range
Optimizer	Adam
Learning rate	0.0001
Number of epochs	180
Momentum	0.9
Weight decay	0.0001
Mini-batch size	64

4.4.3 Leaf Disease Classification using Pre-trained DCNN Model

After generating the high-resolution leaf images in different classes, ShuffleNetV2, DenseNet121 and MobileNetV2 classifiers are performed as per the structure given in Fig 4.15, 4.16, and 4.17, respectively. In these classification models, the networks are built and trained using the parameters listed in Table 4.5. Finally, the trained classifiers are utilized to assign illness classifications to test images of leaves.

Table 4.5 Training Parameters for Different Pre-trained DCNN Classifiers

Models	Learning rate	Batch size	Epochs	Optimizer	Loss
DenseNet-121	0.0005	20	50	Adam	Cross-entropy
MobileNet V2	0.0001		60		
ShuffleNet V2			70		

4.5 RESULTS AND DISCUSSION

The measures used to evaluate the proposed PDATFGAN model's performance in comparison to the state-of-the-art models are shown below. Chapter 3 explains the datasets, assessment measures, and system settings in depth.

Table 4.6 displays the precision, recall, f-measure, and accuracy results obtained from testing the ShuffleNetV2 classifier model with the PVD raw dataset, the PVD augmented by the DATFGAN, and the PDATFGAN models.

Table 4.6 Comparison of the Proposed PDATFGAN Model Using ShuffleNetV2

Performance Evaluation Metrics	Raw dataset	Dataset enhanced by DATFGAN	Dataset enhanced by PDATFGAN
Precision	0.8954	0.9135	0.9148
Recall	0.8958	0.9140	0.9151
F-measure	0.8957	0.9142	0.9150
Accuracy	89.58%	91.38%	91.52%

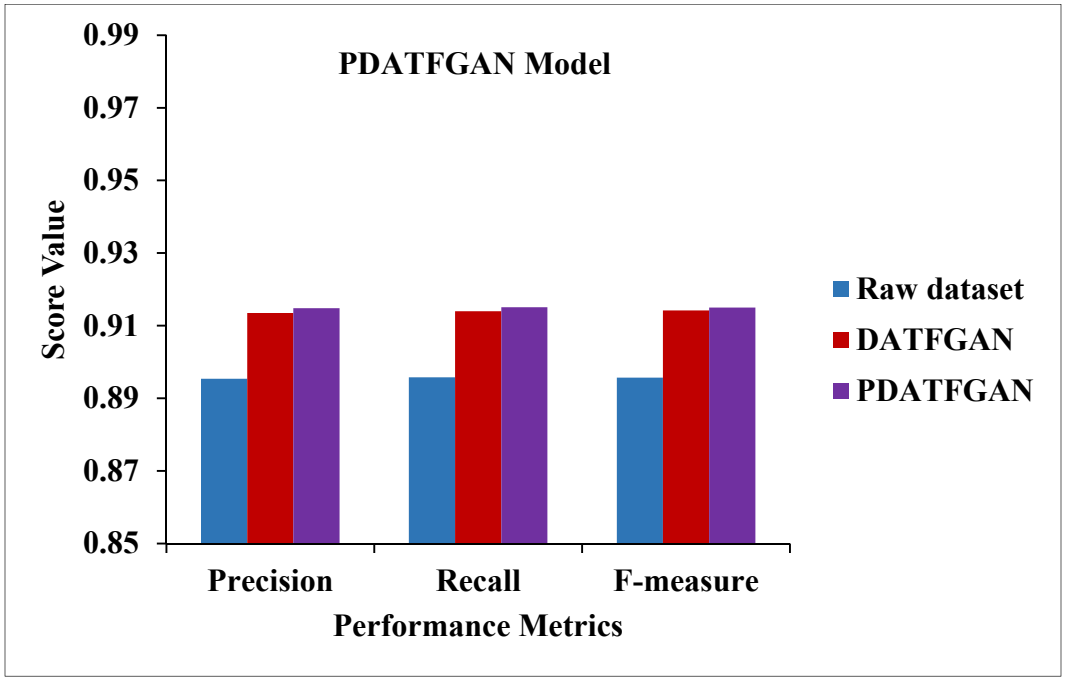

Fig. 4.19 Result of Proposed PDATFGAN Model Using ShuffleNetV2

Precision, recall, and f-measure are compared for a number of different GAN models used with the ShuffleNetV2 classifier on both raw and enriched PVD. Based on these findings, it is clear that the PDATFGAN-enhanced PVD is superior to the raw dataset and the DATFGAN-enhanced dataset for training the ShuffleNetV2. This is seen in Fig. 4.19. The precision of PDATFGAN-ShuffleNetV2 is increased up to 2.17% and 0.14% compared to the ShuffleNetV2 using raw dataset and DATFGAN-ShuffleNetV2, respectively. The recall of PDATFGAN-ShuffleNetV2 is improved by 2.15% and 0.12% compared to the ShuffleNetV2 using the raw dataset and DATFGAN-ShuffleNetV2, respectively. The f-measure of PDATFGAN-ShuffleNetV2 is raised to 2.15% and 0.09% than the ShuffleNetV2 using raw dataset and DATFGAN-ShuffleNetV2, respectively.

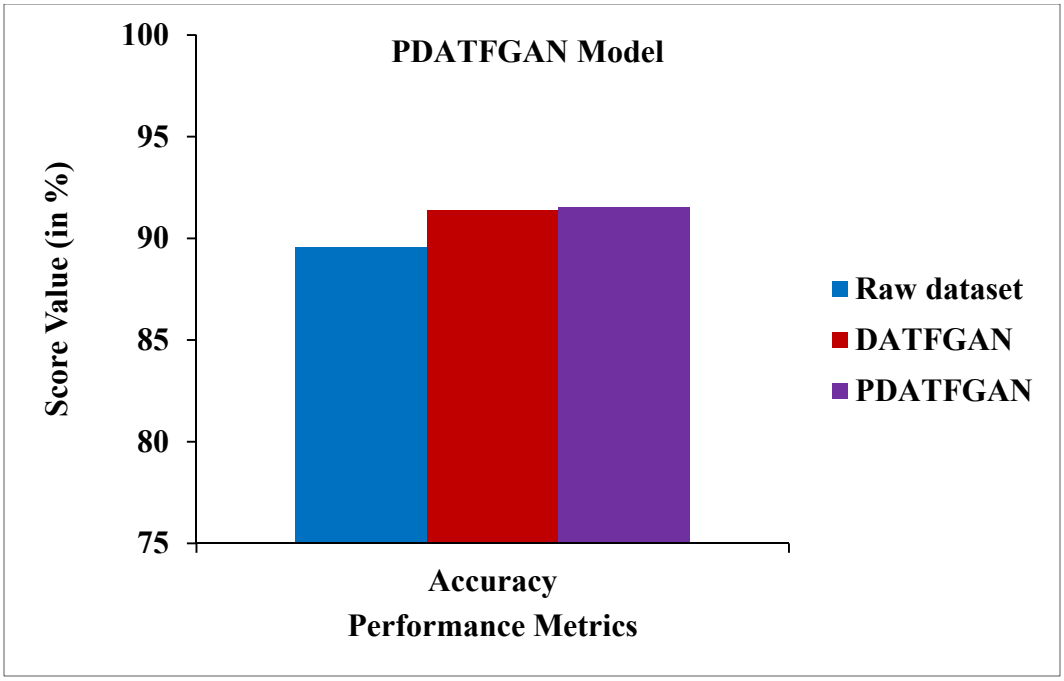

Fig. 4.20 Accuracy Comparison of PDATFGAN Model Using ShuffleNetV2

A performance of the ShuffleNetV2 classifier tested using raw PVD and enhanced PVD is depicted in terms of accuracy. It is shown that the accuracy of PDATFGAN-ShuffleNetV2 is increased by 2.17% and 0.15% compared to the ShuffleNetV2 using the raw dataset and DATFGAN-ShuffleNetV2, respectively as shown in Fig 4.20. This is achieved due to enhancing the leaf image resolutions based on the pixel's positions and minimizing the loss values.

Table 4.7 displays the results of a series of tests conducted on the DenseNet121 classifier model using the PVD raw dataset, the PVD augmented by the DATFGAN, and the PDATFGAN models.

Table 4.7 Comparison of the Proposed PDATFGAN Model Using DenseNet12

Performance Evaluation Metrics	Raw dataset	Dataset enhanced by DATFGAN	Dataset enhanced by PDATFGAN
Precision	0.8841	0.9249	0.9270
Recall	0.8843	0.9252	0.9273
F-measure	0.8842	0.9251	0.9272
Accuracy	88.47%	92.54%	92.74%

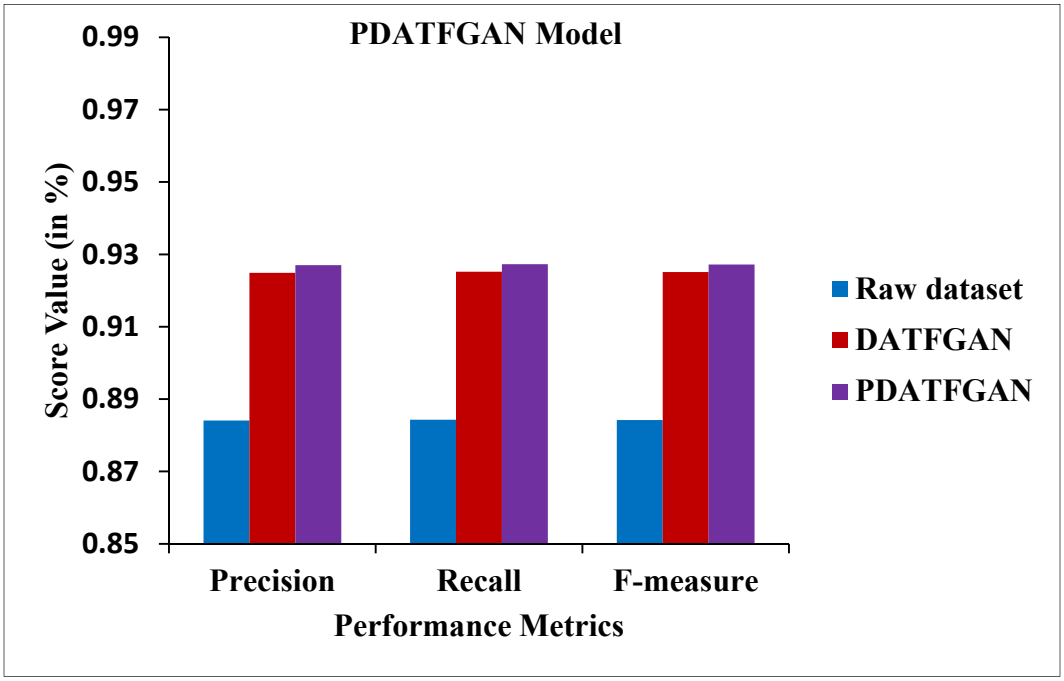

Fig. 4.21 Result of Proposed PDATFGAN Model Using DenseNet121

The precision, recall, and f-measure of several GAN models compared using the DenseNet121 classifier on the raw and improved PVD. DenseNet121 performs better on the PDATFGAN-enhanced PVD than on the raw dataset or the DATFGAN-enhanced

dataset, as shown in Fig. 4.21. PDATFGAN-DenseNet121 improves over DenseNet121 trained on the raw dataset and DATFGAN-DenseNet121 by around 4.86% and 0.23%, respectively, in terms of precision, recall, and f-measure.

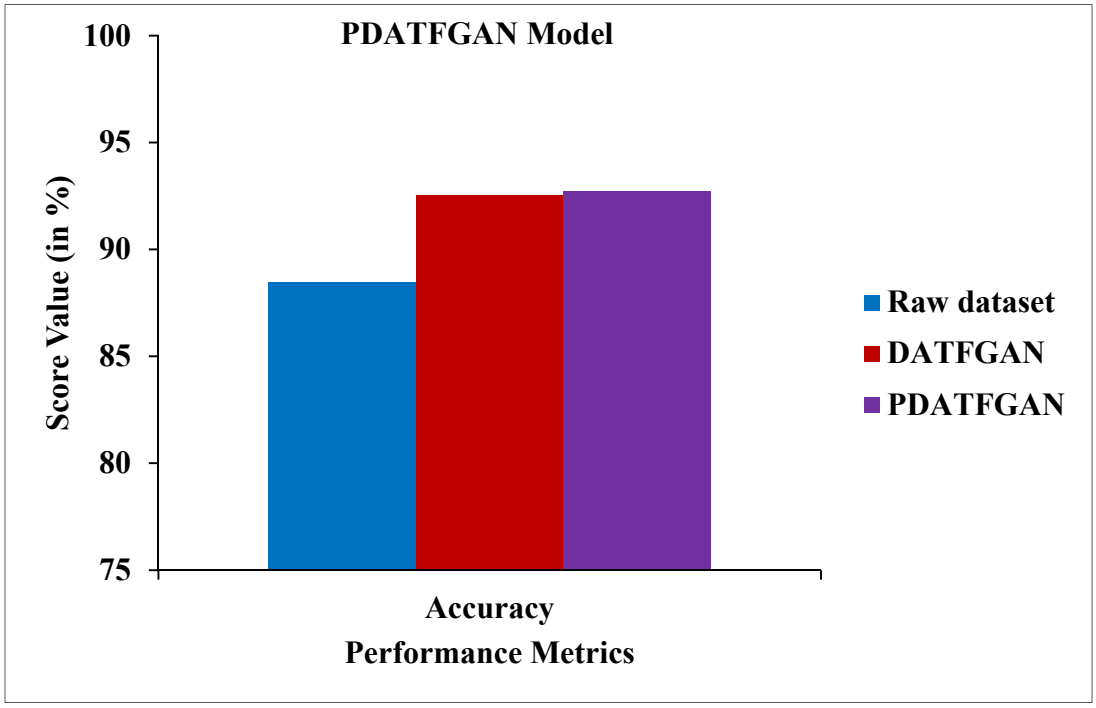

Fig. 4.22 Accuracy Comparison of PDATFGAN Model Using DenseNet121

A performance of the DenseNet121 classifier tested using raw PVD and enhanced PVD is depicted in terms of accuracy. It is noted that the accuracy of PDATFGAN-DenseNet121 is improved by 4.83% and 0.22% compared to the DenseNet121 using the raw dataset and DATFGAN-DenseNet121, respectively as shown in Fig 4.22. This is since generating high-resolution leaf images according to the spatial coordinate system i.e., knowledge about pixels' positions.

The test results for the MobileNetV2 classifier model tested using the PVD raw dataset, enhanced PVD by the DATFGAN and PDATFGAN models are given in Table 4.8.

Table 4.8 Comparison of the Proposed PDATFGAN Model Using MobileNetV2

Performance Evaluation Metrics	Raw dataset	Dataset enhanced by DATFGAN	Dataset enhanced by PDATFGAN
Precision	0.9062	0.9264	0.9283
Recall	0.9065	0.9266	0.9287
F-measure	0.9064	0.9265	0.9285
Accuracy	90.66%	92.69%	92.87%

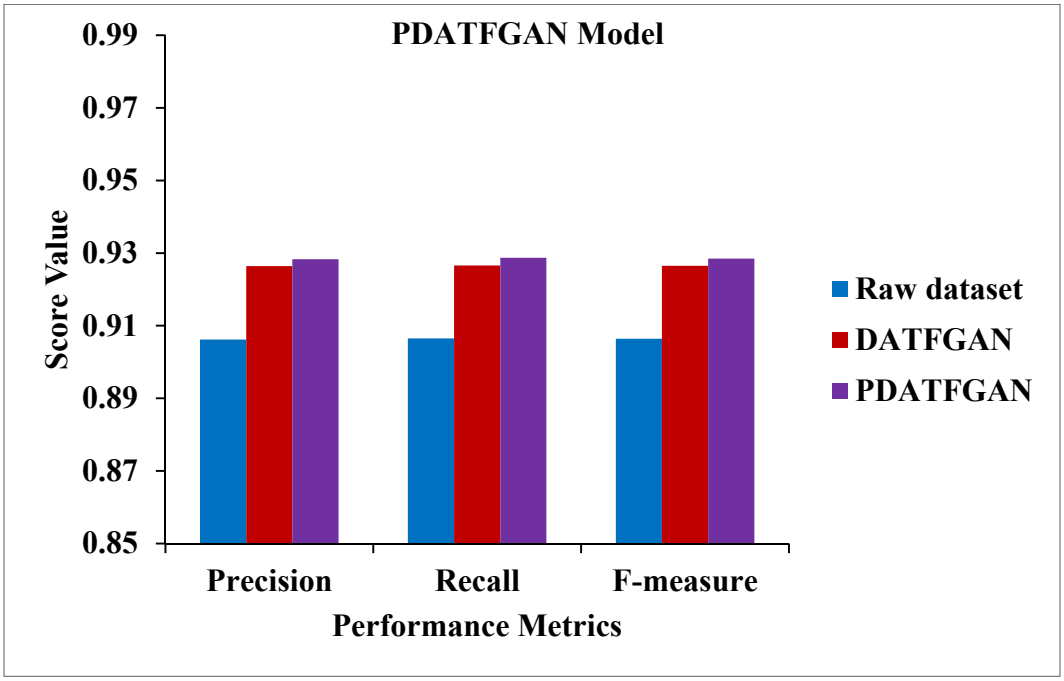

Fig 4.23 Result of Proposed PDATFGAN Model Using MobileNetV2

Precision, recall, and f-measure are compared for a number of different GAN models used by the MobileNetV2 classifier on both raw and enriched PVD. It is addressed that the MobileNetV2 implements well if utilizing the PVD enhanced by the PDATFGAN than the raw dataset and the dataset enhanced by the DATFGAN model. The precision of PDATFGAN-MobileNetV2 is increased up to 2.44% and 0.21% compared to the MobileNetV2 using raw dataset and DATFGAN-MobileNetV2, respectively as shown in Fig 4.23. The recall of PDATFGAN-MobileNetV2 is improved

by 2.45% and 0.23% compared to the MobileNetV2 using the raw dataset and DATFGAN-MobileNetV2, respectively. Also, the f-measure of PDATFGAN-MobileNetV2 is raised to 2.44% and 0.22% than the MobileNetV2 using raw dataset and DATFGAN-MobileNetV2, respectively.

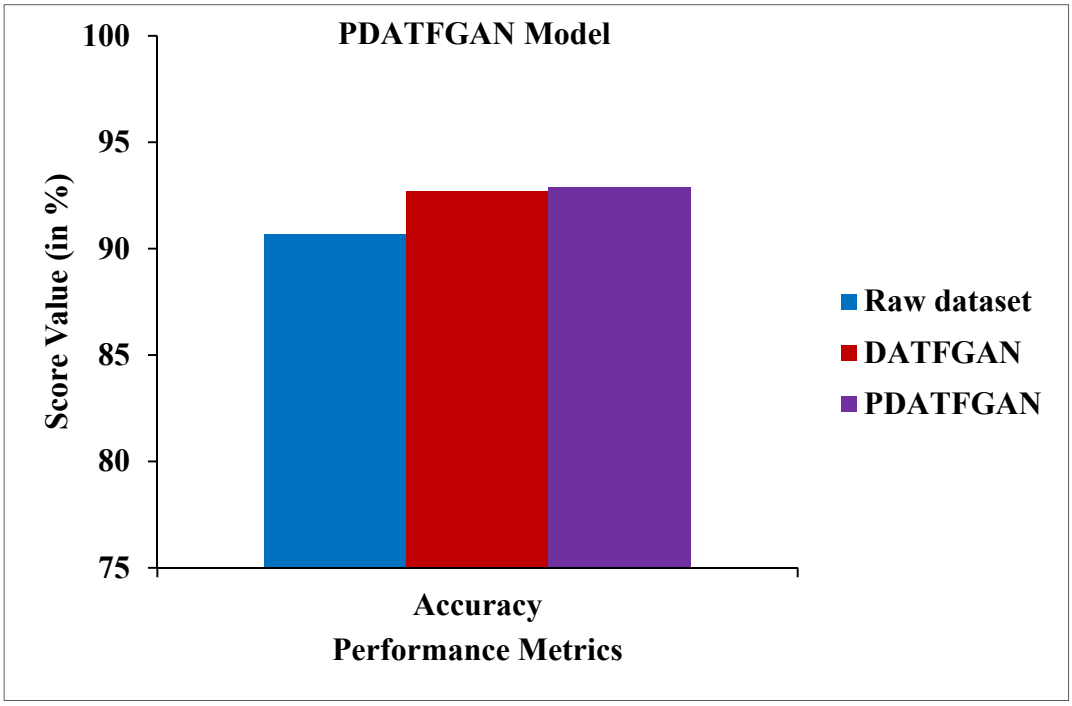

Fig 4.24 Accuracy Comparison of PDATFGAN Model Using MobileNetV2

A performance of the MobileNetV2 classifier tested using raw PVD and enhanced PVD is depicted in terms of accuracy. It is noted that the accuracy of PDATFGAN-MobileNetV2 is improved by 2.44% and 0.19% compared to the MobileNetV2 using the raw dataset and DATFGAN-MobileNetV2, respectively. This is owing to the development of positional-aware GAN for high-resolution leaf image creation, which supports the classifier model to effectively classify leaf diseases from better visual quality leaf images as shown in Fig 4.24

These side-by-side evaluations demonstrate that the MobileNetV2 classifier outperforms the competition on raw and improved PVD. When it comes to accurately classifying leaf diseases, the MobileNetV2 with the PDATFGAN model performs better than other methods.

4.6 SUMMARY

To summarize, an overview of leaf disease image classification and its challenges are discussed in this chapter. A detailed analysis of various techniques used for high-resolution leaf disease image generation and leaf disease classification is provided. The design and development of the PDATFGAN model using PGAN and different pre-trained DCNN classifiers is described. Form leaf images with healthy and various disease classes, the performance of the models are examined. Based to the results, leaf disease image classification could make use of the PDATFGAN model that has been developed.