IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Grape Leaf Disease Prediction Using Convolutional Neural Work with Supplement Suggestion

[1]Boopathy. E, [2]Prabakaran. S,
Student, Assistant Professor,
Computer Science Engineering, Sasuri College of Engineering and Technology
Computer Science Engineering, Sasuri College of Engineering and Technology, Vijayamangalam, India

Abstract: The prediction of grape leaf diseases using Convolutional Neural Networks (CNNs) represents a transformative approach to modern agricultural management. This system explores the application of CNNs to automatically detect and classify grape leaf diseases from images, aiming to enhance early disease detection, optimize intervention strategies, and improve overall grapevine health. The use of CNNs leverages their ability to learn hierarchical features from image data, enabling the accurate identification of various disease symptoms that may be subtle and difficult to detect manually. The CNN-based model is trained on a diverse dataset of grape leaf images, encompassing various disease conditions and growth stages. The System demonstrates high accuracy in classifying disease types and distinguishing between healthy and affected leaves.

Index Terms **– Grape Leaf Diseases, CNN, PyTorch, Flask**

I. INTRODUCTION

India is among the first ten countries in the world in the production of grape. The major producers of grape are Italy, France, Spain, USA, Turkey, China and Argentina. This crop occupies fifth position amongst fruit crops in India with a production of 1.21 million tonnes (around 2% of world's production of 57.40 million tonnes) from an area of 0.05 million ha. About 80% of the production comes from Maharashtra followed by Karnataka and Tamil Nadu.

Having diseases is quite natural in crops due to changing climatic and environmental conditions. Diseases affect the growth and produce of the crops and often difficult to control. To ensure good quality and high production, it is necessary to have accurate disease diagnosis and control actions to prevent them in time. Grape which is widely grown crop in India and it may be affected by different types of diseases on leaf, stem and fruit. Leaf diseases which are the early symptoms caused due to fungi, bacteria and virus.

So, there is a need to have an automatic system that can be used to detect the type of diseases and to take appropriate actions. These diseases are judged by the farmers through their experience or with the help of experts through naked eye observation which is not accurate and time consuming process. Early detection of disease is then very much needed in the agriculture and horticulture field to increase the yield of the crops. The system have mainly focused on four diseases in grapes causing devastating yield losses in most of the years We have proposed a system that can detect and identify diseases in the leaves of the grape plants and the system give suggestion to improve leaves strengths.

OBJECTIVE OF THE STUDY

Using Convolutional Neural Networks (CNNs) for grape leaf disease prediction aims to achieve several key objectives:

- Accurate Disease Classification:
- Early Disease Detection:
- Timely Diagnosis:
- Enhanced Image Processing

II. LITERATURE REVIEW

Kim et al. (2019): In their study on plant disease classification using deep learning, Kim et al. demonstrated the efficacy of CNNs in identifying various plant diseases, including grape leaf diseases. Their work laid the groundwork for applying CNNs to plant pathology by highlighting the benefits of automated feature extraction and classification from images.

Mohanty et al. (2016): This foundational work on plant disease classification using deep learning models showed that CNNs could achieve high accuracy in identifying plant diseases, including grape diseases. The study emphasized the potential of CNNs in recognizing complex patterns in plant leaves that are indicative of different diseases.

Liu et al. (2020): Liu et al. explored various CNN architectures for plant disease detection, including the use of ResNet and DenseNet models. They found that advanced CNN architectures improved classification accuracy and robustness in detecting grape leaf diseases, highlighting the importance of choosing the right model architecture for specific applications.

Zhang et al. (2021): In their research, Zhang et al. applied transfer learning with pre-trained CNN models to grape leaf disease prediction. They demonstrated that transfer learning could enhance model performance, especially when dealing with limited training data, by leveraging features learned from larger datasets.

Data Quality and Variability: Several studies, such as those by Yang et al. (2020), have highlighted challenges related to image quality, variability in environmental conditions, and the need for large, annotated datasets. Addressing these issues is crucial for improving CNN model performance and reliability.

Generalization and Overfitting: Research by Li et al. (2021) has pointed out the risks of overfitting in CNN models, where models trained on specific datasets may not generalize well to new conditions or different grape varieties. Techniques such as data augmentation and cross-validation are essential for mitigating these risks.

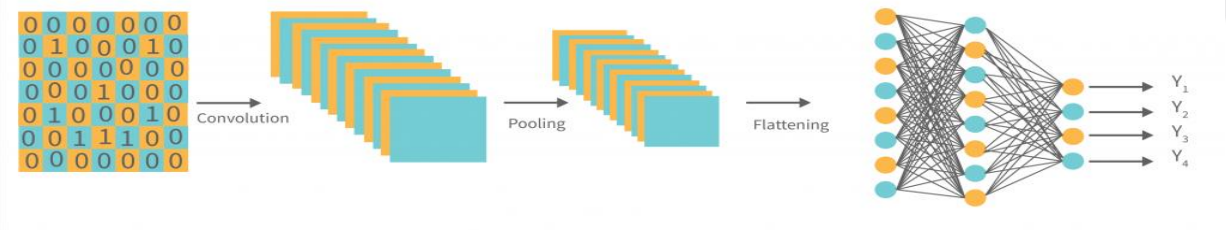
III. RESEARCH METHODOLOGY

CNNs are a type of neural network designed for image and video recognition tasks. They are particularly effective for tasks that involve image classification, object detection, and image segmentation due to their ability to capture spatial and temporal dependencies.

Architecture of CNNs

CNNs typically follow a pattern of stacking convolutional layers, pooling layers, and activation functions. A common architecture includes:

- **Input Layer:** Takes in the raw image data.
- **Convolutional Layer(s):** Extract features from the image.
- **Activation Layer(s):** Apply activation functions like ReLU.
- **Pooling Layer(s):** Downsample the feature maps.
- **Fully Connected Layer(s):** Perform classification or regression based on extracted features.
- **Output Layer:** Provides the final prediction or classification result.



Convolutional Layers

Convolutional layers are designed to automatically and adaptively learn spatial hierarchies of features from input images. The convolution operation involves applying a set of filters (or kernels) to the input image. Each filter extracts specific features, such as edges or textures.

Filter (Kernel):

A small matrix (e.g., 3x3 or 5x5) that slides over the input image to produce feature maps. Each filter extracts specific features from the image, which are then used for further processing in the network. There are three types of filters are

- Edge Detection Filter
- Texture Filter
- Feature Detection Filter

- **Dimensions:** Filters are usually small (e.g., 3x3, 5x5), and they slide across the input image to produce feature maps.
- **Parameters:** Each filter has a set of learnable parameters (weights) that are adjusted during the training process to detect relevant features.

Stride:

Determines how the filter moves across the image. A stride of 1 means the filter moves one pixel at a time. Stride of 2 means filter moves two pixels.

Padding:

Adds extra pixels around the border of the input image to control the spatial dimensions of the output feature maps.

Activation Functions

- **ReLU (Rectified Linear Unit):** The most common activation function used in CNNs. It introduces non-linearity by replacing all negative values in the feature map with zero. Feature map which denotes the input image values.

Pooling Layers

Pooling layers reduce the spatial dimensions of the feature maps and retain the most important information. It has two types are

- **Max Pooling:** Selects the maximum value from each region of the feature map.
- **Average Pooling:** Computes the average value of each region. Typically uses a 2x2 or 3x3 filter with a stride of 2 to reduce the dimensions by half.

Fully Connected Layers

Fully connected (dense) layers are used to make predictions based on the features extracted by the convolutional and pooling layers.Each neuron in a fully connected layer is connected to every neuron in the previous layer. The output is typically a probability distribution over the classes.

3.1 DATA COLLECTION AND PREPROCESSING

DATA PREPARATION

The present investigation sourced its dataset from Kaggle , a publicly accessible online platform. The grape images in the dataset are all of 256 × 256 pixels in size.

Sl. no	Category	Number of images	Leaf symptoms
1	Black Rot	2360	Appear Small brown circular lesions
2	ESCA	2400	Appear dark red or yellow stripes
3	Leaf Blight	2152	Appear many small rounded, polygonal
4	Healthy	2115	–
Total samples		9027	–

Table 1 Sample Grape Leaf Dataset

As shown in Table 1, the group is unbalanced and has a total of 9028 photos from four different classes of grape leaf disease. The dataset comprises both asymptomatic and symptomatic images, about appearances such as black rot, ESCA, and leaf blight symptoms,

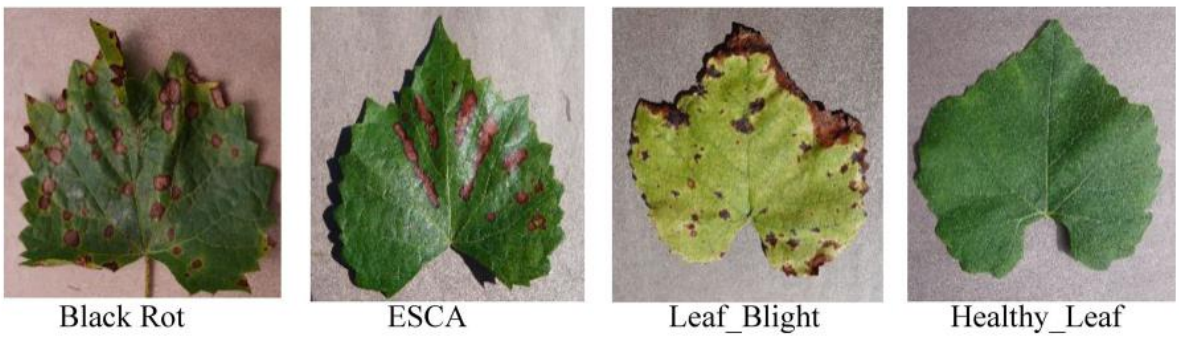


Fig.3.5. Images of several diseased Leaves

In diagnosing grape diseases, the leaf is utilised instead of the flower, fruit, or stem. The persistent presence of grape leaves, in contrast to the ephemeral appearance of blossoms and fruit, accounts for this phenomenon. Furthermore, the leaf exhibits a higher degree of sensitivity towards the overall health of plants and generally imparts more comprehensive information in contrast to the stem. The grape's stem may not promptly exhibit symptoms of illness.

3.2 MODEL SELECTION AND TRAINING

A Convolutional neural network (CNN) is a specialized deep learning algorithm that has been specifically developed for the purpose of analysing visual data, with a particular focus on images. The advent of computer vision has brought about a significant transformation, resulting in notable achievements in various tasks such as image classification, object detection, and image segmentation.

STEPS FOR BUILD Model

1. Start.
2. Build the CNN model:

```
disease_info = pd.read_csv('disease_info.csv' , encoding='cp1252')
supplement_info = pd.read_csv('supplement_info.csv',encoding='cp1252')
model = CNN.CNN(39)
model.load_state_dict(torch.load("plant_disease_model_1_latest.pt"))
model.eval()
```

3. Compile the model:

   a. Specify the optimizer (Adam) with a desired learning rate.

   b. Specify the loss function appropriate for your problem (categorical cross-entropy).

   c. Specify the evaluation metrics to monitor during training (accuracy).

4. Train the model:

```
class CNN(nn.Module):
  def __init__(self, K):
     super(CNN, self).__init__()
     self.conv_layers = nn.Sequential(
# conv1
            kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(32),
        nn.Conv2d(in_channels=32, out_channels=32,
              kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(32),
        nn.MaxPool2d(2),
        # conv2
        nn.Conv2d(in_channels=32, out_channels=64,
              kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(64),
        nn.Conv2d(in_channels=64, out_channels=64,
              kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(64),
        nn.MaxPool2d(2),
        # conv3
        nn.Conv2d(in_channels=64, out_channels=128,
              kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(128),
        nn.Conv2d(in_channels=128, out_channels=128,
              kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(128),
        nn.MaxPool2d(2),
        # conv4
        nn.Conv2d(in_channels=128, out_channels=256,
              kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(256),
        nn.Conv2d(in_channels=256, out_channels=256,
              kernel_size=3, padding=1),
        nn.ReLU(),
        nn.BatchNorm2d(256),
        nn.MaxPool2d(2),
      )
      self.dense_layers = nn.Sequential(
```

```
            nn.Dropout(0.4),
            nn.Linear(50176, 1024),
            nn.ReLU(),
            nn.Dropout(0.4),
            nn.Linear(1024, K),
        )
    def forward(self, X):
        out = self.conv_layers(X)
        # Flatten
        out = out.view(-1, 50176)
        # Fully connected
        out = self.dense_layers(out)
        return out
```

5. Evaluate the model:
   a. Calculate the evaluation metrics on the testing data.
6. Save the trained model for future use.
7. Output the evaluation of test results.
8. End.

At the outset, certain machine learning methodologies commence by extracting features, which are subsequently employed to train a classifier with the aim of automating the classification of leaf diseases. Nevertheless, the process of manually engineering features is a time-consuming endeavour. The model provides a details about the layers followed in CNN.

Convolutional Layers (Conv2d)

Convolutional layers are the core building blocks of a CNN. They apply filters to the input image to detect features such as edges, textures, and patterns. The CNN starts with a series of `Conv2d` layers, each followed by a ReLU activation function and often batch normalization. These layers progressively extract more complex features as the input passes through the network.

Output Shape*:*

Initially, the input has a shape of `[batch_size, 3, 224, 224]`, where `3` is the number of color channels (RGB), and `224x224` is the image size. After each convolution, the number of channels increases while the spatial dimensions (`height` and `width`) typically decrease or remain the same, depending on the stride and padding of the convolution.

ReLU Activation (ReLU)

The ReLU (Rectified Linear Unit) introduces non-linearity into the model, allowing it to learn complex patterns. After every convolutional layer, a ReLU activation function is applied. This ensures that negative values are converted to zero, helping the network to focus on the important features.

Batch Normalization (BatchNorm2d)

Batch normalization stabilizes and accelerates the training process by normalizing the inputs to each layer. This reduces the sensitivity to the initial starting weights. It is applied after the convolution and ReLU layers, maintaining the output's distribution and helping the network generalize better.

Max Pooling Layers (MaxPool2d)

Max pooling layers reduce the spatial dimensions (height and width) of the feature maps. This helps in reducing the computational load and the number of parameters, and it also introduces spatial invariance. The CNN has several MaxPool2d layers interspersed between convolutional layers. Each pooling layer downsamples the feature maps, typically reducing their size by half.

Fully Connected Layers (Linear) or Dense Layer

After the convolutional and pooling layers, the output is flattened into a 1D vector and passed through fully connected layers, which perform the final classification based on the features extracted by the earlier layers.

- The first fully connected layer (Linear-30) has 51,381,248 parameters and outputs 1024 features.
- The second fully connected layer (Linear-33) reduces the 1024 features to the final output, which has 39 classes or probabilities.

Each neuron in a dense layer computes a weighted sum of its inputs, adds a bias, and then applies an activation function (e.g., ReLU or softmax).

$$y_{j=activation(\sum_i w_{ij}x_i+b_j)}$$

Where w_{ij} are the weights, x_i are the inputs, b_j and is the bias.

Dropout Layer

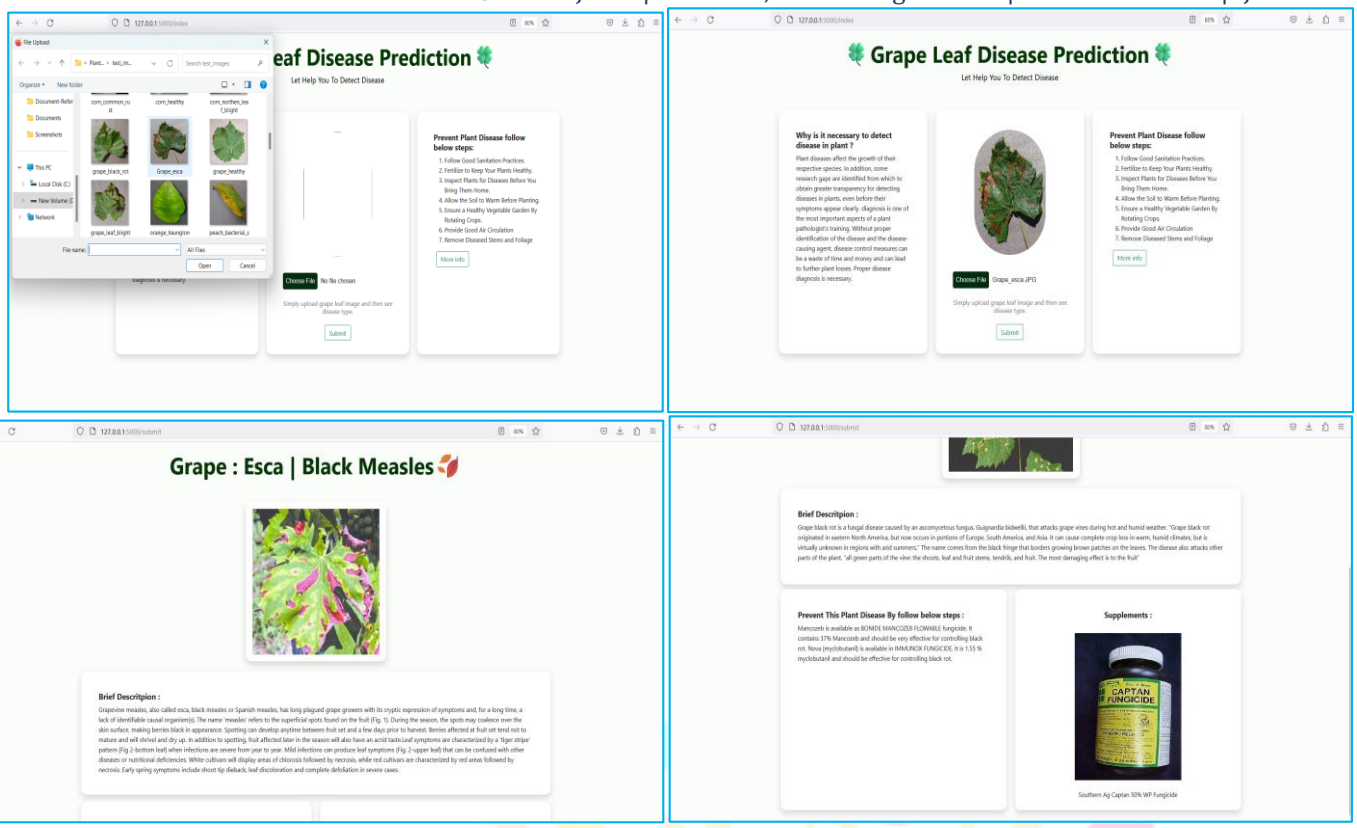
Dropout is used to prevent overfitting by randomly setting a fraction of input units to zero during training. This encourages the network to learn more robust features. A dropout layer is placed before the final fully connected layer.

Parameter Summary

The network has 52,595,399 parameters. All of these are trainable, meaning they will be updated during the training process.

IV. RESULT AND DISCUSSION

An Intel Core i5 processor running Windows 11 with 8 GB of RAM was the system used to implement the code. Anaconda, an integrated Python distribution that streamlines deployment and package management, enabled the code execution. Python served as the primary programming language for writing the code, leveraging its versatility and extensive libraries for deep learning tasks.



Result Module

Result module evaluate the input image with trained models and it shows the output to the user. This module visualize the result to the user. Additionally it provides suggestion for supplements which need for grape leaves growth.

LIMITATION OF FUTURE WORK

When using Convolutional Neural Networks (CNNs) for grape leaf disease prediction, several limitations and challenges can arise. These limitations can affect model performance, scalability, and real-world applicability. Addressing these limitations through future work can significantly enhance the effectiveness and reliability of CNN-based disease prediction systems. Here are some key limitations and potential areas for future work:

- Data Limitations
- Model Limitations
- Model Complexity
- Computational requirements
- Integration and Deployment
- Ethical and privacy

V. CONCLUSION

In conclusion, utilizing Convolutional Neural Networks (CNNs) for grape leaf disease prediction represents a significant advancement in agricultural technology. CNNs offer powerful capabilities for analyzing complex visual patterns and identifying diseases from leaf images with high accuracy. However, several limitations must be addressed to fully realize the potential of CNNs in this domain.

REFERENCES

[1] Kim, J., et al. (2019). "Plant Disease Classification Using Deep Learning Models." Mohanty, S. P., et al. (2016). "Using Deep Learning for Plant Disease Classification."
[2]Liu, Y., et al. (2020). "Comparison of CNN Architectures for Plant Disease Detection."
[3] Zhang, Z., et al. (2021). "Transfer Learning for Grape Leaf Disease Prediction Using Pre-trained CNNs."