|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

Mini Project

|  |  |
| --- | --- |
| Semester | B.E. Semester VIII – Computer Engineering |
| Subject | Social Media Analytics |
| Subject Professor In-charge | Prof. Amit Alyani |
| Academic Year | 2024-25 |

|  |  |
| --- | --- |
| Student Names | Deep Salunkhe 21102A0014 |
|  | Sahil Pokharkar 21102A0006 |
|  | Pranav Redij 21102A0002 |
|  | Omkar Patil 21102A0003 |

**Title: Plant Disease Detection Using ResNet9**

Our mini-project focuses on the detection of plant diseases using deep learning. Leveraging a convolutional neural network (CNN) architecture—**ResNet9**—we trained a classifier on an augmented dataset of plant disease images. The final system is capable of identifying various plant diseases with high accuracy, offering a potential tool for modern agriculture.

Model Code: <https://github.com/deepsalunkhee/PlantDiseaseDetection/blob/master/models/plant-disease-classification.ipynb>

Web app Code: <https://github.com/deepsalunkhee/plantDiseaseDetection>

Web app: <https://pdd.deepsalunkhee.com/>

**Dataset:**

The dataset used in this project consists of labeled images of plant leaves. Each image represents either a healthy leaf or a leaf affected by a specific plant disease. The dataset includes multiple classes(Total 38) such as:

* Healthy
* Bacterial diseases
* Fungal infections
* Viral diseases

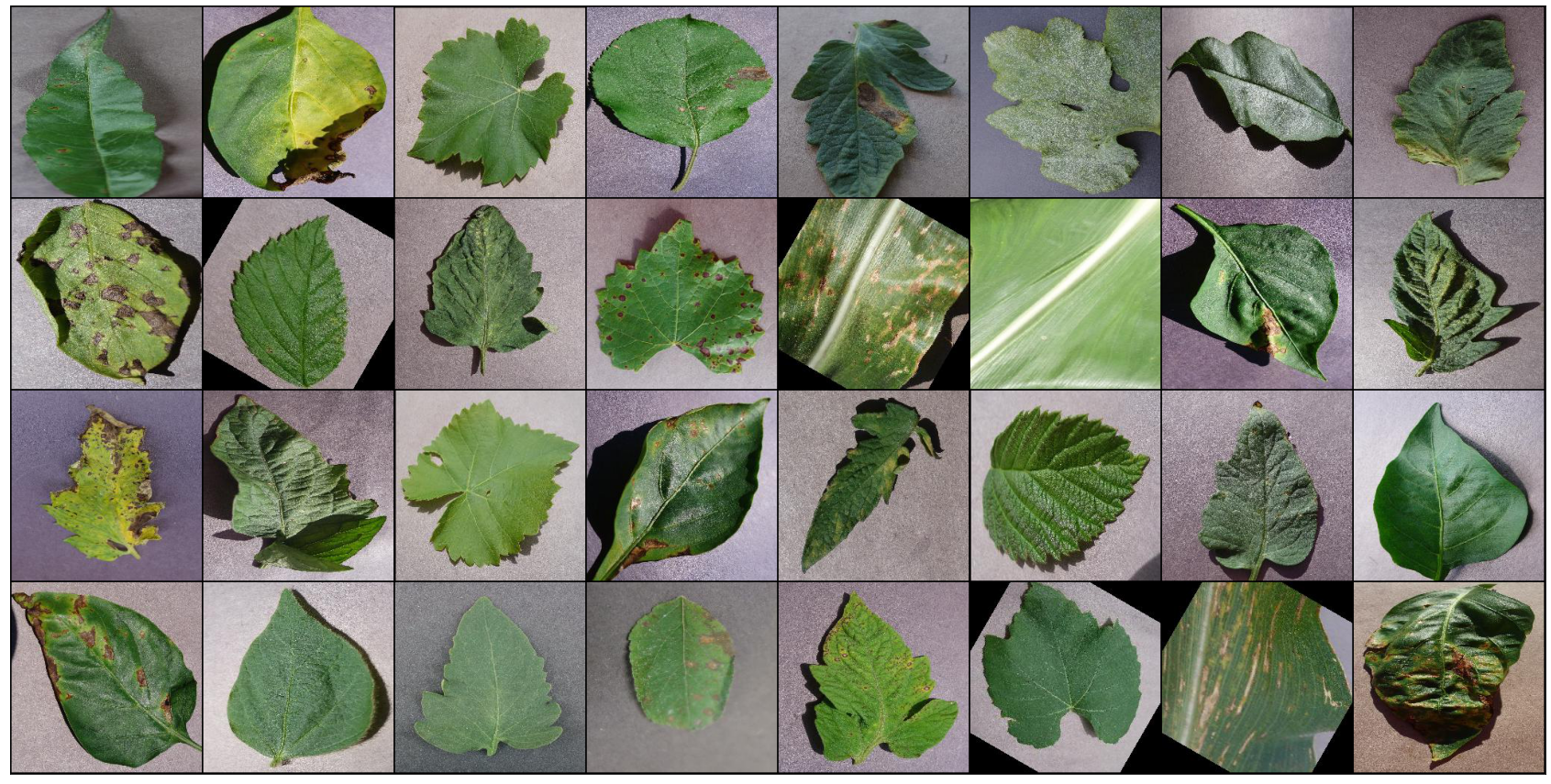
Each image is in color (RGB) and shows variations in lighting, angle, and background, making the dataset more realistic and challenging—ideal for training deep learning models.

The dataset is divided into three main parts:

* Training set: Used to teach the model to recognize patterns.
* Validation set: Used during training to monitor the model’s performance and avoid overfitting.
* Test set or inference images: Used to test the model’s final performance or demonstrate its predictions.

To increase the model's ability to generalize, data augmentation techniques were applied. These include:

* Horizontal and vertical flipping
* Random rotations and cropping
* Adjustments in brightness and contrast
* Random zooming

****

**Model Architecture: ResNet9:**

# Architecture for training

# convolution block with BatchNormalization

**def** ConvBlock(in\_channels, out\_channels, pool**=False**):

    layers **=** [nn.Conv2d(in\_channels, out\_channels, kernel\_size**=**3, padding**=**1),

             nn.BatchNorm2d(out\_channels),

             nn.ReLU(inplace**=True**)]

**if** pool:

        layers.append(nn.MaxPool2d(4))

**return** nn.Sequential(**\***layers)

# resnet architecture

**class** ResNet9(ImageClassificationBase):

**def** \_\_init\_\_(self, in\_channels, num\_diseases):

        super().\_\_init\_\_()

        self.conv1 **=** ConvBlock(in\_channels, 64)

        self.conv2 **=** ConvBlock(64, 128, pool**=True**) # out\_dim : 128 x 64 x 64

        self.res1 **=** nn.Sequential(ConvBlock(128, 128), ConvBlock(128, 128))

        self.conv3 **=** ConvBlock(128, 256, pool**=True**) # out\_dim : 256 x 16 x 16

        self.conv4 **=** ConvBlock(256, 512, pool**=True**) # out\_dim : 512 x 4 x 44

        self.res2 **=** nn.Sequential(ConvBlock(512, 512), ConvBlock(512, 512))

        self.classifier **=** nn.Sequential(nn.MaxPool2d(4),

                                       nn.Flatten(),

                                       nn.Linear(512, num\_diseases))

**def** forward(self, xb): # xb is the loaded batch

        out **=** self.conv1(xb)

        out **=** self.conv2(out)

        out **=** self.res1(out) **+** out

        out **=** self.conv3(out)

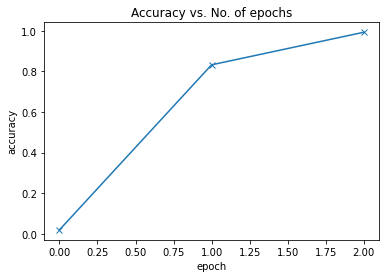
        out **=** self.conv4(out)

        out **=** self.res2(out) **+** out

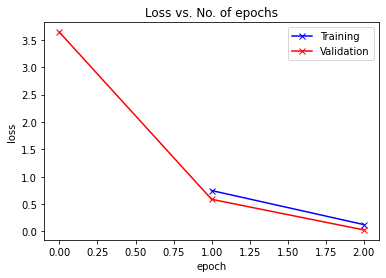
        out **=** self.classifier(out)

**return** out

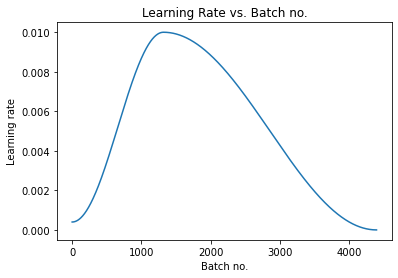
**Validation Accuracy**

****

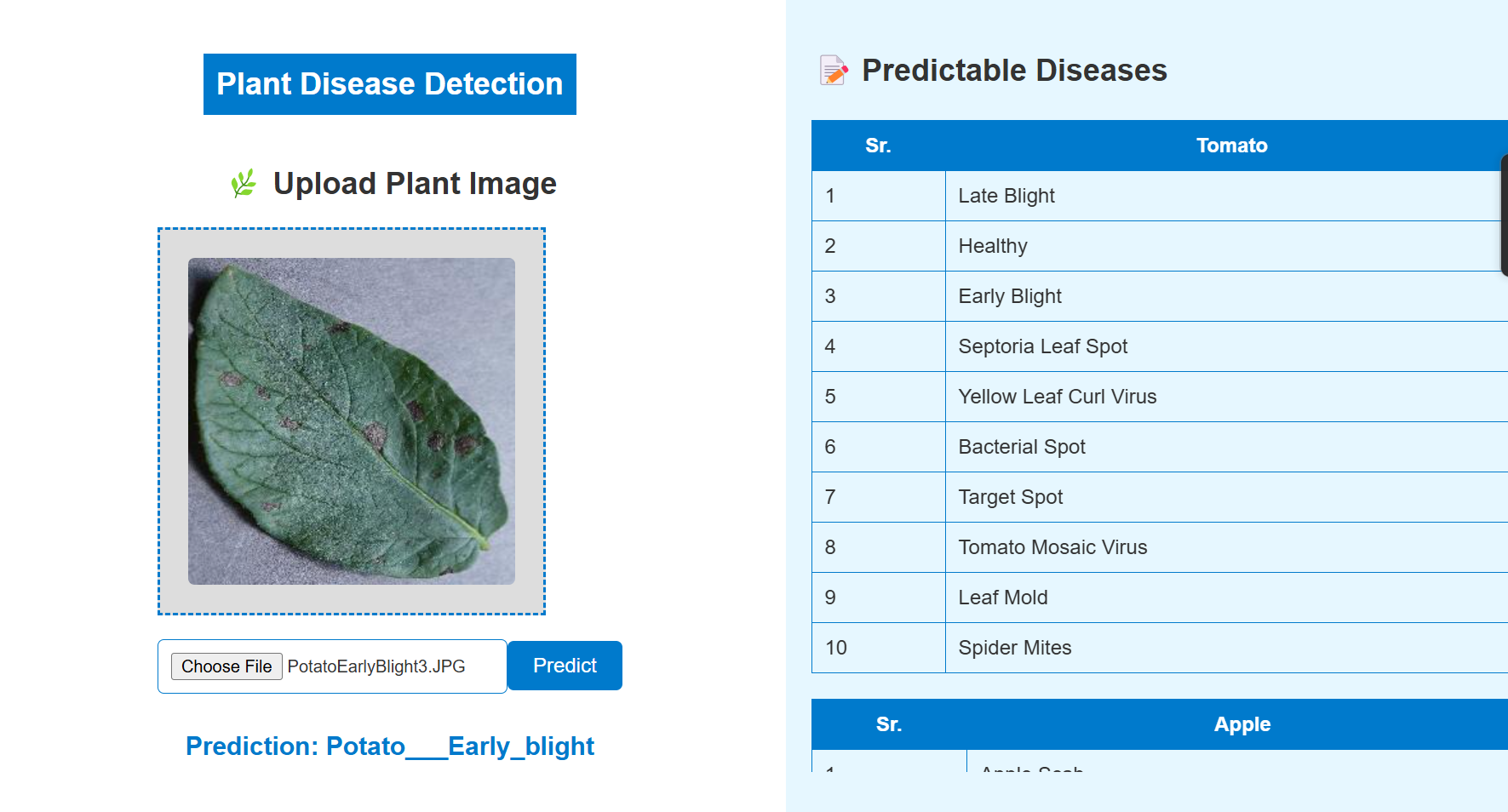
**Validation loss:**

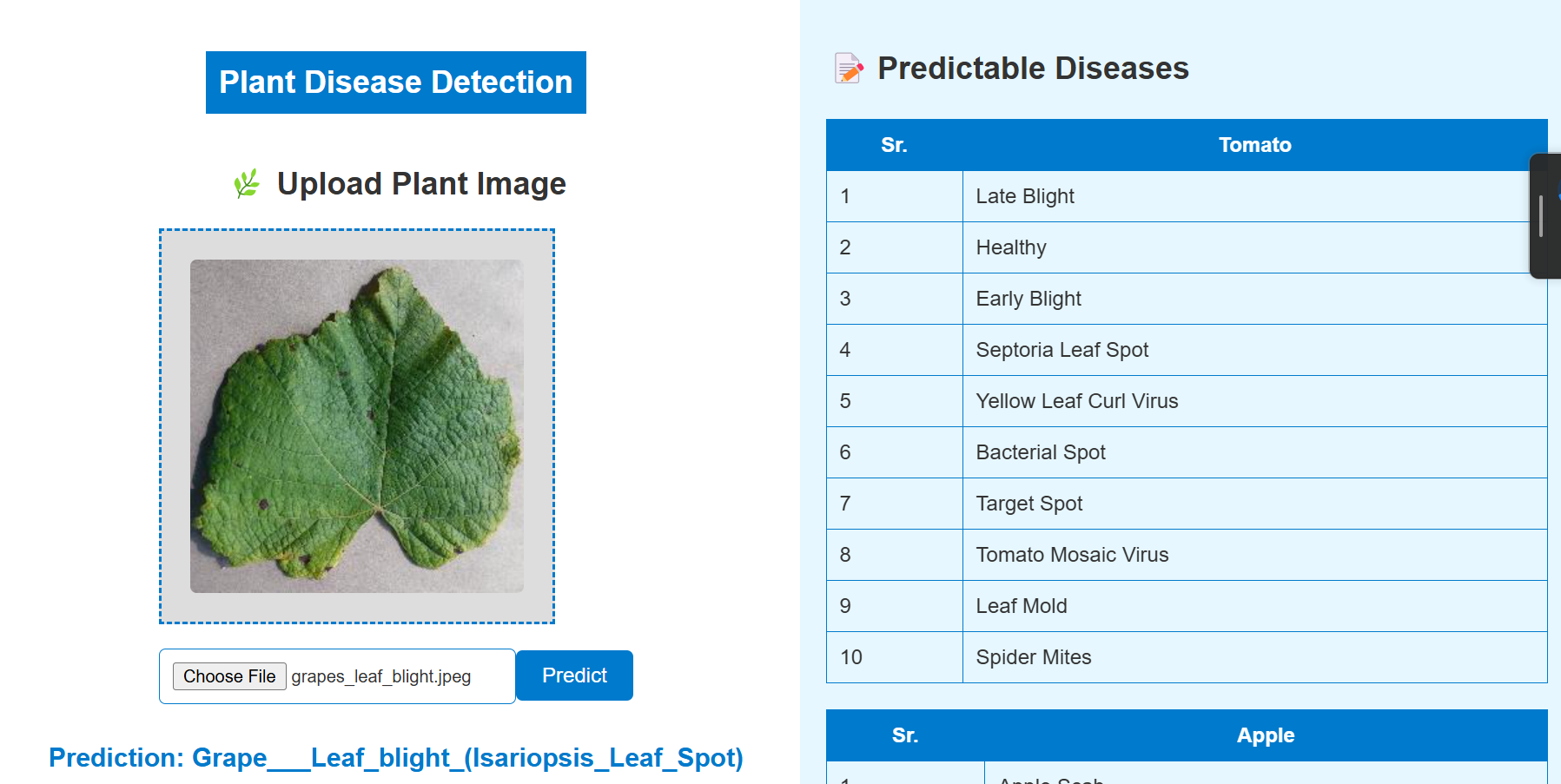
****

**Learning Rate overtime**

****

**Output:**

****

****