DOI: http://dx.doi.org/10.26483/ijarcs.v15i2.7060

Volume 15, No. 2, March-April 2024

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

LEVERAGING DEEP LEARNING FOR ACCURATE RICE LEAF DISEASE RECOGNITION

[1]Dr Radha Karampudi,[2]Priyanshu C,[3]Venkat Swaroop Veerla,[4]P. Abhinav Chowdary,[5]A. Sai Nikhil Balaji,[6] Srikanth Raavi

[1]Assistant Professor, CSE, GITAM University, Hyderabad.

[2]CSE, GITAM University, Hyderabad

[3,4,5,6]CSEDS, GITAM University, Hyderabad,India

Abstract: Imagine a Rice Field, seemingly healthy, yet harboring invisible foes. This paper delves into the world of crop disease detection for the Tan spot, Leaf blight, Sheath decay, Bacterial leaf rot, and False smut Dataset consists of 3546 images. The image segmentation is done by BIRCH clustering followed by GMM clustering; various texture features have been extracted through this. The classification is done using existing models such as VGG16, VGG19, RESNET50, and INCEPTION V3. After comparing various models and validation, the observation was made that VGG19 performs well compared to other models. The overall accuracy obtained for the VGG19 is 95.88.%. The achieved accuracy surpasses that of conventional backpropagation neural network models, indicating significant advancements in crop disease diagnosis. This study introduces a novel approach that opens avenues for future research in the field of deep learning for crop disease diagnosis.

Keywords: GMM Clustering, VGG16, VGG19, crop disease diagnosis.

I. INTRODUCTION

Cereals play a crucial role in sustaining the global human population, with approximately 50% of consumed calories derived from wheat, rice, and maize [1]. Despite rice ranking second in terms of planted area, it holds paramount importance as a staple food source in Asian countries, particularly in the southeast, where it serves as a vital economic crop for millions of farmers and workers across extensive hectares [2]. Rice cultivation dates back 10,000 years, originating in the river valleys of South and Southeast Asia and China, making it a fundamental food source historically. While Asia remains the primary region for rice cultivation, it is also grown in other continents such as Latin America, Europe, parts of Africa, and even the USA [1]. The agricultural community faces significant challenges due to rice diseases like Leaf blasts, Sheath blight, BLB, Brown spots, and False smut, causing substantial crop damage before harvest and leading to significant losses for farmers. Generally, identifying diseases in rice plants relies on either visually assessing symptoms or conducting laboratory experiments to culture pathogens. Visual assessment is subjective and susceptible to errors, while culturing pathogens in the lab is time-consuming and may not yield timely results [3]. Even with conventional methods, farmers face challenges in identifying diseases, especially in rural areas where access to agriculture experts and lab facilities is limited. Consequently, the research community is actively seeking an automated identification and classification method that can promptly categorize diseases and offer farmers necessary guidance or recommend pesticides for infected crops. We captured images of rice diseases in rural areas, specifically obtaining 894 pictures of Brown Spot, 779 images of Bacterial Leaf Blight, 864 images of Sheath Blight, 858 images of False Smut, and 509 images of Leaf Blast. In total, we compiled approximately 3,546 images for training, 240 images for testing, and 120 images for validation purposes.

A. Image Preprocessing

We conducted experiments involving different algorithms and explored various methods, ultimately discovering that clustering segmentations outperformed other available segmentation methods. We have provided detailed insights into several segmentation methods utilized during the experiments and presented their respective outcomes.

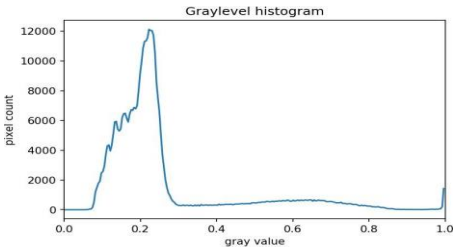

Fig.1(a). Thresholding

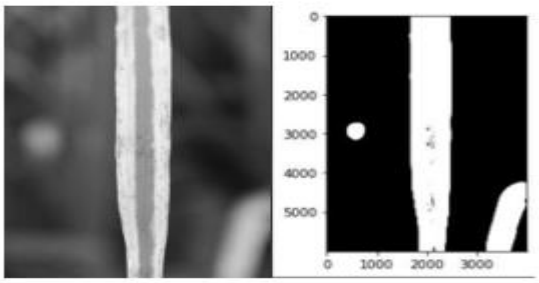

Fig.1(b). Thresholding

B. Thresholding

Thresholding, a form of image segmentation, involves modifying pixel values in an image to simplify its analysis. This process entails converting a color or grayscale image into a binary format, specifically black and white. The primary purpose of thresholding is commonly to identify and isolate areas of interest within an image, disregarding the regions that are not relevant to the analysis ("Thresholding – Image Processing with Python").

1	2	1
2	4	2
1	2	1

C. Canny Edge Detector

1. The Canny edge detector is a multi-step algorithm designed to identify edges in an input image. The process includes a series of steps that need to be followed to effectively detect the edges of the image.
2. Calculating the derivative of a Gaussian filter is done to compute the gradient of image pixels, enabling the determination of magnitude along both the x and y dimensions.

Table 1. Canny Edge Detector

1/16

3. Examining a set of neighboring points along a direction perpendicular to a specified edge, eliminate non-maximum contributors among the pixels associated with the edge.

4. Finally, apply the Hysteresis Thresholding method to retain pixels with a gradient magnitude above a certain threshold value while discarding those below the specified low threshold.

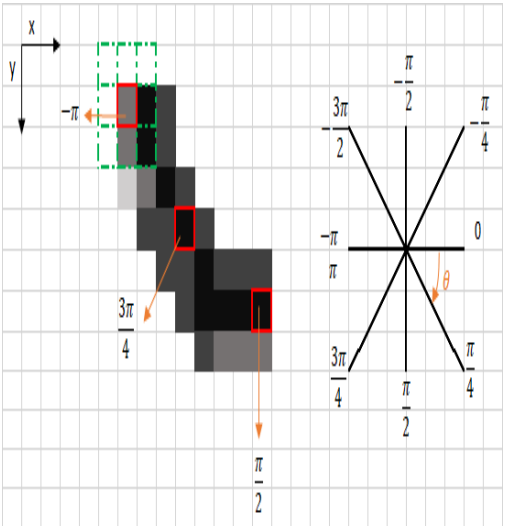

Fig.2(a). Canny edge Detector

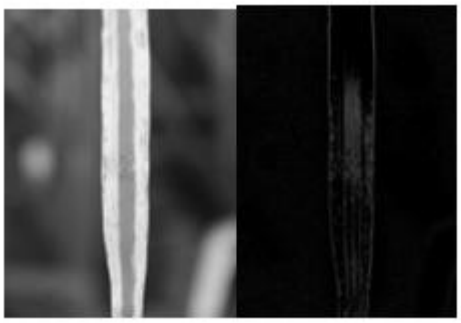

Fig.2(b). Canny edge Detector

D. K-Means

K-Means is a clustering algorithm categorized under unsupervised algorithms, implying the absence of labeled data. Its purpose is to discern distinct classes or clusters within the provided data by assessing the similarity of data points. Clustering algorithms inherently leverage the principle that data points within a cluster exhibit greater proximity in the feature space compared to those falling outside the cluster .K-Means clustering stands out as one of the widely employed algorithms for this purpose, with 'k' denoting the number of clusters.

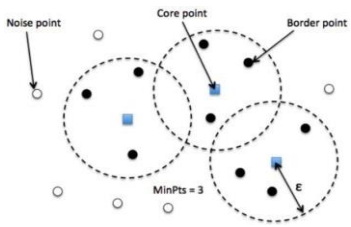

Fig.3(a). K-Means

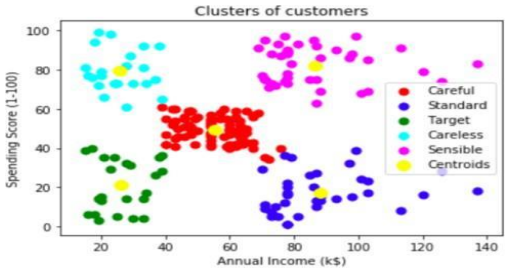

Fig.3(b).K-Means

E. *Elbow Method*

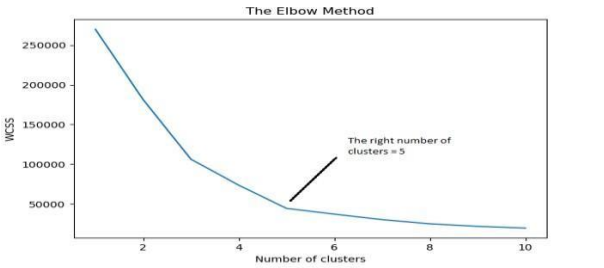

Fig.4(a). Elbow Graph

The total WCSS serves as a metric for the compactness of the clustering, and the objective is to minimize this value, making the clusters as tightly grouped as possible.

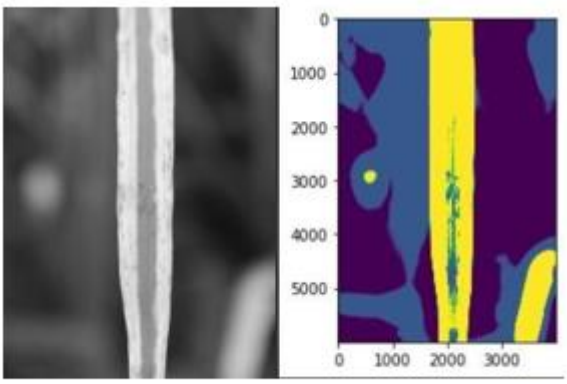

Fig.4(b). Elbow Graph

F. *Mean Shift Segmentation*

Mean Shift segmentation is a technique for local homogenization, particularly effective in mitigating shading or tonality variations within localized objects. Typically, Mean Shift requires three inputs:

1. A distance function, often the Euclidean distance, but other well-defined distance functions such as Manhattan distance may also be utilized.

2. A specified radius, encompassing all pixels within this range for calculation purposes based on the chosen distance metric.

3. A value difference criterion, allowing the inclusion of only those pixels within the radius whose values fall within this specified difference for mean calculation

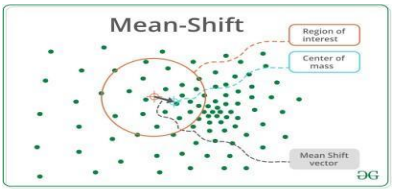

Fig.5(a). Mean Shift Segmentation

Fig.5(b). Mean Shift Segmentation

G. *DBSCAN*

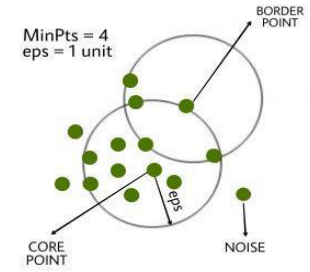

Fig.6. DBSCAN

The DBSCAN algorithm relies on the intuitive concepts of "clusters" and "noise." The fundamental principle is that the point in a cluster must have a minimum number of neighboring points within a specified radius.

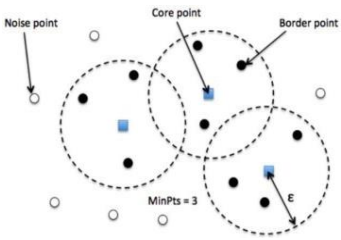

Fig.7(a). Mid Points

DBSCAN algorithm requires two parameters:

In DBSCAN, 'eps' defines a data point's neighborhood range, where distances equal to or less than 'eps' indicate neighboring points. If 'eps' is too small, a substantial portion of the data may be classified as outliers, while an excessively large 'eps' can lead to the merging of clusters, consolidating most data points into the same clusters. Determining the 'eps'

value can involve analyzing the k-distance graph.

MinPts signifies the minimum number of neighbors within the 'eps' radius, with larger datasets requiring a higher MinPts. A general guideline for setting MinPts involves deriving it from the number of dimensions 'D' in the dataset, where MinPts should be at least 'D+1.' It is essential to select a minimum MinPts value of at least 3 (Dey).

Algorithmic steps for DBSCAN clustering

1. Randomly choose a point from the dataset and mark it as visited.
2. Check if there are at least 'minpoint' points within a distance of 'ε' from the selected point.If so, consider all these points as part of the same cluster.
3. Continue expanding the clusters by recursively checking the neighbourhood of each adjacent point. following a recursive approach (Chauhan).

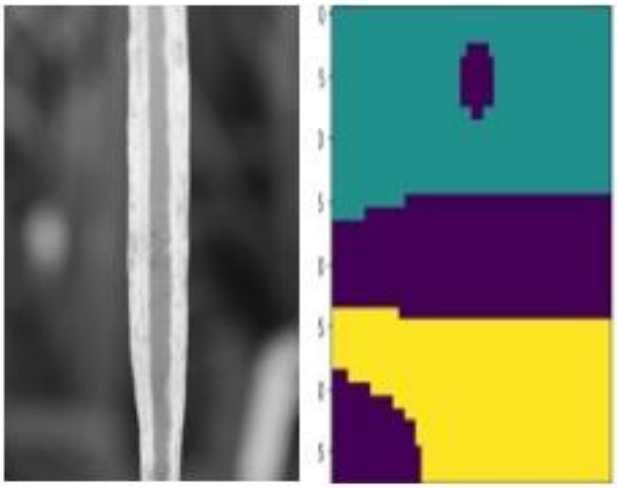

Fig.7(b).DBSCAN clustering recursive approach(Chauhan)

H. BIRCH

The Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is a clustering algorithm for large datasets. It starts by generating a compact summary of the dataset, aiming to retain key information. This summary is clustered instead of the original dataset, making it more manageable. BIRCH is often used in conjunction with other clustering algorithms, offering a summarized dataset for further analysis.

The CF tree, integral to BIRCH, is a tree structure with balanced height. It collects and manages clustering features while retaining essential information about the given data for subsequent hierarchical clustering. This design obviates the necessity to process the entire input dataset. The tree represents the clustered data points as CF, consisting of three numerical values (N, LS, SS).

In the context of the CF tree:

- N represents the number of items in subclusters.

- LS denotes the vector sum of the data points.

- SS stands for the sum of the squared data points.

In the provided image, the values for a specific example are illustrated. For a set of five samples with coordinates (3,4), (2, 6), (4, 5), (4, 7), and (3, 8), the corresponding CF values are N = 5, LS = (16,30), and SS = (54, 190). The image visually depicts the structure of the CF tree based on these values (Gupta).

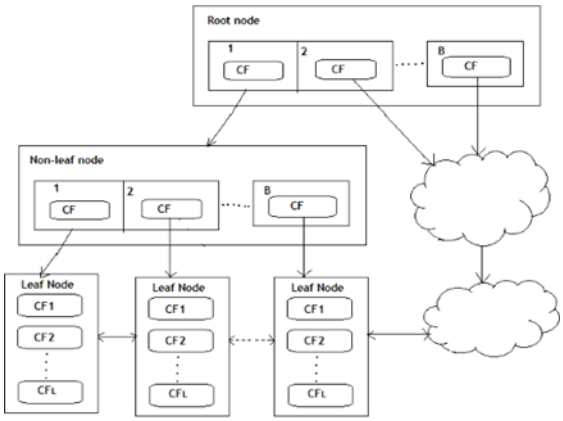

Fig.8 A CF Tree

The diagram shows a hierarchical structure starting with the root node, which contains non-leaf nodes. Each non-leaf node has B entries, and the leaf nodes contain L cluster features (CF). A leaf node is considered a sub-cluster if each CF within it has L entries, satisfying the threshold condition where T is the maximum diameter of the radius. These leaf nodes act as summaries rather than individual data points. The BIRCH algorithm progresses through four main phases.

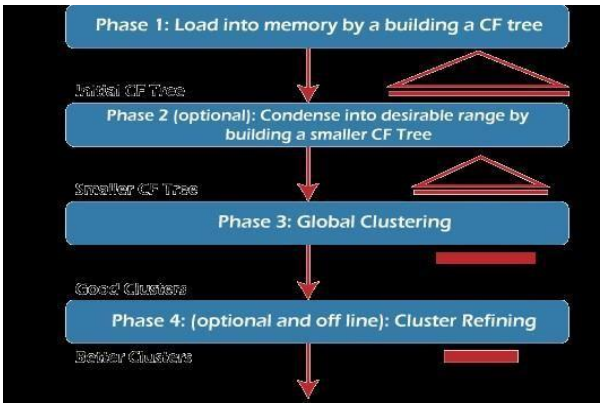

Fig.9(a). BIRCH

Fig.9(b). BIRCH

I. GMM

The Gaussian mixture model (GMM) involves mixing multiple Gaussian distributions. Instead of identifying clusters based on nearest centroids, we fit k Gaussians to the data. We estimate parameters like mean, variance, and weight for each cluster. By learning these parameters for each data point, we can calculate the probabilities of it belonging to each cluster.

Each distribution in the Gaussian mixture model is multiplied by a weight (π), where the sum of all weights equals 1 (π1 + π2 + π3 = 1). This weighting accounts for the unequal sample sizes in each category. For example, if we have 1000 samples from the red cluster and 100,000 samples from the green cluster, the weights would adjust to reflect this disparity.

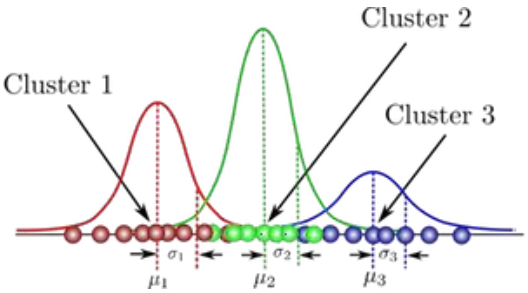

Fig.10.GMM

- Initialize the mean (μk), covariance matrix (Σk), and mixing coefficients (πk) with random or predefined values.

- Calculate the responsibilities (Ck) for each component k.

- Update the parameters (μk, Σk, πk) based on the current responsibilities.

- Compute the log-likelihood function.

- Check for convergence based on a predefined criterion.

- If the log-likelihood value or parameters converge, stop; otherwise, repeat from Step2.

This algorithm can only ensure convergence to a locally optimal solution and does not guarantee convergence to the global optimum. Therefore, if the algorithm is initialized from different starting points, it may converge to different local optima.

II.MODELS

The data we received from the segmentation techniques we are passing through various models and comparing the result here we have used various models which are discussed in detail below.

A. VGG19

The VGG-19 is a convolutional neural network trained by the Visual Geometry Group (VGG) at the University of Oxford. The number"19" signifies the network's total number of layers, including 16 convolutional layers and three fully connected layers.

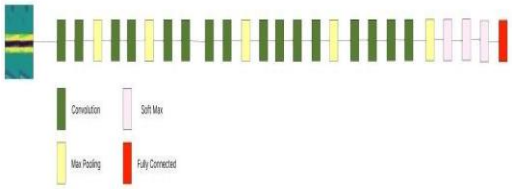

Fig.11 .VGG19

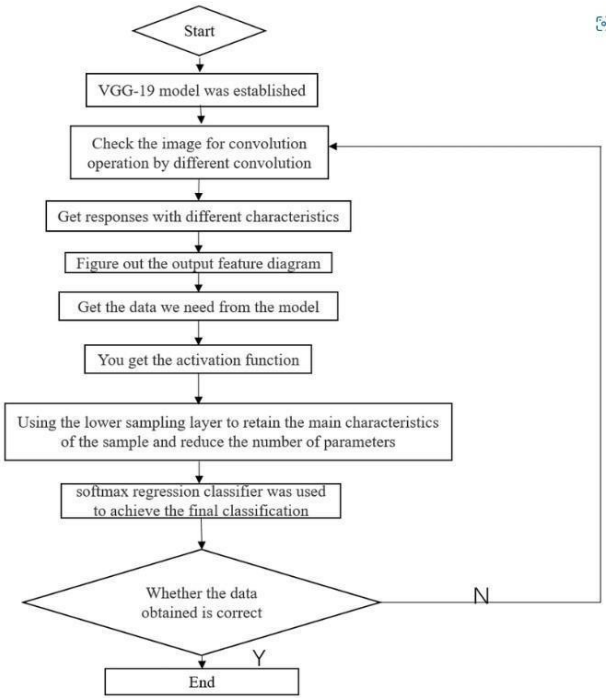

Fig.12 VGG19 Algorithm

III. ARCHITECTURE

- The network took fixed-size (224 * 224) RGB images as input, forming a matrix of shapes (224, 224, 3).
- The only preprocessing step involved subtracting the mean RGB value from each pixel, calculated across the entire training set.
- It utilized 3 * 3 kernels with a stride of 1 pixel to cover the entire image.
- Spatial padding was applied to maintain the spatial resolution of the image.
- Max-pooling was performed over 2 * 2-pixel windows with a stride of 2.
- A Rectified Linear Unit (ReLU) was utilized to introduce non-linearity, enhancing classification accuracy and computational efficiency over older models employing tanh or sigmoid functions.
- The network consisted of three fully connected layers, with the first two having 4096 units each. The final layer had 1000 channels for 1000-way ILSVRC classification, followed by a softmax function.

First, we take the image and resize it in height and width into Below is a summary of the main components:

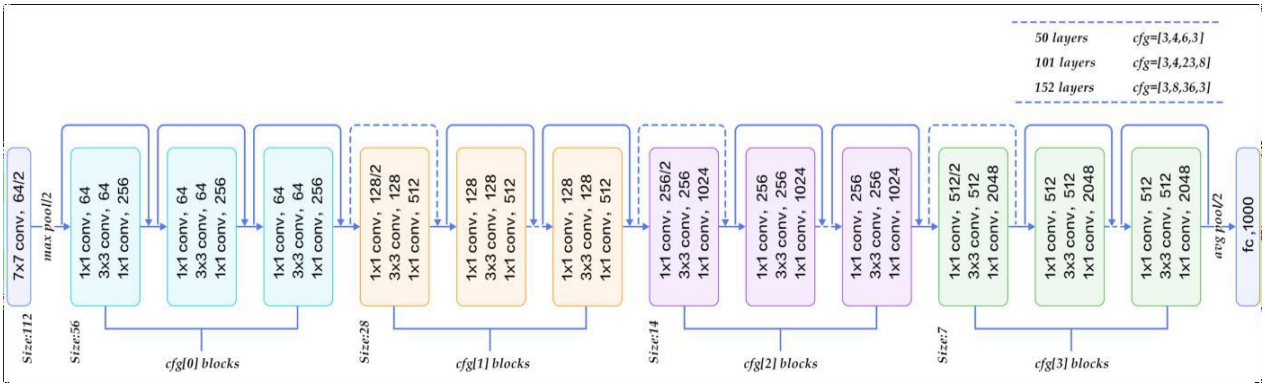

Fig.13 ResNet50

The first process consists of a 7x7 convolutional layer with 64 filters and a 2nd step, followed by a max pooling layer with a 2nd step.
- layers follow the model: 1x1 convolution with 64 filters, 3x3 convolution with 64 filters and 1x1 convolution with 256 filters, repeated 3 times.
- Other models shown: 1x1 convolution with 128 filters, 3x3 convolution with 128 filters and 1x1 convolution with 512 filters, repeated 4 times.
- Then comes 1x1 convolution with 256 filters, 3x3 convolution with 256 filters and 1x1 convolution with 1024 filters, the same is repeated 6 times.
- Then there are models of 1x1 convolution with 512 filters, 3x3 convolution with 512 filters and 1x1 convolution with 2048 filters repeated 3 times.
- Finally, there is the intermediate pooling layer and then the softmax activation function, which is a process connected to 1000 nodes.

The total number of layers of this ResNet-50 architecture is 50, including convolutional layer, layer by layer and all 244X244 pixels format and divided the dataset into training and testing in which we have 3274 images and 240 images from that we execute the program for 200 epoch and add an early.

A. *RESNET50*

A Convolutional Neural Network with 50 layers is known as ResNet-50. ResNet, short for Residual Networks, is a foundational neural network in computer vision applications. Its key innovation was enabling the training of extremely deep networks, surpassing 150 layers.
The "vanishing gradient problem" is a major challenge for neural networks. This occurs when the slope becomes too small during the recovery period, causing the weight to shift slightly. ResNet solves this problem by introducing "cross-connecting" or "fast-connecting," which allows gradients to flow through the network more easily and allows networks to be trained very deeply without experiencing vanishing gradients The network architecture is described as ResNet-50 architecture, unlike the ResNet model.

layers.

We don't actually count the activation functions and the max/ average pooling layers.so totaling this it gives us a 1 + 9 + 12 + 18 + 9 + 1 = 50 layers Deep Convolutional network.

First, we take the image and resize it in height and width into 244X244 pixels format and divided the dataset into training and testing in which we have 3274 images and 240 images from that we execute the program for 200 epochs and add an early stopping model put validation loss for monitoring and put up the patience of 3.

B. VGG16

The entrance into the community comprises an image with dimensions of 224 by 224 pixels and three color channels. Initially, there are convolutional layers with 64 filters, each with a size of three by three pixels, and padding is applied to maintain the spatial dimensions. Subsequently, a max-pooling layer with a stride of (2, 2) is applied. Following this,

there are convolutional layers with 128 filters, also of size three by three, and another max-pooling layer with the same stride. This pattern continues with two sets of convolutional layers, each with 256 filters of size three by three, and another two sets with 512 filters. Additionally, 1 by 1 convolutions are utilized to adjust the number of input channels, and padding is consistently applied after each convolutional layer. This approach contrasts with the larger filter sizes seen in AlexNet and ZF-Net, using three by three filters instead of eleven by eleven or seven by seven.

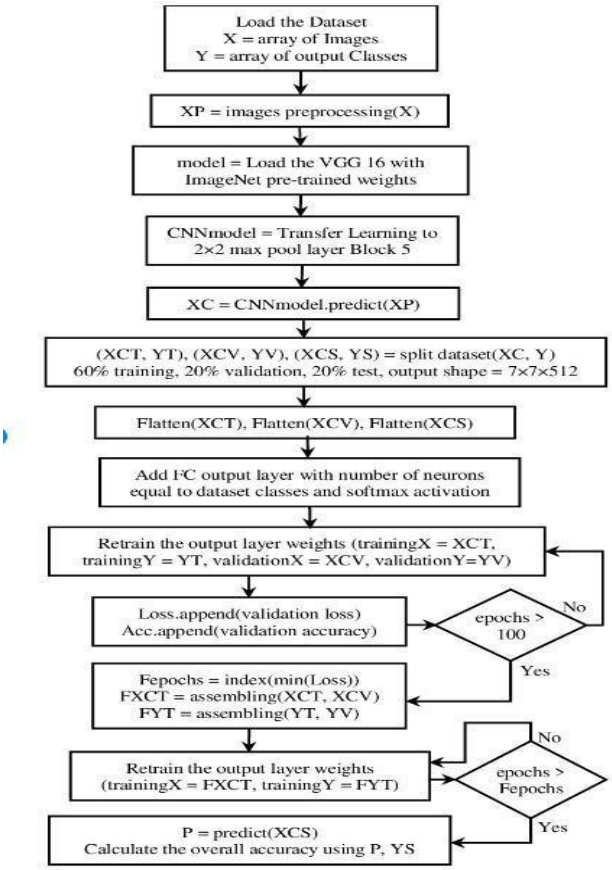

Fig.14 . VGG16

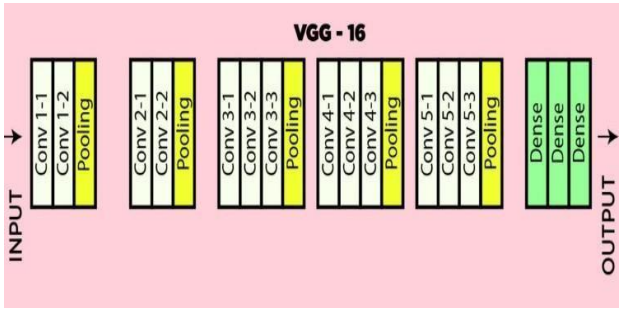

Fig.15. VGG16

C. INCEPTION V3

Inception Networks, including GoogLeNet or Inception v1, are known for their computational efficiency compared to VGGNet. This efficiency is measured by the number of parameters and overall resource costs required. When modifying an Inception Network, it's crucial to maintain these computational advantages.

However, this can be challenging due to uncertainties about the network's performance after modifications. To address this, several optimization strategies have been proposed for Inception v3. These include factorized convolutions, regularization techniques, dimension reduction, and parallelized calculations. These strategies aim to relax constraints and enhance the adaptability of the model, making it more versatile for different use cases.

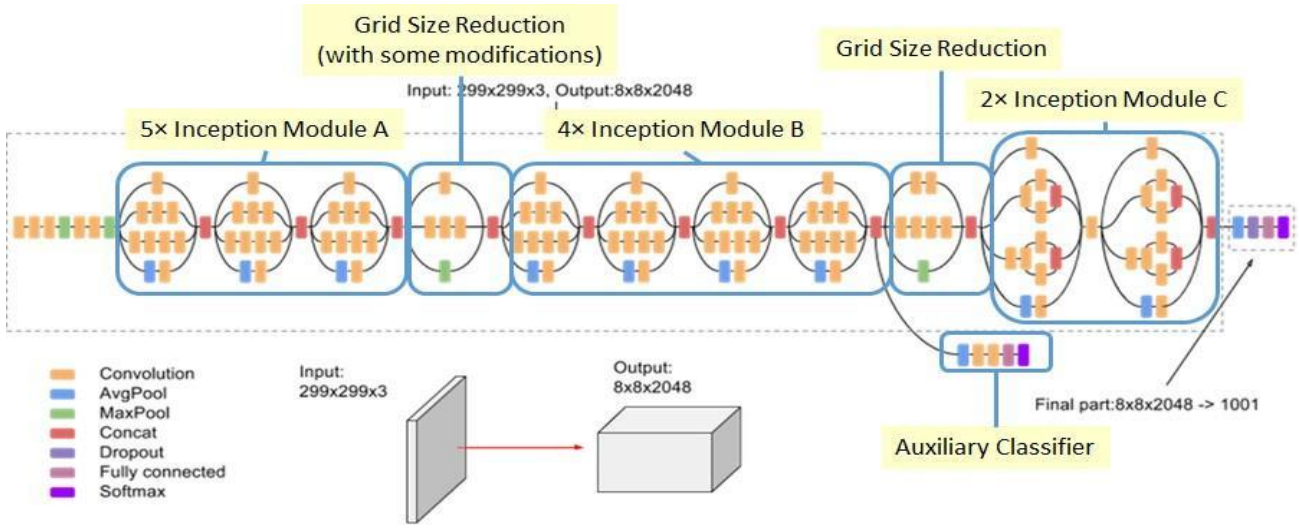

Fig.16 INCEPTION V3

First, we take the image and resize it in height and width into 244X244 pixels format and divided the dataset into training and testing in which we have 3274 images and 240 images from that we execute the program for 200 epochs and add an early stopping model put validation loss for monitoring and put up the patience of 3.

D. Voting Classifier

The voting classifier predicts the output class based on the majority votes of various classifiers. It includes results from distributions such as RESTNET50, VGG19, VGG16, and INCEPTIONV3. Rather than evaluating each candidate individually, voters combine their predictions to determine the final product. This approach is designed to create a more robust model by leveraging all distributions.

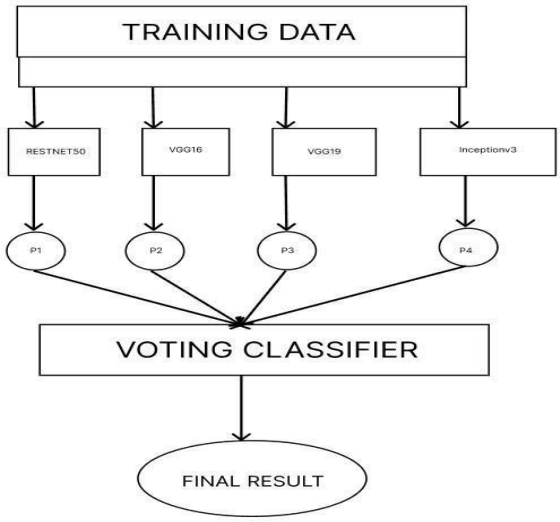

Fig.17. Voting Classifier Algorithm

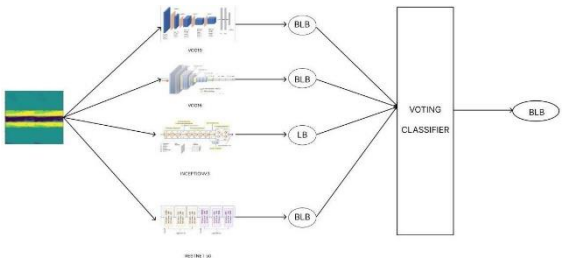

Fig.18. Voting Classifer

IV. VALIDATION AND RESULTS

We need to validate the results of the models we have taken before applying the voting classifier. To that end, we have stored approximately 24 images from each of the five disease categories, from which the algorithm has not yet been trained and passed. For each of the models we have mentioned we have obtained the following results; however, we must first take all the diseases and prepare them for validation. Here are the results for each model after all the photographs were imported for validation and images were running for each model.

Here are the results for each model after all the photographs were imported for validation and images were running for each model.

A. VGG16

We employed three segmentation strategies that produced satisfactory patterns and sent the data to the VGG16. These techniques include Birch segmentation, Mean shift segmentation, and

B. BIRCH and GMM

Table 2. BIRCH and GMM

ACCURACY	91.66
ERROR	8.333

With VGG16 we are getting around 91.66 Validation accuracy.

Table 3. VGG16

	Precision	Recall	F1-score	Support
BLB	0.86	0.86	0.86	7
Brown Spot	1.00	1.00	1.00	4
False Smut	1.00	1.00	1.00	4
Sheath Blight	1.00	0.67	0.80	3
Leaf blast	0.86	1.00	0.92	6
Accuracy			0.92	24

Table. 4. BIRCH

ACCURACY	91.66
ERROR	8.333

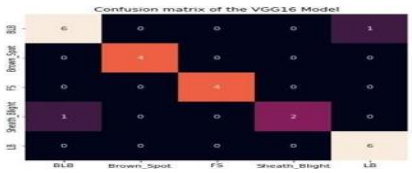

Fig.19(a).VGG16

By using Birch segmentation, we are getting 91.66 accuracy

Table 5. BIRCH Shift Segmentation

	Precision	Recall	F1-score	Support
BLB	0.86	0.86	0.86	7
Brown Spot	1.00	1.00	1.00	4
False Smut	1.00	1.00	1.00	4
Sheath Blight	1.00	0.67	0.80	3

Leaf blast	0.86	1.00	0.92	6
Accuracy			0.92	24
Macro Average	0.94	0.90	0.92	24
Weighted Average	0.92	0.92	0.91	24

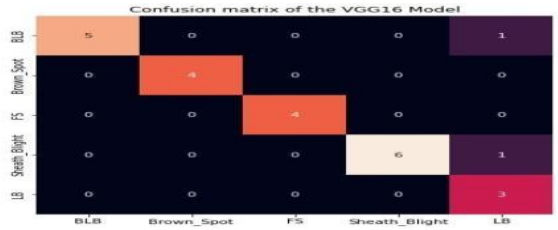

Fig.19(b). BIRCH Segmentation

C. Mean Shift Segmentation

By using Mean Shift segmentation, we are getting 74.19 accuracy

Table 6. Accuracy of Meanshift Segmentation

ACCURACY	74.193
ERROR	25.806

Table 7. Meanshift Segmentation

	Precision	Recall	F1-score	support
BLB	1.00	0.27	0.43	11
Brown Spot	1.00	1.00	1.00	4
False Smut	1.00	1.00	1.00	4
Sheath Blight	0.50	1.00	0.67	6
Leaf blast	0.75	1.00	0.86	6
Accuracy			0.74	31
Macro Average	0.85	0.85	0.79	31
Weighted Average	0.85	0.74	0.711	31

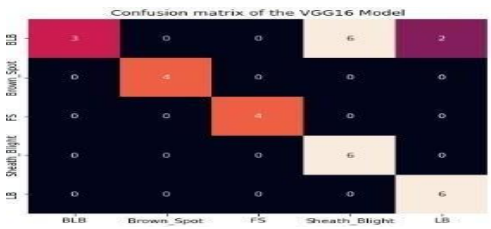

Fig.20.Confusion Matrix for Mean Shift Sementation

D. RESNET 50

Table 8. RESNET 50

ACCURACY	66.66
ERROR	33.33

With RESNET 50 we are getting 66.66 accuracy

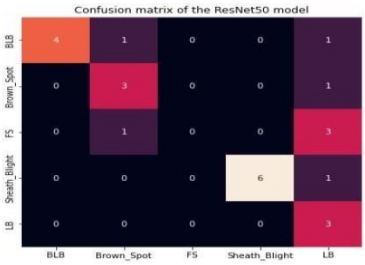

Fig.21. RESNET 50

Table 9. RESNET 50

	Precision	Recall	F1-score	support
BLB	1.00	0.67	0.80	6
Brown Spot	0/60	0.75	0.67	4
False Smut	0.00	0.00	0.00	4
Sheath Blight	1.00	0.86	0.92	7
Leaf blast	0.33	1.00	0.50	3
Accuracy			0.67	24
Macro Average	0.59	0.65	0.58	24
Weighted Average	0.68	0.67	0.64	24

E. BIRCH

we are getting 79.166 accuracy using BIRCH

Table 10. BIRCH

	Precision	Recall	F1-score	support
BLB	0.86	0.86	0.86	7
Brown Spot	1.00	0.75	0.86	4
False Smut	1.00	0.50	0.67	4
Sheath Blight	1.00	1.00	0.67	3
Leaf blast	0.83	0.83	0.83	6
Accuracy			0.79	24
Macro Average	0.84	0.79	0.78	24
Weighted Average	0.85	0.79	0.80	24

F. Mean Shift Segmentation

Table 11.Mean Shift Segmentation

ACCURACY	80.64
ERROR	19.35

we are getting 80.645 accuracy using Mean shift segmentation.

Table 12 . Mean shift segmentation

	Precision	Recall	F1-score	support
BLB	1.00	0.50	0.67	12
Brown Spot	0.67	0.75	0.71	8
False Smut	0.80	1.00	0.89	4
Sheath Blight	0.50	0.80	0.62	5
Leaf blast	0.67	1.00	0.80	2
Accuracy			0.71	31
Macro Average	0.73	0.81	0.74	31
Weighted Average	0.79	0.71	0.71	31

G.VGG19

BIRCH and GMM

Table 13. VGG19

ACCURACY	95.833
ERROR	4.166

With VGG19 we are getting 95.833 Validation accuracy

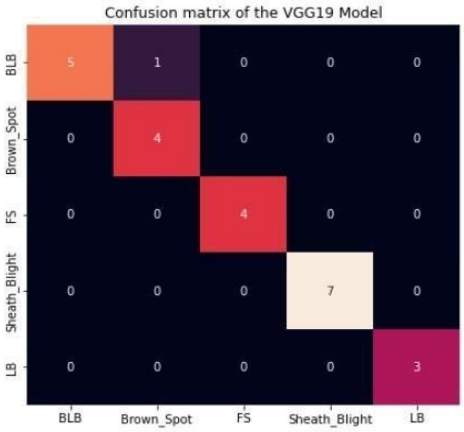

Fig. 22. VGG19 Model

Table 14. Confusion Matrix for VGG19

	Precision	Recall	F1-score	support
BLB	1.00	0.83	0.91	6
Brown Spot	0.80	1.00	0.89	4
False Smut	1.00	1.00	1.00	4
Sheath Blight	1.00	1.00	1.00	7
Leaf blast	1.00	1.00	1.00	3
Accuracy			0.96	24
Macro Average	0.96	0.97	0.96	24
Weighted Average	0.97	0.96	0.96	24

Table 15. BIRCH Segmentation

ACCURACY	91.66
ERROR	8.333

By using Birch segmentation we are getting 91.66 accuracy

H. Mean Shift Segmentation

Table 16. Mean Shift Segmentation

ACCURACY	77.419
ERROR	22.5806

Table 17. Mean Shift Segmentation

	Precision	Recall	F1-score	support
BLB	1.00	0.64	0.78	11
Brown Spot	0.80	1.00	0.89	4
False Smut	0.75	0.75	0.75	4
Sheath Blight	0.62	0.83	0.71	6
Leaf blast	0.71	0.83	0.77	6
Accuracy			0.77	31
Macro Average	0.78	0.81	0.78	31
Weighted Average	0.81	0.77	0.77	31

we are getting 77.419 using mean shift segmentation

Table 18. Mean Shift Segmentation

	Precision	Recall	F1-score	support
BLB	1.00	0.64	0.78	11
Brown Spot	0.80	1.00	0.89	4
False Smut	0.75	0.75	0.75	4
Sheath Blight	0.62	0.83	0.71	6
Leaf blast	0.71	0.83	0.77	6
Accuracy			0.77	31
Macro Average	0.78	0.81	0.78	31
Weighted Average	0.81	0.77	0.77	31

I. *INCEPTION V3*

Table 19. Inception V3

ACCURACY	95.833
ERROR	4.166

With INCEPTION V3 we are getting 91.66 Validation accuracy.

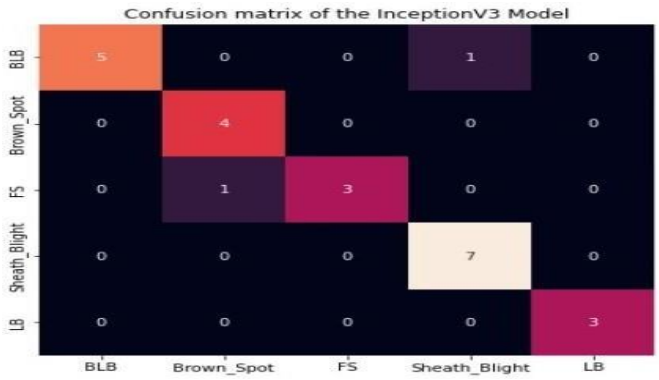

Fig.22. Inception v3 confusion matrix

Table 20. Inception v3 confusion matrix

	Precision	**Recall**	**F1-Score**	**Support**
BLB	0.89	0.73	0.80	11
Brown spot	0.67	1.00	0.80	4
False Smut	0.80	1.00	0.89	4
Sheath Blight	0.80	0.67	0.73	6
Leaf Blast	1.00	1.00	1.00	6
Accuracy			0.84	31
Macro Average	0.83	0.88	0.84	31
Weighted Average	0.85	0.84	0.84	31

J. MEAN SHIFT SEGMENTATION

Table 21. Mean Shift Segmentation

ACCURACY	83.87
ERROR	16.12

here we are getting 83.837 accuracy using Mean shift segmentation.

Table 22. Confusion Matrix for Mean Shift Segmentation

	Precision	**Recall**	**F1-Score**	**Support**
BLB	0.89	0.73	0.80	11
Brown spot	0.67	1.00	0.80	4
False Smut	0.80	1.00	0.89	4
Sheath Blight	0.80	0.67	0.73	6
Leaf Blast	1.00	1.00	1.00	6
Accuracy			0.84	31
Macro Average	0.83	0.88	0.84	31
Weighted Average	0.85	0.84	0.84	31

V. VOTING CLASSIFIER

Now we take all the 4 model and make a voting classifier.

A. GMM AND BIRCH

```
Test set score for Voting Classifier on : 93.809524

Confusion Matrix for Majority Voting Classifier :

[[5 0 0 0 0]
 [0 4 0 0 0]
 [0 0 4 0 0]
 [0 0 0 6 0]
 [1 0 0 1 3]]
```

We can see we are getting an accuracy of 93.80

B. BIRCH

we can see we are getting accuracy of 90.476

C. Mean Shift Segmentation

```
Test set score for Voting Classifier on : 90.476190

Confusion Matrix for Majority Voting Classifier :

[[6 0 0 1 0]
 [0 4 0 0 0]
 [0 0 4 0 0]
 [0 0 0 2 0]
 [1 0 0 0 6]]

Test set score for Voting Classifier on : 87.878788

Confusion Matrix for Majority Voting Classifier :

[[8 0 0 1 0]
 [2 4 0 0 0]
 [0 0 4 1 0]
 [1 0 0 4 0]
 [0 0 0 0 6]]
```

We can see we are getting accuracy of 87.8787.

```
VoteForOne = np.zeros([6,1])for j in range(2):
VoteForOne[TotalPredict[i][j]]+=TotalAccuracy[j] # Addingaccuracy based on
predicted Y
FinalVote = np.argmax(VoteForOne) # Returns index of maximum
row
VoteY.append(FinalVote)



VoteConfMat = confusion_matrix(VoteY, Y_valid)VoteAccuracy =
GetAccuracy(VoteConfMat)
print("Test set score for Voting Classifier on "+": %f" %VoteAccuracy)

return VoteConfMat, VoteY


ConfMat, VoteY =
GetVotingClassifier(yPred1,accuracy1,yPred2,accuracy2,yPred3,accuracy3,
yPred4,accuracy4)
print('\nConfusion Matrix for Majority Voting Classifier :\n')print(ConfMat)
```

Fig.22(a). Predicting the Accuracy of Voting Classifier.

```
def GetAccuracy(CMat):
ConfSum = np.sum(CMat,axis=0)Num = []
for i in range(len(CMat)):
for j in range(len(CMat[i])):if(i==j):
Num.append(CMat[i][j])
Value = 0
for i in range(len(ConfSum)):Value += Num[i]/ConfSum[i]
Value = Value/len(Num) return float(Value)*100
def GetVotingClassifier(pred1,acc1,pred2,acc2,pred3,acc3,pred4,acc4):
TotalPredict = []
for i in range(len(pred1)):
TotalPredict.append([pred1[i],pred2[i],pred3[i],pred4[i]])

TotalAccuracy = [acc1,acc2,acc3,acc4]VoteY = []
for i in range(len(pred1)):
```

Fig.22(b). Predicting the Accuracy of Voting Classifier.

VI. CONCLUSION

We can observe that when we compared the above data of Mean shift , Birch ,Birch and GMM of different models given which are Resnet50, VGG16,Inception V3,VGG19 The VGG19 of Birch and GMM is performing really well compared to the above mentioned models we are getting around 95.88% accuracy when we used VGG19 which is the highest observed validation accuracy compared to other models mentioned. In Summary we can conclude that combination of BIRCH and GMM image segmentation and VGG 19 can effectively Identify the Rice leaf diseases and may provide the support for farmer in rice diseases detection.

VII. REFERENCES

1. Gnanamanickam SS (2009) Rice and Its Importance to Human Life. Prog Biol Con 8: 1-11 Gomez KA (2001) Rice, the grain of culture. The Siam Society, Thailand.
2. Barbedo, Jayme Garcia Arnal, 2013. Digital Image Processing Techniques forDetecting,Quantifying and Classifying Plant Diseases, vol. 2. SpringerPlus.
3. Chauhan, Nagesh Singh. “DBSCAN Clustering Algorithm in Machine Learning.”*KDnuggets*, Accessed 7 July 2022.
4. Dey, Debomit. “DBSCAN Clustering in ML | Density based clustering.” *GeeksforGeeks*, 15 June 2022,Gupta, Aman. “Balanced Iterative Reducing and Clustering using Hierarchies — BIRCH.”*Medium*, 1 June 2021, Accessed 7 July 2022.
5. Narein, Adith, and Aditya L. Rao. “Inception V3 Model Architecture.” *OpenGenus IQ*,/.