Import Dependencies

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import torch
from torchvision import datasets, transforms, models  # datsets  , transforms
from torch.utils.data.sampler import SubsetRandomSampler
import torch.nn as nn
import torch.nn.functional as F
from datetime import datetime
```

```
%load_ext nb_black
```

```
<IPython.core.display.Javascript object>
```

Import Dataset

Dataset Link (Plant Vliiage Dataset):
https://data.mendeley.com/datasets/tywbtsjrjv/1

```
transform = transforms.Compose(
    [transforms.Resize(255), transforms.CenterCrop(224), transforms.ToTensor()]
)
```

```
<IPython.core.display.Javascript object>
```

```
dataset = datasets.ImageFolder("Dataset", transform=transform)
```

```
<IPython.core.display.Javascript object>
```

```
dataset
```

```
Dataset ImageFolder
    Number of datapoints: 61486
    Root Location: Dataset
    Transforms (if any): Compose(
                             Resize(size=255, interpolation=PIL.Image.BILINEAR)
                             CenterCrop(size=(224, 224))
                             ToTensor()
                         )
    Target Transforms (if any): None
```

```
<IPython.core.display.Javascript object>
```

```
indices = list(range(len(dataset)))
```

```
<IPython.core.display.Javascript object>
```

```
split = int(np.floor(0.85 * len(dataset)))  # train_size
```

```
<IPython.core.display.Javascript object>
```

```
validation = int(np.floor(0.70 * split))  # validation
```

```
<IPython.core.display.Javascript object>
```

```
print(0, validation, split, len(dataset))
```

```
0 36584 52263 61486
```

```
<IPython.core.display.Javascript object>
```

```
print(f"length of train size :{validation}")
print(f"length of validation size :{split - validation}")
print(f"length of test size :{len(dataset)-validation}")
```

```
length of train size :36584
length of validation size :15679
length of test size :24902



<IPython.core.display.Javascript object>
```

```
np.random.shuffle(indices)
```

```
<IPython.core.display.Javascript object>
```

Split into Train and Test

```
train_indices, validation_indices, test_indices = (
    indices[:validation],
    indices[validation:split],
    indices[split:],
)
```

```
<IPython.core.display.Javascript object>
```

```
train_sampler = SubsetRandomSampler(train_indices)
validation_sampler = SubsetRandomSampler(validation_indices)
test_sampler = SubsetRandomSampler(test_indices)
```

```
<IPython.core.display.Javascript object>
```

```
targets_size = len(dataset.class_to_idx)
```

```
<IPython.core.display.Javascript object>
```

Model

Convolution Aithmetic Equation : (W - F + 2P) / S + 1
W = Input Size
F = Filter Size
P = Padding Size
S = Stride

Transfer Learning

```
# model = models.vgg16(pretrained=True)
```

```
# for params in model.parameters():
#     params.requires_grad = False
```

```
# model
```

```
# n_features = model.classifier[0].in_features
# n_features
```

```
# model.classifier = nn.Sequential(
#     nn.Linear(n_features, 1024),
#     nn.ReLU(),
#     nn.Dropout(0.4),
#     nn.Linear(1024, targets_size),
# )
```

```
# model
```

Original Modeling

```
class CNN(nn.Module):
    def __init__(self, K):
        super(CNN, self).__init__()
        self.conv_layers = nn.Sequential(
            # conv1
            nn.Conv2d(in_channels=3, out_channels=32, 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
```

```
<IPython.core.display.Javascript object>
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cpu


<IPython.core.display.Javascript object>
```

```
device = "cpu"
```

```
<IPython.core.display.Javascript object>
```

```
model = CNN(targets_size)
```

```
<IPython.core.display.Javascript object>
```

```
model.to(device)
```

```
CNN(
  (conv_layers): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
```

```
    (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (7): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (10): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU()
    (12): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (14): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU()
    (16): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (17): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU()
    (19): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (20): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (21): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU()
    (23): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (24): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU()
    (26): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (dense_layers): Sequential(
    (0): Dropout(p=0.4, inplace=False)
    (1): Linear(in_features=50176, out_features=1024, bias=True)
    (2): ReLU()
    (3): Dropout(p=0.4, inplace=False)
    (4): Linear(in_features=1024, out_features=39, bias=True)
  )
)
```

```
<IPython.core.display.Javascript object>
```

```
from torchsummary import summary

summary(model, (3, 224, 224))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 224, 224]             896
              ReLU-2         [-1, 32, 224, 224]               0
       BatchNorm2d-3         [-1, 32, 224, 224]              64
            Conv2d-4         [-1, 32, 224, 224]           9,248
              ReLU-5         [-1, 32, 224, 224]               0
       BatchNorm2d-6         [-1, 32, 224, 224]              64
         MaxPool2d-7         [-1, 32, 112, 112]               0
            Conv2d-8         [-1, 64, 112, 112]          18,496
              ReLU-9         [-1, 64, 112, 112]               0
      BatchNorm2d-10         [-1, 64, 112, 112]             128
           Conv2d-11         [-1, 64, 112, 112]          36,928
             ReLU-12         [-1, 64, 112, 112]               0
      BatchNorm2d-13         [-1, 64, 112, 112]             128
        MaxPool2d-14           [-1, 64, 56, 56]               0
           Conv2d-15          [-1, 128, 56, 56]          73,856
             ReLU-16          [-1, 128, 56, 56]               0
      BatchNorm2d-17          [-1, 128, 56, 56]             256
           Conv2d-18          [-1, 128, 56, 56]         147,584
             ReLU-19          [-1, 128, 56, 56]               0
      BatchNorm2d-20          [-1, 128, 56, 56]             256
        MaxPool2d-21          [-1, 128, 28, 28]               0
           Conv2d-22          [-1, 256, 28, 28]         295,168
             ReLU-23          [-1, 256, 28, 28]               0
      BatchNorm2d-24          [-1, 256, 28, 28]             512
           Conv2d-25          [-1, 256, 28, 28]         590,080
             ReLU-26          [-1, 256, 28, 28]               0
      BatchNorm2d-27          [-1, 256, 28, 28]             512
        MaxPool2d-28          [-1, 256, 14, 14]               0
          Dropout-29                [-1, 50176]               0
           Linear-30                 [-1, 1024]      51,381,248
             ReLU-31                 [-1, 1024]               0
          Dropout-32                 [-1, 1024]               0
           Linear-33                   [-1, 39]          39,975
================================================================
Total params: 52,595,399
Trainable params: 52,595,399
Non-trainable params: 0
```

```
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 143.96
Params size (MB): 200.64
Estimated Total Size (MB): 345.17
----------------------------------------------------------------



<IPython.core.display.Javascript object>
```

```
criterion = nn.CrossEntropyLoss()  # this include softmax + cross entropy loss
optimizer = torch.optim.Adam(model.parameters())
```

```
<IPython.core.display.Javascript object>
```

Batch Gradient Descent

```
def batch_gd(model, criterion, train_loader, test_laoder, epochs):
    train_losses = np.zeros(epochs)
    test_losses = np.zeros(epochs)

    for e in range(epochs):
        t0 = datetime.now()
        train_loss = []
        for inputs, targets in train_loader:
            inputs, targets = inputs.to(device), targets.to(device)

            optimizer.zero_grad()

            output = model(inputs)

            loss = criterion(output, targets)

            train_loss.append(loss.item())  # torch to numpy world

            loss.backward()
            optimizer.step()

        train_loss = np.mean(train_loss)

        validation_loss = []

        for inputs, targets in validation_loader:

            inputs, targets = inputs.to(device), targets.to(device)
```

```
            output = model(inputs)

            loss = criterion(output, targets)

            validation_loss.append(loss.item())  # torch to numpy world

        validation_loss = np.mean(validation_loss)

        train_losses[e] = train_loss
        validation_losses[e] = validation_loss

        dt = datetime.now() - t0

        print(
            f"Epoch : {e+1}/{epochs} Train_loss:{train_loss:.3f} Test_loss:
{validation_loss:.3f} Duration:{dt}"
        )

    return train_losses, validation_losses
```

```
<IPython.core.display.Javascript object>
```

```
device = "cpu"
```

```
<IPython.core.display.Javascript object>
```

```
batch_size = 64
train_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=train_sampler
)
test_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=test_sampler
)
validation_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=validation_sampler
)
```

```
<IPython.core.display.Javascript object>
```

```
train_losses, validation_losses = batch_gd(
    model, criterion, train_loader, validation_loader, 5
)
```

```
<IPython.core.display.Javascript object>
```

Save the Model

```
# torch.save(model.state_dict() , 'plant_disease_model_1.pt')
```

```
<IPython.core.display.Javascript object>
```

Load Model

```
targets_size = 39
model = CNN(targets_size)
model.load_state_dict(torch.load("plant_disease_model_1_latest.pt"))
model.eval()
```

```
CNN(
  (conv_layers): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (7): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (10): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU()
    (12): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
```

```
ceil_mode=False)
    (14): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU()
    (16): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (17): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU()
    (19): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (20): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (21): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU()
    (23): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (24): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU()
    (26): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (dense_layers): Sequential(
    (0): Dropout(p=0.4, inplace=False)
    (1): Linear(in_features=50176, out_features=1024, bias=True)
    (2): ReLU()
    (3): Dropout(p=0.4, inplace=False)
    (4): Linear(in_features=1024, out_features=39, bias=True)
  )
)
```

```
# %matplotlib notebook
```

Plot the loss

```
plt.plot(train_losses , label = 'train_loss')
plt.plot(validation_losses , label = 'validation_loss')
plt.xlabel('No of Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Accuracy

```
def accuracy(loader):
    n_correct = 0
    n_total = 0

    for inputs, targets in loader:
        inputs, targets = inputs.to(device), targets.to(device)

        outputs = model(inputs)

        _, predictions = torch.max(outputs, 1)

        n_correct += (predictions == targets).sum().item()
        n_total += targets.shape[0]

    acc = n_correct / n_total
    return acc
```

```
<IPython.core.display.Javascript object>
```

```
train_acc = accuracy(train_loader)
test_acc = accuracy(test_loader)
validation_acc = accuracy(validation_loader)
```

```
print(
    f"Train Accuracy : {train_acc}\nTest Accuracy : {test_acc}\nValidation
Accuracy : {validation_acc}"
)
```

```
Train Accuracy : 96.7
Test Accuracy : 98.9
Validation Accuracy : 98.7



<IPython.core.display.Javascript object>
```

Single Image Prediction

```
transform_index_to_disease = dataset.class_to_idx
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-9-0e3bd74576a2> in <module>
----> 1 transform_index_to_disease = dataset.class_to_idx


NameError: name 'dataset' is not defined



<IPython.core.display.Javascript object>
```

```
transform_index_to_disease = dict(
    [(value, key) for key, value in transform_index_to_disease.items()]
)  # reverse the index
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-10-1fe109ff4fe8> in <module>
      1 transform_index_to_disease = dict(
----> 2     [(value, key) for key, value in transform_index_to_disease.items()]
      3 )  # reverse the index


NameError: name 'transform_index_to_disease' is not defined



<IPython.core.display.Javascript object>
```

```
data = pd.read_csv("disease_info.csv", encoding="cp1252")
```

```
from PIL import Image
import torchvision.transforms.functional as TF
```

```
def single_prediction(image_path):
    image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
    print("Original : ", image_path[12:-4])
    pred_csv = data["disease_name"][index]
    print(pred_csv)
```

```
single_prediction("test_images/Apple_ceder_apple_rust.JPG")
```

```
Original :  Apple_ceder_apple_rust
Apple : Cedar rust
```

Wrong Prediction

```
single_prediction("test_images/Apple_scab.JPG")
```

```
Original :  Apple_scab
Tomato : Septoria Leaf Spot
```

```
single_prediction("test_images/Grape_esca.JPG")
```

```
Original :  Grape_esca
Grape : Esca | Black Measles
```

```
single_prediction("test_images/apple_black_rot.JPG")
```

```
Original :  apple_black_rot
Pepper bell : Healthy
```

```
single_prediction("test_images/apple_healthy.JPG")
```

```
Original :  apple_healthy
Apple : Healthy
```

```
single_prediction("test_images/background_without_leaves.jpg")
```

```
Original :  background_without_leaves
Background Without Leaves
```

```
single_prediction("test_images/blueberry_healthy.JPG")
```

```
Original :  blueberry_healthy
Blueberry : Healthy
```

```
single_prediction("test_images/cherry_healthy.JPG")
```

```
Original :  cherry_healthy
Cherry : Healthy
```

```
single_prediction("test_images/cherry_powdery_mildew.JPG")
```

```
Original :  cherry_powdery_mildew
Cherry : Powdery Mildew
```

```
single_prediction("test_images/corn_cercospora_leaf.JPG")
```

```
Original :  corn_cercospora_leaf
Corn : Cercospora Leaf Spot | Gray Leaf Spot
```

```
single_prediction("test_images/corn_common_rust.JPG")
```

```
Original :  corn_common_rust
Corn : Common Rust
```

```
single_prediction("test_images/corn_healthy.jpg")
```

```
Original :  corn_healthy
Corn : Healthy
```

```
single_prediction("test_images/corn_northen_leaf_blight.JPG")
```

```
Original :  corn_northen_leaf_blight
Corn : Northern Leaf Blight
```

```
single_prediction("test_images/grape_black_rot.JPG")
```

```
Original :  grape_black_rot
Grape : Black Rot
```

```
single_prediction("test_images/grape_healthy.JPG")
```

```
Original :  grape_healthy
Grape : Healthy
```

```
single_prediction("test_images/grape_leaf_blight.JPG")
```

```
Original :  grape_leaf_blight
Grape : Leaf Blight | Isariopsis Leaf Spot
```

```
single_prediction("test_images/orange_haunglongbing.JPG")
```

```
Original :  orange_haunglongbing
Orange : Haunglongbing | Citrus Greening
```

```
single_prediction("test_images/peach_bacterial_spot.JPG")
```

```
Original :  peach_bacterial_spot
Peach : Bacterial Spot
```

```
single_prediction("test_images/peach_healthy.JPG")
```

```
Original :  peach_healthy
Peach : Healthy
```

```
single_prediction("test_images/pepper_bacterial_spot.JPG")
```

```
Original :  pepper_bacterial_spot
Pepper bell : Healthy
```

```
single_prediction("test_images/pepper_bell_healthy.JPG")
```

```
Original :  pepper_bell_healthy
Pepper bell : Healthy
```

```
single_prediction("test_images/potato_early_blight.JPG")
```

```
Original :  potato_early_blight
Potato : Early Blight
```

```
single_prediction("test_images/potato_healthy.JPG")
```

```
Original :  potato_healthy
Potato : Healthy
```

```
single_prediction("test_images/potato_late_blight.JPG")
```

```
Original :  potato_late_blight
Potato : Late Blight
```

```
single_prediction("test_images/raspberry_healthy.JPG")
```

```
Original :  raspberry_healthy
Raspberry : Healthy
```

```
single_prediction("test_images/soyaben healthy.JPG")
```

```
Original :  soyaben healthy
Soybean : Healthy
```

```
single_prediction("test_images/potato_late_blight.JPG")
```

```
Original :  potato_late_blight
Potato : Late Blight
```

```
single_prediction("test_images/squash_powdery_mildew.JPG")
```

```
Original :  squash_powdery_mildew
Squash : Powdery Mildew
```

```
single_prediction("test_images/starwberry_healthy.JPG")
```

```
Original :  starwberry_healthy
Strawberry : Healthy
```

```
single_prediction("test_images/starwberry_leaf_scorch.JPG")
```

```
Original :  starwberry_leaf_scorch
Strawberry : Leaf Scorch
```

```
single_prediction("test_images/tomato_bacterial_spot.JPG")
```

```
Original :  tomato_bacterial_spot
Tomato : Early Blight
```

```
single_prediction("test_images/tomato_early_blight.JPG")
```

```
Original :  tomato_early_blight
Tomato : Early Blight
```

```
single_prediction("test_images/tomato_healthy.JPG")
```

```
Original :  tomato_healthy
Tomato : Healthy
```

```
single_prediction("test_images/tomato_late_blight.JPG")
```

```
Original :  tomato_late_blight
Tomato : Late Blight
```

```
single_prediction("test_images/tomato_leaf_mold.JPG")
```

```
Original :  tomato_leaf_mold
Tomato : Leaf Mold
```

```
single_prediction("test_images/tomato_mosaic_virus.JPG")
```

```
Original :  tomato_mosaic_virus
Tomato : Mosaic Virus
```

```
single_prediction("test_images/tomato_septoria_leaf_spot.JPG")
```

```
Original :  tomato_septoria_leaf_spot
Tomato : Septoria Leaf Spot
```

```
single_prediction("test_images/tomato_spider_mites_two_spotted_spider_mites.JPG")
```

```
Original :  tomato_spider_mites_two_spotted_spider_mites
Tomato : Spider Mites | Two-Spotted Spider Mite
```

```
single_prediction("test_images/tomato_target_spot.JPG")
```

```
Original :  tomato_target_spot
Tomato : Target Spot
```

```
single_prediction("test_images/tomato_yellow_leaf_curl_virus.JPG")
```

```
Original :  tomato_yellow_leaf_curl_virus
Tomato : Yellow Leaf Curl Virus
```