International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online))Volume 4, Issue 6, June 2015)

# Deep Fake Face Detection using Deep Learning based DenseNet 121 Architecture

Yashkeerti Baderiya[1], Prof. Adarsh Raushan[2], Dr. Sadhna K Mishra[3]
[1]*M. Tech. Scholar, Department of Computer Science and Engineering, LNCT, Bhopal*
[2]*Assistant Professor, Department of Computer Science and Engineering, LNCT, Bhopal*
[3]*Head of Dept., Department of Computer Science and Engineering, LNCT, Bhopal*

**Abstract: Deepfake is an advanced synthetic media technology that can generate deceptively authentic yet forged images and videos by modifying a person's likeliness. The term "Deepfake" is a portmanteau of "Deep learning" and "Fake," which reflects the utilization of artificial intelligence and deep learning algorithms in creating deepfake. The deepfake generation involved training to learn the nuances of facial attributes, facial expressions, motion movement, and speech patterns to produce fabricated media that are indistinguishable from the actual footage. Recently, deep learning has significantly improved the accuracy of image classification and object detection systems. In this study, we used convolutional neural network (CNN)-based pre-trained models to effectively identify plant diseases. We fine-tuned the hyperparameters of popular pre-trained models, including DenseNet-121. The deepfake dataset consists of all 70k REAL faces from the Flickr dataset collected by Nvidia, as well as 70k fake faces sampled from the 1 Million FAKE faces (generated by StyleGAN) that was provided by Bojan.**

Keywords: **Deep learning advancements, image classification accuracy, object detection improvements, convolutional neural networks (CNN), DenseNet121**

## I. INTRODUCTION

Visual aids are commonly utilized across various industries such as law, medicine, and entertainment [1, 2]. However, the extensive usage of visual media also presents a risk of misuse. Media forgery has been prevalent in digital culture for a while, where software tools like Photoshop are used for manual manipulation of media content. With the recent advancements in Computer Vision (CV) and Machine Learning (ML) technologies, media forgery has become more accessible and widespread.

In 2012, the field of CV experienced a significant breakthrough when AlexNet, an AI model developed by Alex, outperformed other models in the image recognition challenge by a large margin. Since then, AlexNet, which is classic convolutional neural network architecture, has been instrumental in many CV applications. Another leap forward in CV research was made in 2014 when Goodfellow introduced the Generative Adversarial Network (GAN). GAN enables the creation of realistic-looking images from scratch without human intervention or manual editing.

The rapid evolution of hardware that supports artificial neural network models' training has catalyzed the growth of deep learning. In 2017, a novel deep learning-based media forgery algorithm called 'Deepfake' emerged and wreaked havoc, threatening society's security and privacy. Deepfake is a synthetic technique that replaces the person in an existing image or video with someone else's likeness or characteristics. It is a portmanteau of 'deep learning' and 'fake'. It originated from an anonymous individual under the pseudonym 'deepfake' who uploaded numerous pornographic videos to the Reddit website. The actresses' faces in the videos were swapped with those of other celebrities using deep learning [3, 4]. Figure 1 outlines the examples of deepfake based on different generation methods. Based on the figure,

- Puppet Master refers to the transfer of motion movement by synthesizing the motion of the source and regenerating onto the target output [5];
- Face Swapping involves swapping the facial regions between two people from one to another [6];
- Involve facial reenactment where Neural Textures focuses on deferred neural rendering to integrate neural textures in the parametric vectors for facial synthesis [7] while Face2Face uses GAN, such as CycleGAN and Star- GAN to achieve the synthesis output [8];

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online))Volume 4, Issue 6, June 2015)

- Present entire face synthesis to produce non-exist human outputs by the training on the different source data to capture their significant facial characteristics [9, 10];
- Leverages GAN's capability to modify certain facial attributes on target.

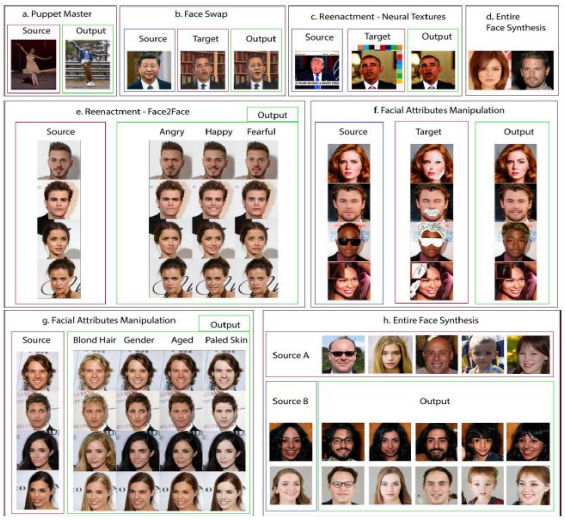



Figure 1: Examples of Deepfake

## II. CONVOLUTIONAL NEURAL NETWORK

CNN models excel at object recognition and classification when working with image databases. However, they come with challenges, including lengthy training times and the need for large datasets. Deep CNN models are necessary to extract both low-level and complex features from images, which further complicates the training process. Transfer learning offers a solution to these issues by utilizing pre-trained networks, allowing the parameters learned on one dataset to be applied to different problems. In this section, we explore the methodologies employed in this work.

Plant disease datasets consist of numerous images of both infected and healthy plant samples, with each sample assigned to a specific class. For example, if we consider banana plants as a class, all images of healthy and diseased banana plants are mapped to that class. Classification of a target image relies on the features extracted from the source images. In the case of banana plants, the class includes four types of diseases: xanthomonas wilt, fusarium wilt, bunchy top virus, and black sigatoka. When a sample of one particular disease is used as input, and the model has been trained with all four disease samples under the banana class, the output during the testing phase will accurately classify the specific disease label among the four categories within that class.

In multi-class classification, each category is mutually exclusive, meaning that each sample is assigned to only one category within the class. In contrast, multi-label classification treats each category within a class as a separate class. If there are N classes, we refer to N multi-classes. If each of these N classes contains M categories, then each category within each of the N classes is considered its own class.

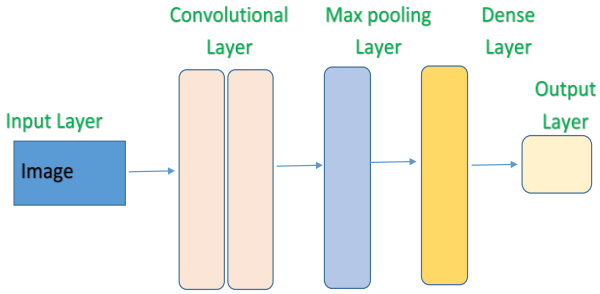



Fig. 2: CNN Architecture

### 2.1 DenseNet121

Densely Connected Convolutional Network, is a type of convolutional neural network (CNN) that introduces dense connections between layers. Each layer receives input from all preceding layers, which encourages feature reuse and mitigates the vanishing gradient problem. DenseNet121, in particular, has 121 layers and is composed of several building blocks, including dense blocks, transition layers, and convolutional layers. DenseNet121's architecture encourages feature reuse, which reduces the number of parameters and mitigates the vanishing gradient problem, leading to improved training efficiency and accuracy. Dense connections allow the model to learn more complex features and improve gradient flow through the network, which is particularly beneficial for deep architectures. DenseNet121 is a powerful and efficient deep learning model designed to tackle complex image recognition tasks by leveraging dense connectivity patterns, ensuring robust feature propagation and gradient flow throughout the network.

Initial Convolution Layer

The network starts with a convolutional layer that performs a $7 \times 7$ convolution with 64 filters, followed by a $3 \times 3$ max pooling operation with a stride of 2.

Mathematically, this can be expressed as:

$$X_1 = ReLU\left(BatchNorm\left(Conv2D(X_0, 64, 7 \times 7, stride = 2)\right)\right)$$

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online))Volume 4, Issue 6, June 2015)

Where $X_0$ is the input image, $X_1$ is the output of the first convolutional layer and Conv2D denotes a 2D convolution.
Dense blocks are the core components of DenseNet. Each dense block consists of multiple layers where each layer receives input from all previous layers within the block. If a dense block has L layers, the $l-$th layer receives feature maps from all preceding layers $\{0, 1, \ldots, l-1\}$.

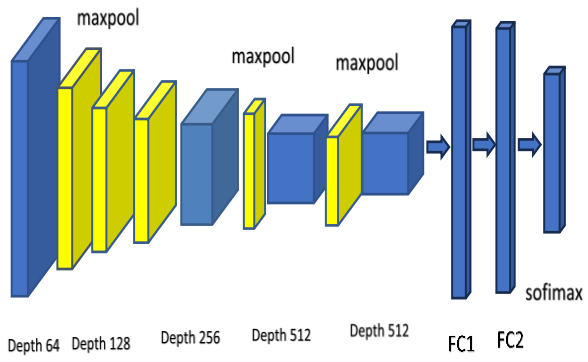



Fig. 3; DenseNet 121 Architecture

For the $l$-th layer in a dense block:

$$X_l = H_l([X_0, X_1, \ldots, X_{l-1}])$$

Where $H_l$ is a composite function of batch normalization (BN), ReLU, and Convolution (Conv). Specifically:

$$H_l(X) = Conv\ (ReLU(BatchNorm(X)))$$

Each layer produces $k$ feature maps, where $k$ is the growth rate.
Between dense blocks, transition layers are used to control the complexity of the model and reduce the size of the feature maps. A transition layer consists of a 1x1 convolution followed by a 2x2 average pooling with a stride of 2.

$$X_{transition} = AvgPool2D(Conv2D(X_{input}, filters = \theta \cdot filters_{input}, 1 \times 1), stride = 2)$$

Where $\theta$ is the compression factor, typically set to 0.5, reducing the number of feature maps by half.

## III. PROPOSED METHODOLOGY

The proposed hybrid deep learning model that combines DenseNet-121 and a custom Convolutional Neural Network (CNN) is to accurately identify plant diseases from the Plant Village dataset.

Step 1: The input to the model is an image tensor $X$ with dimensions 256×256×3, as per the PlantVillage dataset. Detail of dataset is given in Table 1.

$$X \in \mathbb{R}^{256\times256\times3}$$

Step 2: DenseNet-121 Base Layers: Apply the DenseNet-121 model (pre-trained on ImageNet) to the input image. Exclude the top classification layers.

$$F_{DenseNet} = DenseNet121(X, include_{top} = False)$$

Here, $F_{DenseNet} \in \mathbb{R}^{8\times8\times1024}$ (adjusted for the input size)

Step 3: Global Average Pooling
Apply global average pooling to the feature map to reduce its dimensions.

$$g = GlobalAveragePooling2D(F_{DenseNet})$$

The resulting vector $g \in \mathbb{R}^{1024}$
Step 4: The same input image tensor $X$ is used as input to the custom CNN $X \in \mathbb{R}^{256\times256\times3}$
Step 5: Apply a convolutional layer with 32 filters of size $3 \times 3$, followed by a ReLU activation function.

$$C_1 = ReLU\left(Conv2D(32, (3,3))(X)\right)$$

This operation results in an output tensor $C_1 \in \mathbb{R}^{254\times254\times32}$
Step 6: Apply max pooling with a $2 \times 2$ window to reduce the spatial dimension.

$$P_1 = MaxPooling2D((2,2))(C_1)$$

The resulting tensor $P_1 \in \mathbb{R}^{127\times127\times32}$

Step 7: Apply a second convolutional layer with 64 filters of size $3 \times 3$, followed by a ReLU activation function.

$$C_1 = ReLU\left(Conv2D(64, (3,3))(P_1)\right)$$

This operation results in an output tensor $C_2 \in \mathbb{R}^{125\times125\times64}$

Step 8: Apply another max pooling with a $2 \times 2$ window.

$$P_2 = MaxPooling2D((2,2))(C_2)$$

The resulting tensor $P_2 \in \mathbb{R}^{62\times62\times64}$

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online))Volume 4, Issue 6, June 2015)

Step 9: Flatten the features maps to convert the 3D tensor into a 1D vector

$$f = Flatten(P_2)$$
$$P_2 = MaxPooling2D((2,2))(C_2)$$

The resulting tensor $P_2 \in \mathbb{R}^{246016}$

Step 10: Apply a dense (fully connected) layer with 128 units a ReLU activation function.

$$d = ReLU\big(Dense(128)(f)\big)$$

The resulting vector $d \in \mathbb{R}^{128}$

Step 11: Concatenate the outputs of DenseNet 121 (g) and the custom CNN (d).

$$z = Concatenate\ ([g, d])$$

The resulting concatenated vector $z \in \mathbb{R}^{1152}$

Step 12: Apply a final dense layer for classification with the number of units equal to the number of classes in the PlantVillage dataset (num_classes=38), and a softmax activation function to produce the output probabilities.

$$y = Softmax(Dense(38)(z))$$

The results in a probability vector y$\in \mathbb{R}^{38}$

Step 13: Compile the model with an appropriate optimizer (e.g., 'adam'), loss function (e.g., 'categorical_crossentropy'), and metrics (e.g., 'accuracy')

Step 14: Train the model using the training data $(X_{train}, Y_{train})$ for a specified number of epochs and batch size with a validation split.

Step 15: Evaluate the trained model on a separate test set to measure its performance.

## IV. RESULT ANALYSIS

Performance metrics are critical for assessing the effectiveness of neural network learning models in image classification tasks, providing quantitative measures of their accuracy, reliability, and generalization capabilities. Several key performance metrics, along with their corresponding formulas, are commonly utilized in evaluating these models:

**Accuracy:** Accuracy measures the proportion of correctly classified images out of the total number of images in the dataset. It is calculated as the ratio of the sum of true positive (TP) and true negative predictions (TN) to the total number of predictions. The formula for accuracy is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100$$

**Precision:** Precision assesses the model's capacity to accurately recognize positive cases among all instances labeled as positive. It is determined by the ratio of true positive predictions to the total of true positive and false positive predictions. The formula for precision is:

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

**Recall (Sensitivity):** Recall quantifies the proportion of true positive cases correctly identified by the model out of all actual positive cases. It is calculated as the ratio of true positive predictions to the total of true positive and false negative predictions. The formula for recall is:

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

**Specificity:** Specificity evaluates the model's capability to accurately identify negative cases out of all instances classified as negative. It is determined by the ratio of true negative predictions to the total of true negative and false positive predictions. The formula for specificity is:

$$Specificity = \frac{TN}{TN + FP}$$

**F1-score:** The F1-score represents the harmonic mean of precision and recall, offering a balanced assessment of the model's performance. It is calculated as follows:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

**Area under the Receiver Operating Characteristic Curve (AUC-ROC):** AUC-ROC provides a summary of the model's overall performance across different classification thresholds by plotting the true positive rate (sensitivity) against the false positive rate (1-specificity). The AUC-ROC score ranges from 0 to 1, with higher values indicating better class discrimination.

**Confusion Matrix Analysis:** The confusion matrix offers a comprehensive overview of the model's predictions, indicating the counts of true positives (TP), true negatives (TN), false positives (FP), and false

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online))Volume 4, Issue 6, June 2015)

negatives (FN). Using the confusion matrix, various performance metrics, including accuracy, precision, recall, and specificity, can be calculated.

Following are the Results of the Proposed DenseNet model:

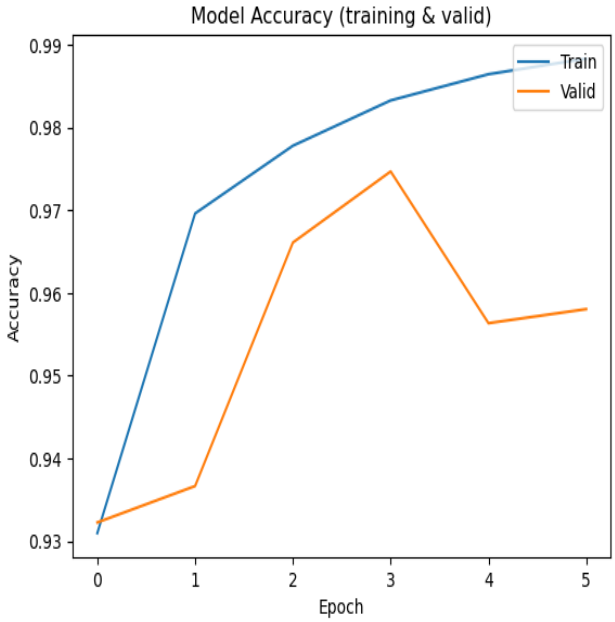



Figure 4: Training and Validation Accuracy

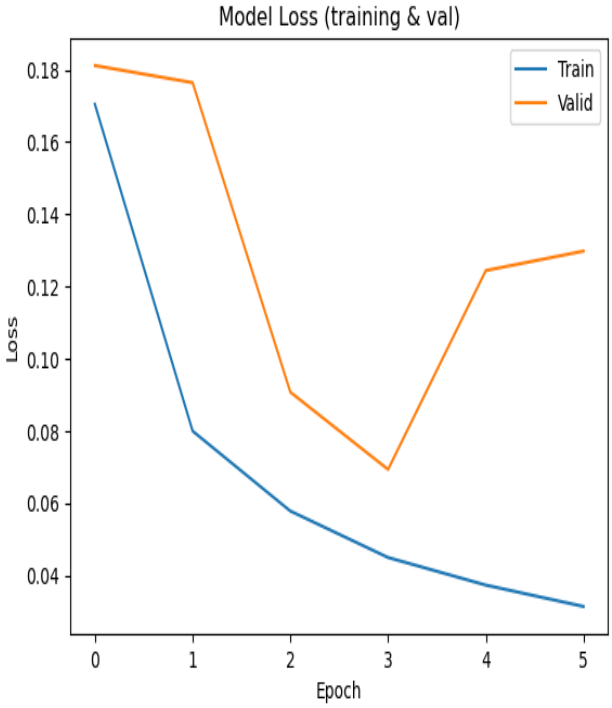
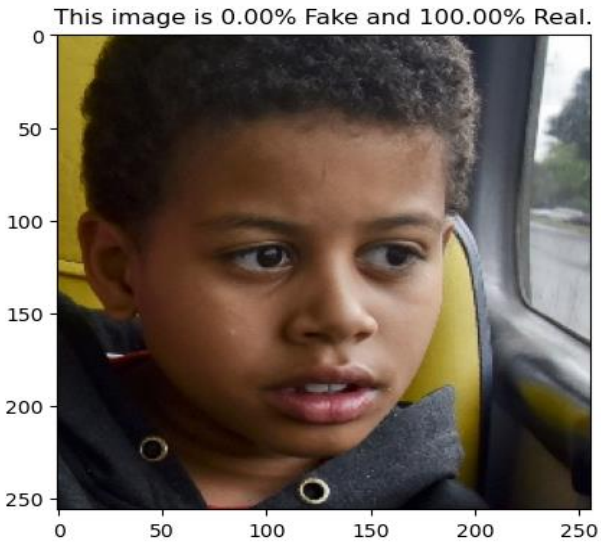



Figure 5: Training and Validation Loss

Qualitative Evaluation:

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online))Volume 4, Issue 6, June 2015)





Quantitative Results:

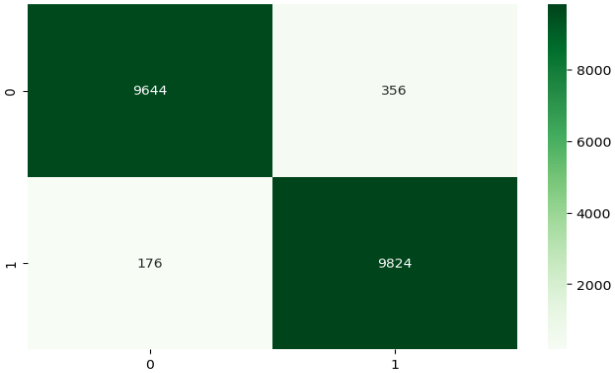



Figure 6: Confusion Matrix for Custom DenseNet121 model on test set

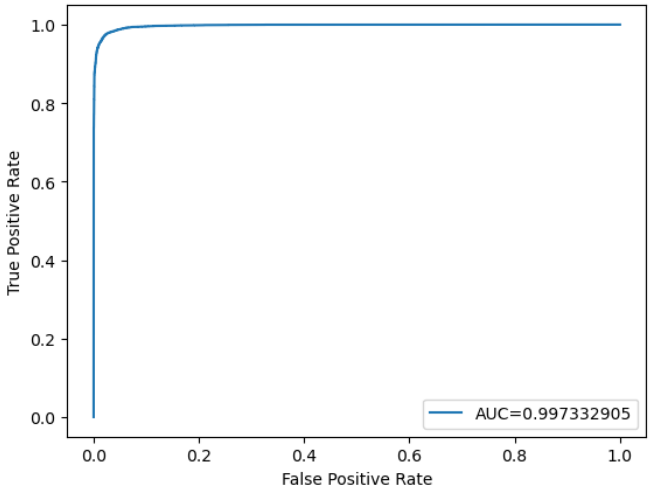



Figure 7: ROC Curve for True and False Predictions

Test performance:
313/313 [==============================] - 21 5s 689ms/step - loss: 0.0727 - accuracy: 0.9734
Test Loss: 0.0726810023188591
Test Accuracy: 0.9733999967575073

| Classification Report : | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.96 | 0.97 | 10000 |
| 1 | 0.97 | 0.98 | 0.97 | 10000 |
| accuracy | | | 0.97 | 20000 |
| macro avg | 0.97 | 0.97 | 0.97 | 20000 |
| weighted avg | 0.97 | 0.97 | 0.97 | 20000 |

Figure 8: Classification Report on test Set

Table 1 displays the results of Ali Raza et al. [1] and proposed method in terms of accuracy and precision. Ali Raza et al. [1] give an accuracy of 89%, & a precision of 89% for CNN model, an accuracy of 90%, & a precision of 88% for VGG19 model, an accuracy of 88%, & a precision of 86% for MobileNet model, an accuracy of 94%, & a precision of 95% for DFP model. The proposed DenseNet121 model provides an accuracy of 97.33% and a precision of 98%. Clearly, the proposed DenseNet121 model is a 3.421% improvement accuracy and 3.061% improvement precision compared to Ali Raza et al. [1]. Fig. 9 and figure 10 shows the graphical representation of the comparison result.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online))Volume 4, Issue 6, June 2015)

**Table 1: Comparison Result**

| Model | Technique | Accuracy Score | Precision Score |
|---|---|---|---|
| Dee Learning | CNN | 89% | 89% |
| Transfer Learning | VGG16 | 90% | 88% |
| Transfer Learning | MobileNet | 88% | 86% |
| Dee Learning | DFP | 94% | 95% |
| Dee Learning | Proposed DenseNet121 | 97.33% | 98% |

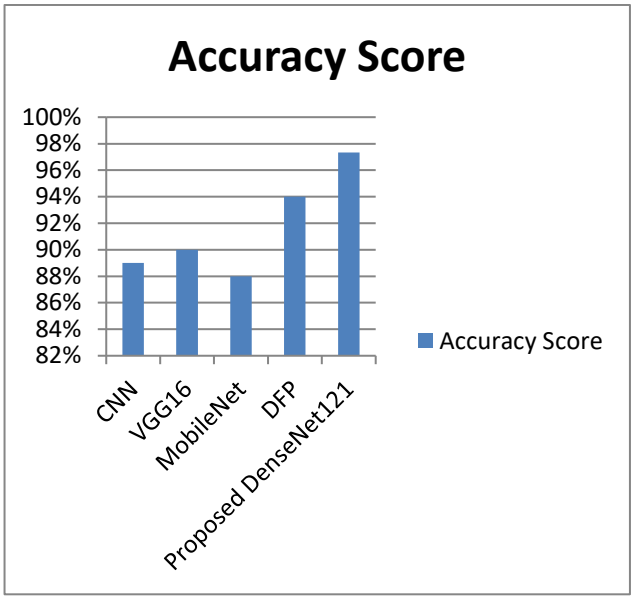



Figure 9: Graphical Accuracy

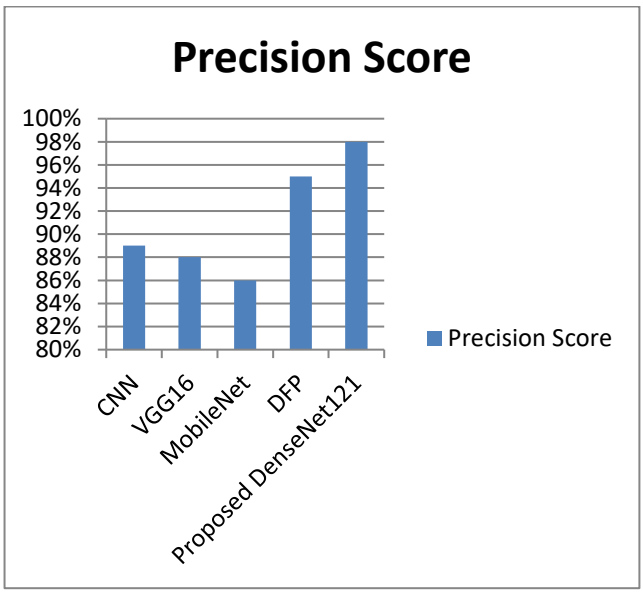



Figure 10: Graphical Precision

## V. CONCLUSION

The hybrid model developed for the Plant Village dataset demonstrates a robust approach to detecting diseases in plant leaves, leveraging a combination of DenseNet121 and custom convolutional neural network (CNN) layers. DenseNet121, a powerful feature extractor pretrained on ImageNet, provides a strong foundation for capturing intricate patterns and features from the leaf images. This pretrained base is augmented by additional CNN layers designed to further refine and process these features, enhancing the model's ability to distinguish between deepfake image accurately.

To enhance training efficiency, all but the last 50 layers of DenseNet121 are frozen, ensuring that the pre-trained knowledge is retained while allowing the final layers to be fine-tuned. The output from DenseNet121 is passed through a Global Average Pooling layer, reducing the feature maps to a single vector.

## REFERENCES

[1] Ali Raza, Kashif Munir and Mubarak Almutairi, "A Novel Deep Learning Approach for Deepfake Image Detection", Apply Science, pp. 01-15, 2022.

[2] Zobaed, S.; Rabby, F.; Hossain, I.; Hossain, E.; Hasan, S.; Karim, A.; Hasib, K.M. Deepfakes: Detecting forged and synthetic media content using machine learning. In Artificial Intelligence in Cyber Security: Impact and Implications; Springer: Berlin/Heidelberg, Germany, 2021; pp. 177–201.

[3] Thambawita, V.; Isaksen, J.L.; Hicks, S.A.; Ghouse, J.; Ahlberg, G.; Linneberg, A.; Grarup, N.; Ellervik, C.; Olesen, M.S.; Hansen, T.; et al. DeepFake electrocardiograms using generative adversarial networks are the beginning of the end for privacy issues in medicine. Sci. Rep. 2021, 11, 21869.

[4] Ahmed, M.F.B.; Miah, M.S.U.; Bhowmik, A.; Sulaiman, J.B. Awareness to Deepfake: A resistance mechanism to Deepfake. In Proceedings of the 2021 International Congress of Advanced Technology and Engineering (ICOTEN), Taiz, Yemen, 4–5 July 2021; pp. 1–5.

[5] Brian Dolhansky, Russ Howes, Ben Pflaum, Nicole Baram, and Cristian Canton Ferrer. The Deepfake Detection Challenge (DFDC) Preview Dataset. arXiv eprints, page arXiv:1910.08854, 2019.

[6] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The DeepFake Detection Challenge Dataset. arXiv e-prints, page arXiv:2006.07397, 2020.

[7] Md Shohel Rana, Mohammad Nur Nobi, Beddhu Murali, and Andrew H Sung. Deepfake detection: A systematic literature review. IEEE access, 10:25494–25513, 2022.

[8] Jin, B.; Cruz, L.; Gonçalves, N. Deep facial diagnosis: Deep transfer learning from face recognition to facial diagnosis. IEEE Access 2020, 8, 123649–123661.

[9] Lewis, J.K.; Toubal, I.E.; Chen, H.; Sandesera, V.; Lomnitz, M.; Hampel-Arias, Z.; Prasad, C.; Palaniappan, K. Deepfake Video Detection Based on Spatial, Spectral, and Temporal Inconsistencies Using Multimodal Deep Learning. In Proceedings of the 2020 IEEE Applied Imagery Pattern RecognitionWorkshop (AIPR),Washington DC, DC, USA, 13–15 October 2020; pp. 1–9.

[10] Lee, H.; Park, S.H.; Yoo, J.H.; Jung, S.H.; Huh, J.H. Face recognition at a distance for a stand-alone access control system. Sensors 2020, 20, 785.

[11] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen. Attgan: Facial attribute editing by only changing what you want. IEEE Transactions on Image Processing, 28(11): 5464–5478, 2019.

[12] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. pages 1526–1535, 2018.

[13] Guha Balakrishnan, Amy Zhao, Adrian V. Dalca, Fr´edo Durand, and John Guttag. Synthesizing images of humans in unseen poses. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[14] Soumya Tripathy, Juho Kannala, and Esa Rahtu. Icface: Interpretable and controllable face reenactment using gans. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2020.

[15] Wayne Wu, Yunxuan Zhang, Cheng Li, Chen Qian, and Chen Change Loy. Reenactgan: Learning to reenact faces via boundary transfer. In Proceedings of the European Conference on Computer 2018.