SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I3P115

Volume 12 Issue 3, 151-160, March 2025

Original Article

Enhancing Sugarcane Disease Classification Using Transfer Learning with Convolutional Neural Networks

Meenakshi Thalor[1], Chinmay Nakwa[2], Sanjay Mate[3], Ashpana Shiralkar[4]

[1,2,4]*AISSMS Institute of Information Technology, Maharashtra, India.*
[3]*Department of Information Technology, Government Polytechnic, Daman, India.*

[1]*Corresponding Author : meenakshi.thalor@aissmsioit.org*

Received: 09 January 2025 Revised: 10 February 2025 Accepted: 14 March 2025 Published: 29 March 2025

Abstract *- The economic implications of sugarcane diseases on local farmers in India are significant and multifaceted, affecting not only their immediate yields but also their overall financial stability and livelihoods. About 70 percent of India's rural households still primarily depend on agriculture for their livelihood. As a cash crop, sugarcane holds a very important place in India's agrarian economy. India is not only the largest consumer of sugar but also its second-largest producer. Identifying the diseases in their initial stages helps not only the farmer but also reduces the burden on the country in many aspects. This paper discusses DenseNet, VGG, and ConvNeXt for classifying diseases in sugarcane plants, along with the detailed experimentation conducted. Based on evaluation metrics, ConvNeXt outperforms with 96% accuracy compared to DenseNet and VGG architectures on sugarcane disease detection.*

Keywords *- ConvNeXt, Deep Learning, DenseNet, Sugarcane dsisease, VGG.*

1. Introduction

India is the world's largest consumer and the second-largest producer of sugar. With a 15% share in global sugar consumption and around 20% of global sugar production, trends in India's sugar industry have a significant impact on global markets [1]. Sugarcane farmers in India receive the highest cane prices in the world. India offers its citizens the cheapest sugar despite the world's sugar prices reaching record levels.

The 10-year average of sugar production from the year 2014 to 2023 was 31.92 Million Metric Tons. The marketing year 2023/2024 saw a decline of 3 Million Metric Tons; the production stood at 34 Million Metric Tons [2]. The current year's production estimate has been revised downward to 34 MMT, or 32 MMT of crystal white sugar, due to delayed rainfall in Maharashtra and Karnataka and a red rot infestation in central Uttar Pradesh [3].

Around 50 million farmers and their families are estimated to be involved in sugarcane cultivation, while approximately 500,000 skilled and unskilled workers are employed in sugar mills and related industries. The sugar industry in India plays a crucial role in the socio-economic development of rural areas by mobilizing local resources, creating jobs, and boosting farm incomes [4]. Ensuring plant health and identifying diseases are essential for promoting sustainable agriculture. However, relying solely on manual methods for monitoring plant diseases presents significant challenges due to the complexity and time-intensive nature of the task. Implementing efficient and scalable solutions is vital to address these issues effectively. Plants are often affected by various diseases, and some of these diseases can destroy the whole crop if not diagnosed and treated in time [5].

The primary diseases responsible for sugarcane yield losses include wilt (infecting the vascular system), red rot (attacks the cane’s inner tissues), Pokkah Boeng (abnormal growth, including twisted stems and stunted plants), sugarcane Grassy Shoot (yellowing, stunting, and the death of shoots), and pineapple sett rot (causing rotting and decay) [6].

Major challenges associated with these diseases are many of these diseases exhibit symptoms only in later stages, making early detection difficult and limiting effective management; certain diseases, such as sugarcane Grassy Shoot, are spread by insect vectors, adding another layer of complexity to disease control; weather patterns, such as delayed or irregular rainfall, can exacerbate disease spread and affect crop health; over time, pathogens may develop resistance to chemical treatments, making management even more challenging; Disease outbreaks lead to significant financial losses for farmers, affecting not only yields but also the quality of sugarcane, which influences sugar production and prices.

Traditional disease detection methods, like laboratory analysis or visual evaluation, require a lot of time and qualified personnel. Due to environmental variations and the similar symptoms that various diseases exhibit, these methodologies are also prone to human error. The lack of specialists and diagnostic facilities in rural or resource-constrained areas makes prompt disease management even more difficult.

CNNs and Transformers can play a pivotal role in shielding agricultural industries. By leveraging these advanced models, farmers can identify signs of diseases at an early stage, allowing for timely interventions and reducing crop loss. The power of transfer learning is employed with the help of pre-trained CNNs.

This paper discusses DenseNet, VGG, and ConvNeXt regarding the classification of diseases in sugarcane plants, along with the detailed experimentation conducted. The approach combines advanced deep learning models with basic hyperparameter tuning and data augmentation to tackle the challenges of sugarcane disease detection. This helps create a reliable and effective system that enables timely interventions and minimizes crop loss.

2. Literature Review

The field of artificial intelligence has witnessed rapid advancements over the past decade, with significant contributions in the field of healthcare. Among these, diagnostic imaging has emerged as a turning point where AI applications show great promise. This literature review explores existing studies to identify advancements, challenges, and opportunities in AI-driven medical imaging.

In the study [7], the authors used a dataset consisting of 4 classes and a total of 1990 images of the sugarcane crop. They have used 4 pretrained neural network models comprising Alexnet, VGG19, Resnet18 and Densenet201. Due to the small amount of data, they used data augmentation to increase the model's performance. It was found that VGG19 outperformed all of the other models.

In the study [8], the authors have used two datasets: one is of rice leaves and the other is of potato leaves. The first one contains 5932 images, and the following one contains 1500 images. The datasets were split 80:20 for training and testing. Along with CNN, they also used Machine Learning Models consisting of SVM, KNN, Decision Tree and Random Forest. CNN had outperformed all of them. The CNN scored a total of 99.58 accuracy for the rice dataset and 97.66 accuracy for the leaf dataset.

In the study [9], the author researched tomato crops. The dataset consisted of 13,262 images. Utilized two pretrained models, Alexnet and VGG16 net. The dataset contained 7 classes. The study was divided into two parts. In the first part, augmented images were given to the models. The second part consisted of the number of modified images, and the hyperparameters were changed. Using AlexNet and VGG16, classification accuracies of 97.49% and 97.29% were achieved for the first part.

In the study [10], the authors have used classifiers on pretrained models. The pretrained models are Inception v3, VGG16, and VGG19, and the classifiers used on them are SVM, KNN, Neural Networks, Logistic Regression, AdaBoost, SGD and Naive Bayes. Their dataset consisted of a total of 240 images. The outcome of this experiment was that the VGG16 with SVM was chosen. SVM could classify sugarcane as diseased and non-diseased with the highest AUC, i.e., 90.2%.

In the study [11], the authors have used a dataset on sugarcane leaf disease consisting of 5 classes. For their experiment, their proposed model is stacking, an ML ensemble technique.

The paper is divided into two parts; in the first part, they compared all the pretrained models, and in the second, they used stacking. Their proposed model performed better in comparison to just using the pretrained models. It gave an accuracy of 0.8653.

This study capitalises on the benefits of transfer learning by harnessing the power of three pioneering CNN architectures (VGG, DenseNet, and ConvNeXt). These models, pre-trained on vast datasets such as ImageNet, bring a wealth of pre-learned feature representations to the table.

This strategic approach not only streamlines the training process but also significantly boosts model performance, outperforming the traditional method of building models from the ground up.

3. Dataset

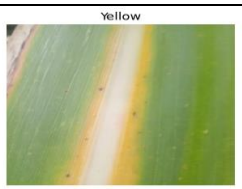
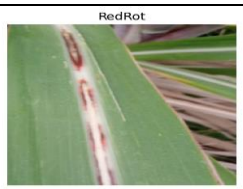
The dataset being used is publicly available [12]. It consists of 5 classes: Yellow, RedRot, Rust, Healthy and Mosaic, as shown in Table 1. A total of 2521 images are contained in the dataset.

4. Research Methodology

VGG, DenseNet, and ConvNeXt, three novel Convolutional Neural Network (CNN) architectures, were used in the experimental setup. Each one of them has marked an important milestone in the world of deep learning.

VGG is an example of the early CNN era, emphasizing depth and simplicity through the use of deep sequential layers and small convolution filters.

Table 1. Dataset distribution

Class	Mosaic	Rust	Yellow	RedRot	Healthy
Instances	462	514	505	518	522
Image					

DenseNet, which brought about a great paradigm shift with architectural design using dense connectivity based on feature reuse, is an example of the middle road taken while putting efficiency and gradient propagation forward. Finally, ConvNeXt concentrates on scalability and fine-grained performance optimization. It is a model that embodies the new wave that stems from developments in hierarchical design, influenced by the novel vision transformer. The experiment integrates three distinct architectures, each representing a different era in the evolution of computer vision. Transfer learning is a knowledge-sharing technique that reduces the volume of training data, training time, and processing costs for creating deep learning models.

Training CNN models from scratch is very computationally expensive and time-consuming, particularly for deep learning architectures with parameters reaching a count of millions. It requires a lot of labelled data to achieve broad applicability and significant computational resources, such as high-performance GPUs or TPUs, to handle the extensive matrix operations and backpropagation calculations. Moreover, the training process usually contains several iterations regarding hyperparameters and adjustments, such as data preprocessing and optimization, adding to cost and time. In those situations, when the size of the datasets is limited or computing resources are insufficient, training CNN from scratch becomes too expensive or even impossible. Diversions such as transfer learning with pre-trained models have gained vast relevance today.

Pre-trained models, such as those available in TensorFlow Applications, are trained on large benchmark datasets like ImageNet and provide a solidly built feature extraction backbone that can be fine-tuned for specific tasks. This reduces training time and computational cost while improving model performance by using the rich feature representations learned during pre-training. This research fine-tuned the last 10 layers of each model. This approach allowed the models to retain learned general representations while adapting to domain-specific features in sugarcane disease images. Experimental results showed that fine-tuned models consistently outperformed their frozen counterparts, highlighting the importance of training deeper layers for improved classification accuracy.

5. Deep Learning Model's

5.1. DenseNet

DenseNet, an acronym for Densely Connected Convolutional Networks, establishes feedforward connections between each layer and every other layer, as shown in Figure 1. In DenseNets architecture, dense blocks and transition blocks are repeatedly used to strengthen feature propagation and encourage feature reuse [13]. Each convolutional layer in a dense block generates a fixed number of output channels. This number is called the "growth rate."

The growth rate is a hyper-parameter that controls how many features each layer of a dense block adds and, therefore, directly affects memory and computational requirements. The growth rate is typically set to 32. A higher growth rate results in a larger number of feature maps at each layer, which can increase model capacity but also the number of parameters. Each dense block contains multiple layers, and each layer receives the output from all previous layers in that block.

This is a key characteristic that distinguishes DenseNet from other traditional architectures. This architecture can be seen in Figure 1. Transition layers are used to connect one dense block to another dense block. During the experiment, data augmentation on the dataset was utilized for DenseNet. The images are flipped horizontally, rotated, zoomed, and rescaled (1/.255), and the height and width are also adjusted.

In this paper, the 3 different versions of DenseNet are as follows:

1) DenseNet121: 'BatchNormalization': 121, 'Activation': 121, 'Conv2D': 120, 'Concatenate': 58, 'AveragePooling2D': 3, 'ZeroPadding2D': 2, 'InputLayer': 1, 'MaxPooling2D': 1
2) DenseNet169: 'BatchNormalization': 169, 'Activation': 169, 'Conv2D': 168, 'Concatenate': 82, 'AveragePooling2D': 3, 'ZeroPadding2D': 2, 'InputLayer': 1, 'MaxPooling2D': 1
3) DenseNet201: 'BatchNormalization': 201, 'Activation': 201, 'Conv2D': 200, 'Concatenate': 98, 'AveragePooling2D': 3, 'ZeroPadding2D': 2, 'InputLayer': 1, 'MaxPooling2D': 1

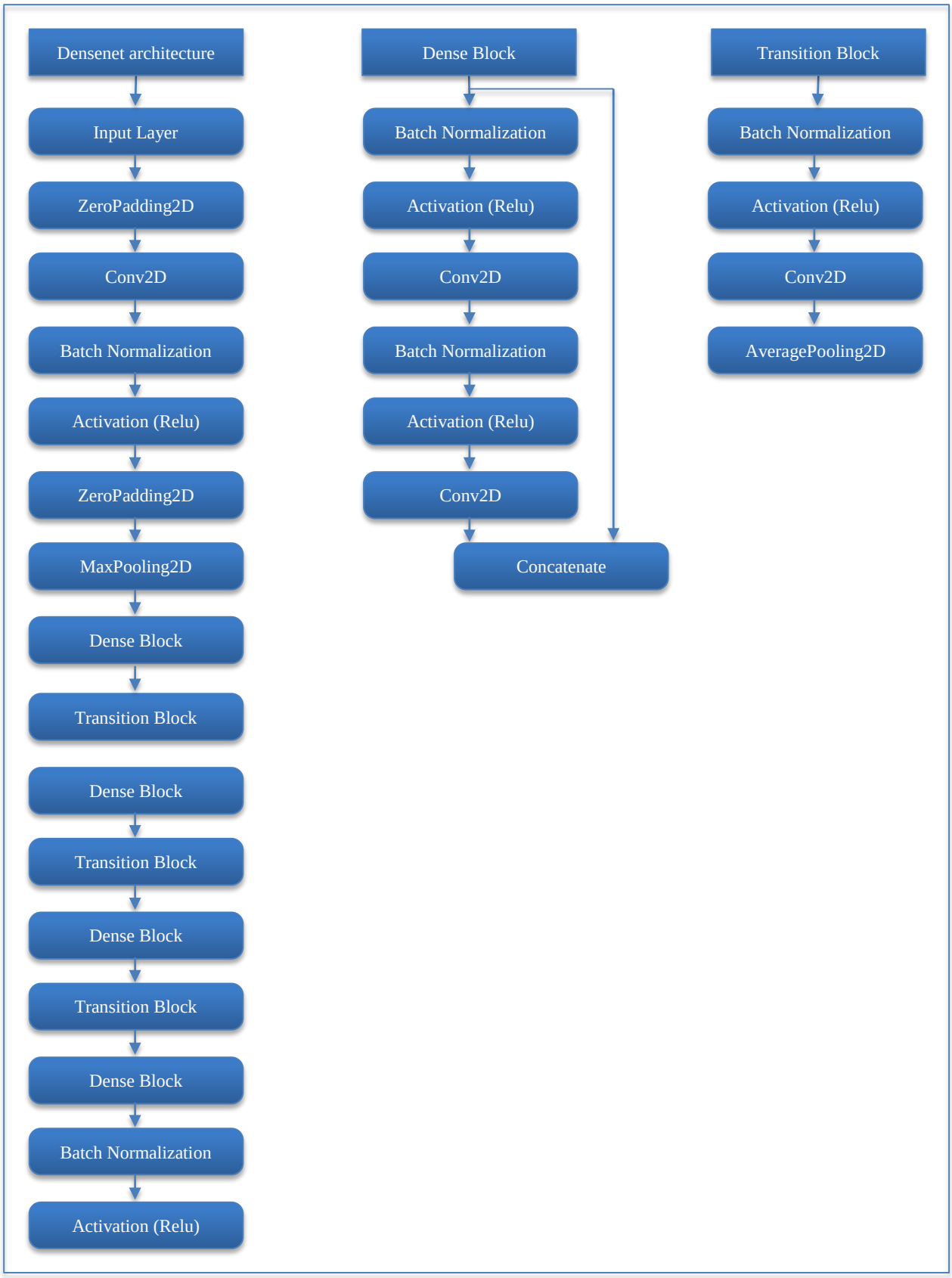


Fig. 1 DenseNet architecture

Table 2. Details of DenseNet architectures

Variants	Layers	Dense Block Configuration	Parameters (Pretrained)	Size (MB)
DenseNet121	427	6-12-24-16	8.1M	33
DenseNet169	595	6-12-32-32	14.3M	57
DenseNet201	707	6-12-48-32	20.2M	80

The explanation of each phase of DenseNet used are:

1) BatchNormalization is applied after each convolutional operation to normalize the feature maps.
2) Activation layers introduce non-linearity and help the network learn complex patterns.
3) Conv2D is responsible for learning spatial hierarchies in the data (e.g., edges, textures). Each convolution is followed by a batch normalization layer and an activation function.
4) Concatenate layers combine the feature maps from different layers in the network.
5) Average Pooling is a down-sampling operation that reduces the size of the feature maps after certain blocks to make the computation more manageable as the model progresses through the layers.
6) Zero Padding is used to add extra pixels (usually zero) around the border of the feature maps to maintain spatial dimensions.
7) The Input Layer is where the data enters and specifies the shape of the input.
8) MaxPooling2D is another down-sampling operation that reduces the spatial dimensions of the feature maps by selecting the maximum value from a defined region.

Input layer reshaping the images to (224, 224, 3) and a data augmentation layer have been added before the pretrained model. GlobalAveragePooling2D and Dense Layer have been added after the pretrained model. The last 10 layers of each model are unfrozen to achieve greater accuracy. The epochs being used are 10. The details of DenseNet architectures are shown in Table 2.

5.2. VGG

It laid the foundation for deep learning in computer vision. VGG stands for Visual Geometry Group. VGG uses a simple design philosophy, stacking small 3x3 convolutional filters with ReLU activations and a stride of 1, followed by max-pooling layers (2x2 filters with a stride of 2) to gradually reduce spatial dimensions while increasing feature complexity. VGG provides better results while having smaller convolutional layers since it doesn't rely on a large number of hyper-parameters [14].

VGG demonstrated that increasing network depth significantly improves performance. This architecture can be seen in Figure 2. Before input, the image needs to be pre-processed to convert RGB to BGR. Input layer reshaping the images to (224, 224, 3) and a data augmentation layer have been added before the pretrained model. GlobalAveragePooling2D and Dense Layer have been added after the pretrained model. The details of VGG architectures are shown in Table 3.

Data augmentation on the dataset is utilized for the VGG during the experiment. The images are flipped horizontally, rotated, and zoomed, and height and width are adjusted.

In this paper, the 2 different versions of VGG used are:

1) VGG16:'Conv2D': 13, 'MaxPooling2D': 5, 'InputLayer': 1
2) VGG19:'Conv2D': 16, 'MaxPooling2D': 5, 'InputLayer': 1

The explanation of each phase of VGG is as follows:

1) Conv2D stands for the 2D convolutional layers in the network. These layers extract the input images' features like edges, textures, shapes, etc.. These layers are arranged in blocks to learn hierarchical representations.
2) MaxPooling2D layers are used to down sample the feature maps after the convolutional operations. These layers reduce the spatial size (height and width) of the feature maps, helping to control overfitting and reduce computation.
3) The InputLayer is where the image data enters the network. This layer specifies the shape and format of the input data.

5.3. ConvNeXt

ConvNeXt divides the network into stages, similar to Transformers, gradually increasing the number of channels while downsampling the spatial dimensions, as shown in Figure 3. The core component is the ConvNeXt block that includes depthwise convolution and pointwise convolution, bringing better efficiency and a module named LayerScale applied on each ConvNeXt block which adds a learnable scale for the output of each block. ConvNeXT is constructed entirely from standard ConvNet modules [15]. The inputs of ConvNeXt models are expected to be float or uint8 tensors of pixels with values between 0 and 255. Input layer reshaping the images to (224, 224, 3) has been added before the pretrained model. GlobalAveragePooling2D and Dense Layer have been added after the pretrained model. The last 10 layers of each model are unfrozen to achieve greater accuracy. The number of epochs being used is 10. The details of ConvNeXt architectures are shown in Table 4. In this paper, the 5 different versions of ConvNeXt used are:

1) ConvNeXtBase: 'Dense': 72, 'Activation': 72, 'LayerNormalization': 37, 'Conv2D': 36, 'LayerScale': 36, 'Sequential': 4, 'InputLayer': 1, 'Normalization': 1
2) ConvNeXtLarge: 'Dense': 72, 'Activation': 72, 'LayerNormalization': 37, 'Conv2D': 36, 'LayerScale': 36, 'Sequential': 4, 'InputLayer': 1, 'Normalization': 1
3) ConvNeXtSmall: 'Dense': 72, 'Activation': 72, 'LayerNormalization': 37, 'Conv2D': 36, 'LayerScale': 36, 'Sequential': 4, 'InputLayer': 1, 'Normalization': 1
4) ConvNeXtTiny: 'Dense': 36, 'Activation': 36, 'LayerNormalization': 19, 'Conv2D': 18, 'LayerScale': 18, 'Sequential': 4, 'InputLayer': 1, 'Normalization': 1
5) ConvNeXtLarge: 'Dense': 72, 'Activation': 72, 'LayerNormalization': 37, 'Conv2D': 36, 'LayerScale': 36, 'Sequential': 4, 'InputLayer': 1, 'Normalization': 1

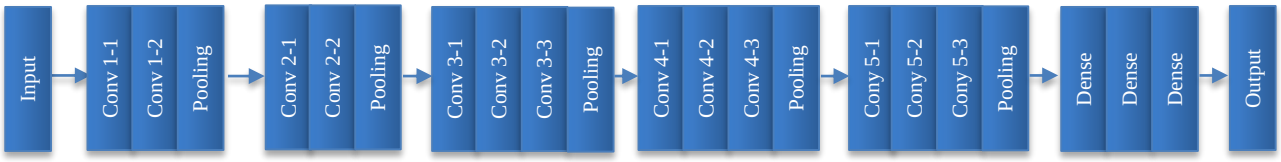


Fig. 2 VGG architectures

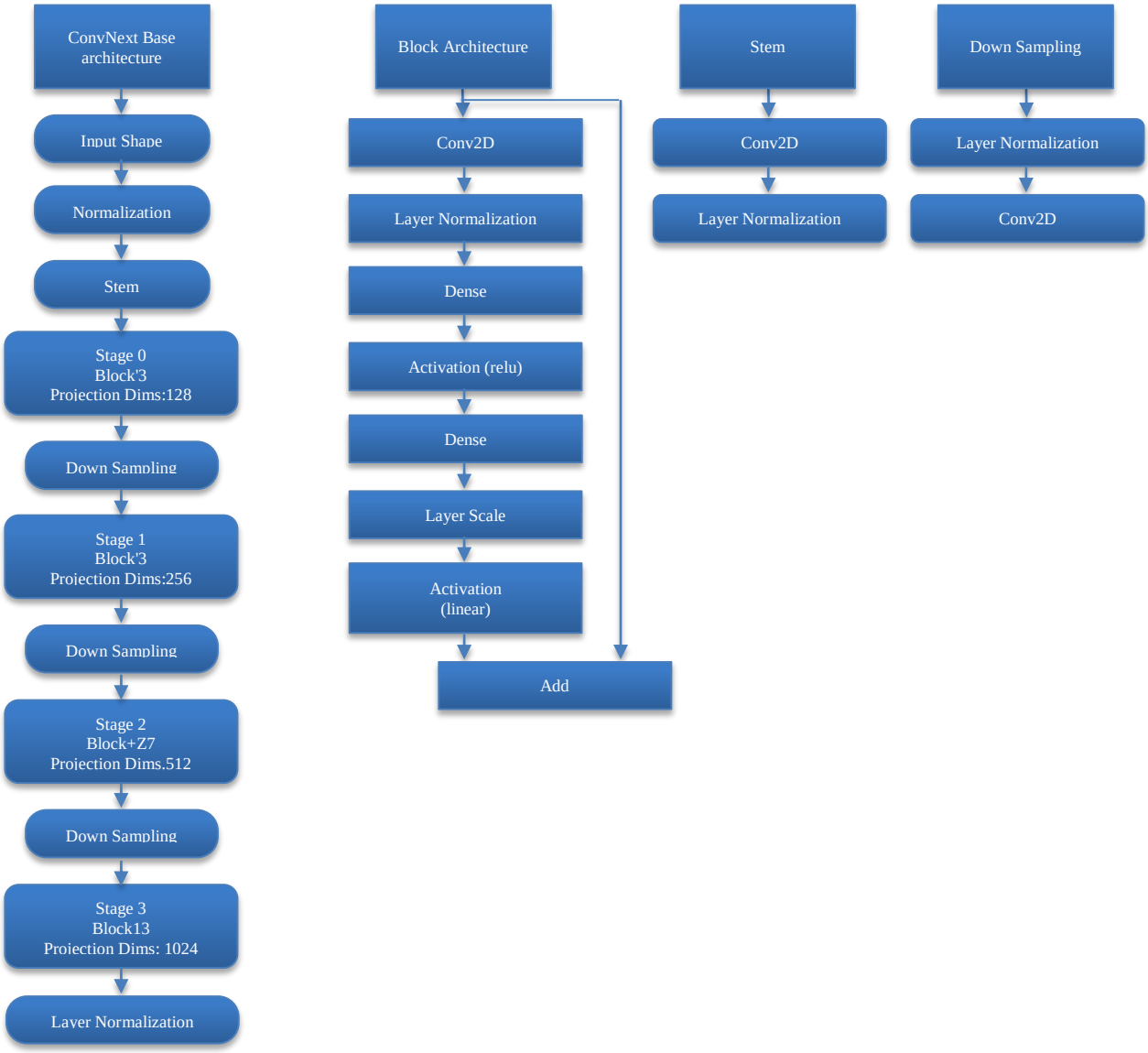


Fig. 3 ConvNeXt architectures

Table 3. Details of VGG architectures

Variants	Conv2D Layers	MaxPooling2D	Input Layer	Total Layers	Parameters	Pretrained On	Size(MB)
VGG16	13	5	1	19	138.4M	ImageNet	528
VGG19	16	5	1	22	143.7M	ImageNet	549

Table 4. Details of ConvNeXt architectures

Variants	Layers	Parameters	Pretrained on	Size (MB)	Block Configuration	Projection dims (per Block)
ConvNeXtTiny	133	28.6M	ImageNet	109.42	3, 3, 9, 3	96, 192, 384, 768
ConvNeXtSmall	259	50.2M	ImageNet	192.29	3, 3, 27, 3	96, 192, 384, 768
ConvNeXtBase	259	88.5M	ImageNet	338.58	3, 3, 27, 3	128, 256, 512, 1024
ConvNeXtLarge	259	197.7M	ImageNet	755.07	3, 3, 27, 3	192, 384, 768, 1536
ConvNeXtXLarge	259	350.1M	ImageNet	1310	3, 3, 27, 3	256, 512, 1024, 2048

The explanation of each phase of ConvNeXt is as follows:

1) Dense layers in ConvNeXt are fully connected layers and help combine the features extracted by the earlier convolutional layers to make high-level decisions about the input data.
2) Activation layers introduce non-linearity by an activation function such as ReLU or GELU, which helps the network learn more complex relationships between features.
3) Layer Normalization is a practice used to normalize each layer's inputs, ensuring that the activations remain centered around zero with a standard deviation of one. It is applied at various points in the model to ensure smooth training by lessening internal covariate shift Conv2D layers perform convolution operations to extract meaningful features from the input images, forming the basis of the network's learning.
4) Conv2D layers perform convolution operations to extract meaningful features from the input images, forming the basis of the network's learning.
5) LayerScale is a technique used to scale the outputs of certain layers to maintain numerical stability and improve training. It allows the model to adjust the importance of each layer during training. This method helps fine-tune the contributions of each layer to the network's overall performance.
6) Sequential is a container for linearly stacking layers. Each block may consist of multiple types of layers (e.g., Conv2D, Dense, LayerNormalization), and they work together to progressively extract and refine features.
7) The InputLayer is where the data (typically an image) enters the network. This layer doesn't perform any computation but simply specifies the shape and format of the input data.
8) Normalization can refer to various forms of normalizing the data (e.g., batch normalization, layer normalization, or other forms) to standardize the inputs to a network layer.

6. Evaluation Measures

Evaluation metrics are frequently used to discover how well machine learning models achieve, particularly regarding classification problems. The confusion matrix for all disease classes is taken into consideration, as shown in Figure 4. While precision and recall focus on particular areas, like the accuracy of positive predictions and an ability to recognize many important cases, accuracy provides a broad picture of the model's overall correctness.

		Predicted Outcome	
		Positive	Negative
Actual Outcome	Positive	TP	FN
	Negative	FP	TN

Fig. 4 Confusion matrix

6.1. Accuracy

The proportion of correctly classified occurrences (True Positive and True Negative) to the total number of occurrences.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (1)$$

6.2. Precision

The proportion of correctly classified correct positive occurrences (True Positive) to the total number of positive occurrences.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

6.3. Recall

Also known as Sensitivity. The proportion of correctly classified correct positive occurrence (True Positive) to the sum of true positive and false negative instances.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

7. Experimental Results

During the implementation of the system, different CNN architectures, such as DenseNet, VGG, and ConvNeXt, are employed. This paper leverages pretrained models from TensorFlow and Keras. The dataset was split 80:20 between train and test. Categorical Crossentropy as loss function and Adam optimizer are used during experimentation in all models. During the implementation of this work, Nvidia T4 provided by Google Colaboratory was used to compile all the models. To summarize the whole experimental set, a dataset consisting of 2521 instances and 5 classes is used. The data converted to the shape of (224, 224, 3) was made into batches of 32, and they were shuffled for the training dataset and not for the validation dataset. The pretrained models were taken from TensorFlow. An augmentation layer was applied for the DenseNet and VGG. The last 10 layers were unfrozen for ConvNeXt and DenseNet models to achieve higher accuracy. The models were fit for a total of 10 epochs. The analysis encompasses both traditional architectures like VGG variants, DenseNet variants, and modern ConvNeXt models, which are evaluated across multiple performance metrics. Figures 4 and 5 show the train data and test data evaluation results for all CNN models taken into consideration.

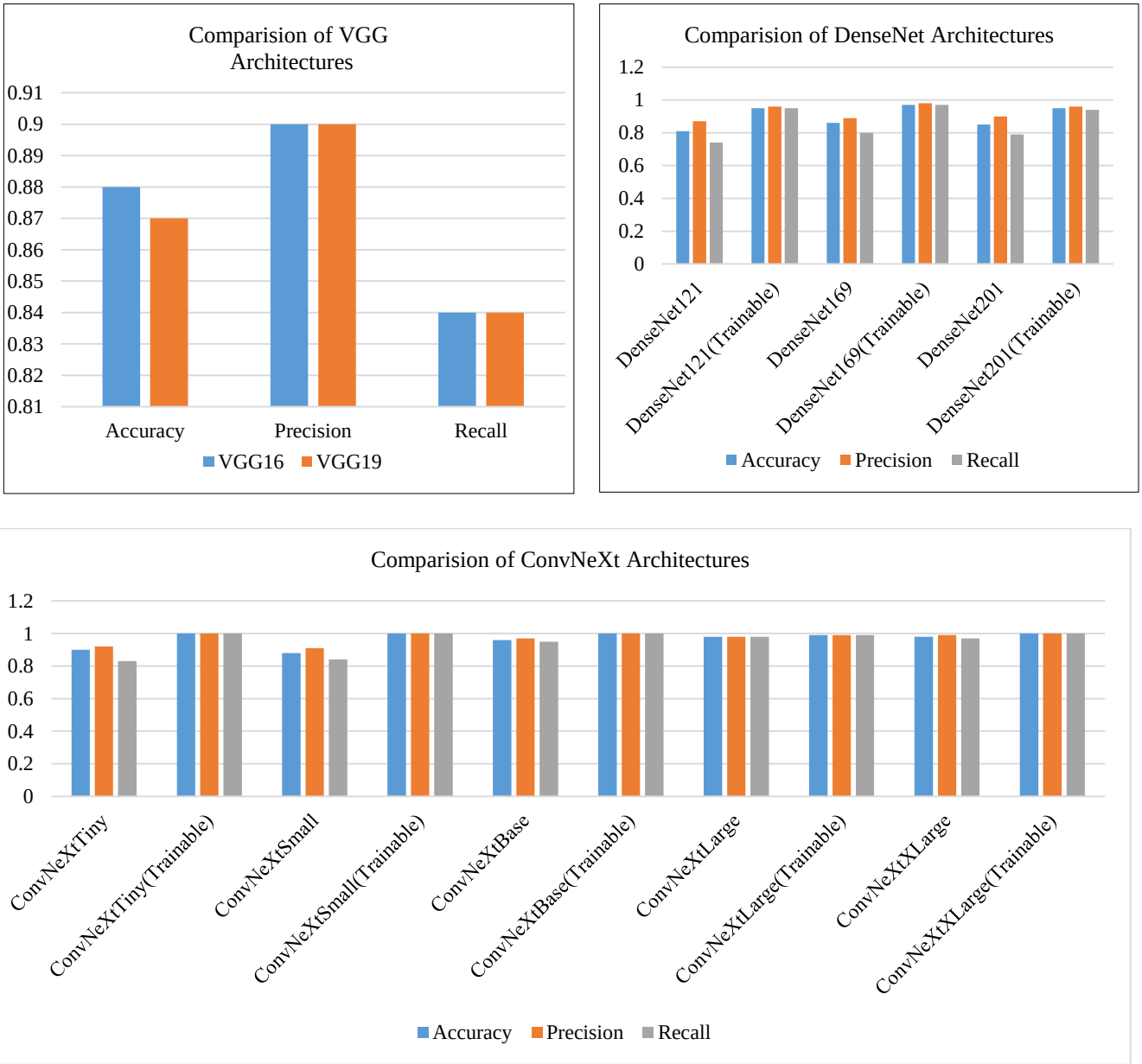


Fig. 5 Results on Train Data

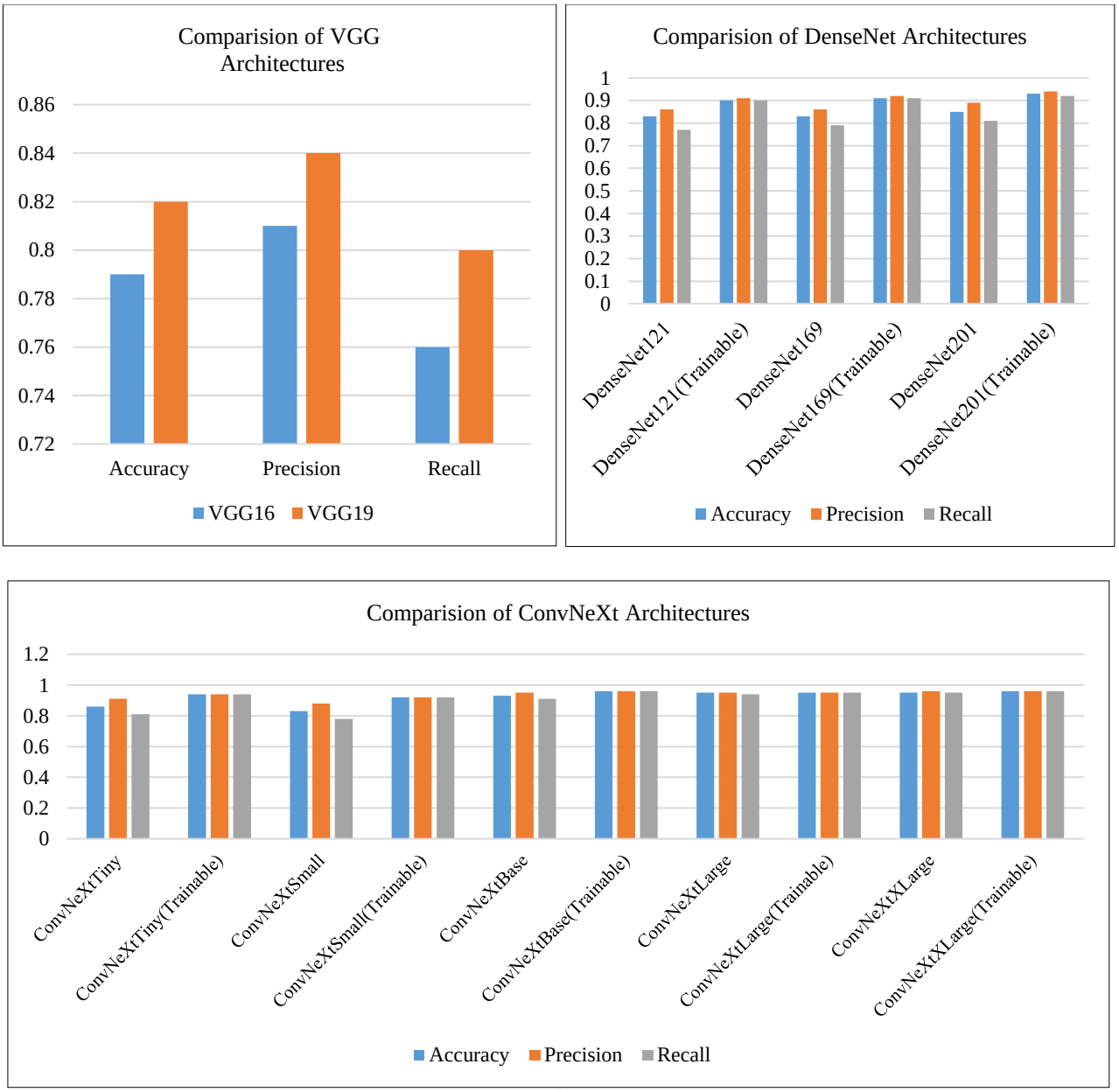


Fig. 6 Results of validation data

The ConvNeXt family of models generally outperforms the older architectures (VGG and DenseNet). Making models trainable (fine-tuning) consistently improves performance across all architectures. Fine-tuned models consistently outperform their frozen counterparts, with improvements in validation accuracy. While several ConvNeXt variants achieve perfect training accuracy (1.0), their validation performance suggests moderate overfitting.

Fine-tuning substantially improved performance across all architectures, with the most significant gains observed in the DenseNet variants. The generalization gap between training and validation metrics provides crucial insights; while ConvNeXt variants achieved perfect or near-perfect training metrics (1.00) when trainable, their validation performance showed realistic degradation.

8. Conclusion

Sugarcane is a vital crop for the global economy, contributing significantly to sugar production and bioenergy. However, diseases in sugarcane can lead to severe yield losses, affecting both farmers and industries dependent on this crop. With a focus on sugarcane disease classification, this paper discusses the use of deep learning in agriculture. In this paper, three categories of deep learning, VGG, DenseNet, and ConvNeXt, are evaluated. The verdict of this study aims to assist sustainable agriculture by enabling precision farming practices, reducing crop losses, and promoting food security.

For future work, a bigger dataset could be used. The hyperparameter of said models can be optimized using nature-inspired search and optimization algorithms. Transformers like Vision Transformer and Swin Transformer could be used to better understand what classification is suitable, such as CNN or Transformer.

Acknowledgement

We extend our sincere thanks to everyone who directly or indirectly played a role in this work. A special appreciation goes to the AISSMS Institute of Information Technology for providing an experimental setup environment.

References

[1] India becomes Chair of International Sugar Organisation (ISO) for 2024 to lead Global Sugar Sector, Ministry of Consumer Affairs, Food & Public Distribution, 2023. [Online]. Available: https://pib.gov.in/PressReleaseIframePage.aspx?PRID=1979507#:~:text=In%20its%2063rd%20council,of%20the%20organisation%20for%202024.

[2] Production - Sugar, United States Department of Agriculture Service, 2024. [Online]. Available: https://www.fas.usda.gov/data/production/commodity/0612000

[3] Shilpita Das, and Joanna Brown, "*Sugar Annual*," United States Department of Agriculture Service, Report, pp. 1-14, 2024, [Publisher Link]

[4] "*Sugarcane Brief Note*," National Food Security Mission, Report, pp. 1-11, 2016. [Publisher Link]

[5] Arpan Kumar, and Anamika Tiwari, "Detection of Sugarcane Disease and Classification using Image Processing," *International Journal for Research in Applied Science & Engineering Technology*, vol. 7, no. 5, pp. 2023-2030, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] J. Jernisha et al., "Plant Growth Promoting Microorganisms and Emerging Biotechnological Approaches for Sugarcane Disease Management," *Journal of Pure and Applied Microbiology*, vol. 8, no. 4, pp. 2205-2217, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[7] A. Vivekreddy et al., "Artificial Intelligence Framework for Multiclass Sugarcane Leaf Diseases Classification Using Deep Learning Algorithms," *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 10, pp. 4277-5290, 2024. [Google Scholar] [Publisher Link]

[8] Rahul Sharma et al., "Plant Disease Diagnosis and Image Classification Using Deep Learning," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 2125-2140, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[9] Aravind Krishnaswamy Rangarajan, Raja Purushothaman, and Aniirudh Ramesh, "Tomato Crop Disease Classification using Pre-Trained Deep Learning Algorithm," *Procedia Computer Science*, vol. 133, pp. 1040-1047, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[10] Sakshi Srivastava et al., "A Novel Deep Learning Framework Approach for Sugarcane Disease Detection," *SN Computer Science*, vol. 1, pp. 1-7, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Swapnil Dadabhau Daphal, and Sanjay M. Koli, "Enhancing Sugarcane Disease Classification with Ensemble Deep Learning: A Comparative Study with Transfer Learning Techniques," *Heliyon*, vol. 9, no. 8, pp. 1-19. 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Nirmal Sankalana, Swapnil Daphal, and Sanjay Koli, Sugarcane Leaf Disease Dataset, Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/nirmalsankalana/sugarcane-leaf-disease-dataset/

[13] Gao Huang et al., "Densely Connected Convolutional Networks," *arXiv Preprint*, pp. 1-9, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[14] Anwar Abdullah Alatawi et al., "Plant Disease Detection using AI based VGG-16 Model," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, pp. 718-727, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Zhuang Liu et al., "A ConvNet for the 2020s," *arXiv Preprint*, pp. 1-15, 2022. [CrossRef] [Google Scholar] [Publisher Link]