PLANT DISEASE DETECTION USING LEAF IMAGE CLASSIFICATION

ORALGAZIYEV ELDAR, BDA-2201

19.11.2024

PROJECT GOAL:

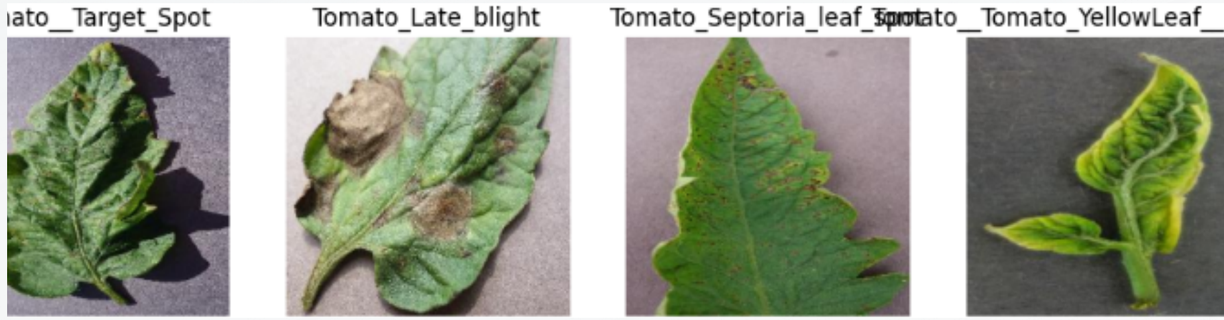
Creating a model for classifying images into categories.

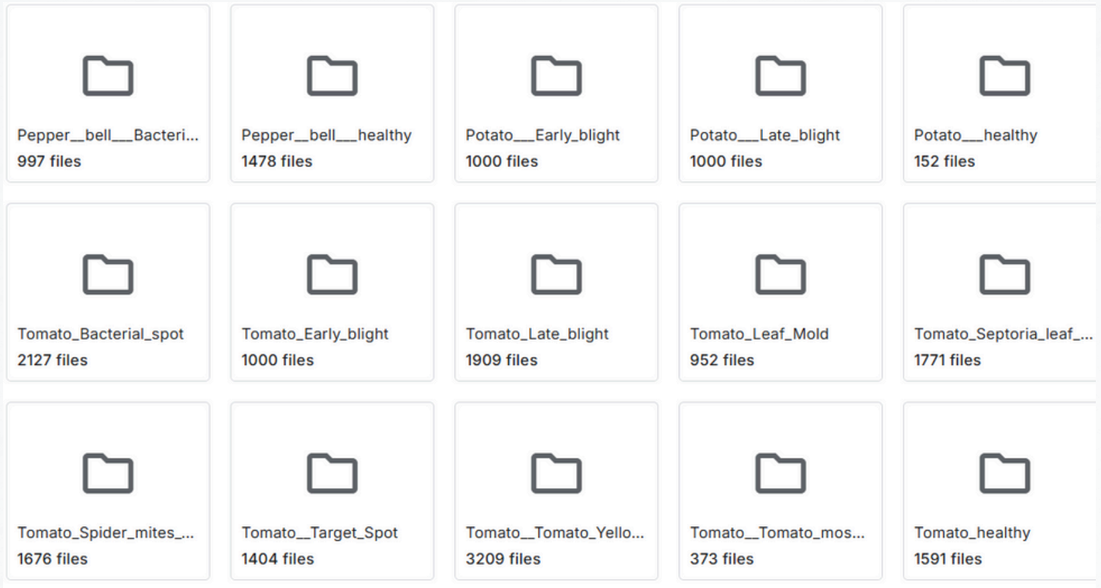
use the ResNet18 neural network to improve classification accuracy.

DATA

The dataset is called PlantVillage and was taken from Kaggle. (https://www.kaggle.com/datasets/emmarex/plantdisease)

It has 15 different classes that store photos of leaves of different states






```
counts = Counter(labels)
print("Category distribution:", dict(counts))
```

Executed at 2024.11.19 00:58:43 in 49ms

```
Category distribution: {0: 997, 1: 1478, 2: 1000, 3: 152, 4: 1000, 5: 2127, 6: 1000, 7:
1591, 8: 1909, 9: 952, 10: 1771, 11: 1676, 12: 1404, 13: 373, 14: 3208}
```

01

RESIZING

All images are reduced to 128x128 size to unify them for the model and speed up work

02

CONVERSION TO TENSORS

Convert images to a format that PyTorch works with.

03

NORMALIZATION

Bringing pixel values into a range suitable for ResNet18

```
transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

RESNET18

- ResNet18 is a pre-trained neural network that is already trained on a large set of images (ImageNet).

- We use it by replacing the last layer with ours so that the network classifies our data.

```
baseline_model = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
baseline_model.fc = nn.Linear(baseline_model.fc.in_features, len(set(labels)))
baseline_model = baseline_model.to(device)
```

Executed at 2024.11.19 00:59:03 in 315ms

ENHANCED MODEL

- First, the input data is "flattened" into a one-dimensional vector (nn.Flatten()).
- Then they pass through a fully connected layer, are activated by the ReLU function, and some neurons are “turned off” for regularization (Dropout).
- Finally, the data is converted into probabilities or logits corresponding to each of the classes through the output layer (nn.Linear(256, num_classes)).

```
class EnhancedModel(nn.Module):
    def __init__(self, base_model, num_classes):
        super().__init__()
        self.base = nn.Sequential(*list(base_model.children())[:-1])
        self.fc = nn.Sequential(
            nn.Flatten(),
            nn.Linear(base_model.fc.in_features, 256),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(256, num_classes)
        )

    def forward(self, x):
        x = self.base(x)
        return self.fc(x)

enhanced_model = EnhancedModel(baseline_model, len(set(labels))).to(device)
```

BASELINE VS ENHACED MODEL

```
baseline_metrics = validate_model(trained_baseline, test_loader, nn.CrossEntropyLoss())
print("Baseline Model Metrics:", baseline_metrics)
```

Executed at 2024.11.19 00:54:06 in 14s 250ms

```
Baseline Model Metrics: {'loss': 0.26943516147922175, 'accuracy': 0.9341085271317829, 'f1':
 0.9339816767316406, 'precision': 0.9383141358907013, 'recall': 0.9341085271317829}
```

```
enhanced_metrics = validate_model(trained_enhanced, test_loader, nn.CrossEntropyLoss())
print("Enhanced Model Metrics:", enhanced_metrics)
```

Executed at 2024.11.19 01:12:21 in 13s 64ms

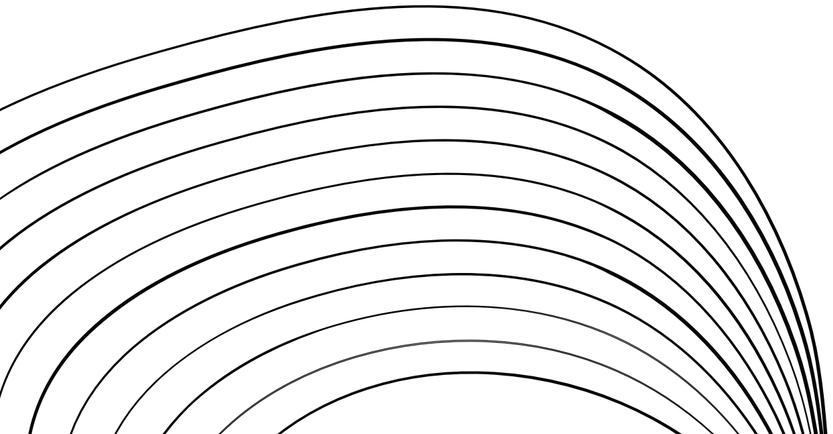
```
Enhanced Model Metrics: {'loss': 0.29702675692708225, 'accuracy': 0.940406976744186, 'f1':
0.94051588869709 99, 'precision': 0.9471288603468063, 'recall': 0.940406976744186}
```

EXPANDING CLASS

After analyzing the dataset, a small number of images were discovered in Potato_healthy. So I extended this class by help of “Potato Leaf (Healthy and Late Blight)” dataset

```
Category distribution: {0: 997, 1: 1478, 2: 1000, 3: 152, 4: 1000, 5: 2127, 6: 1000, 7:
1591, 8: 1909, 9: 952, 10: 1771, 11: 1676, 12: 1404, 13: 373, 14: 3208}

 Category distribution: {0: 997, 1: 1478, 2: 1000, 3: 445, 4: 1000, 5: 2127, 6: 1000, 7:
 1591, 8: 1909, 9: 952, 10: 1771, 11: 1676, 12: 1404, 13: 373, 14: 3208}
```



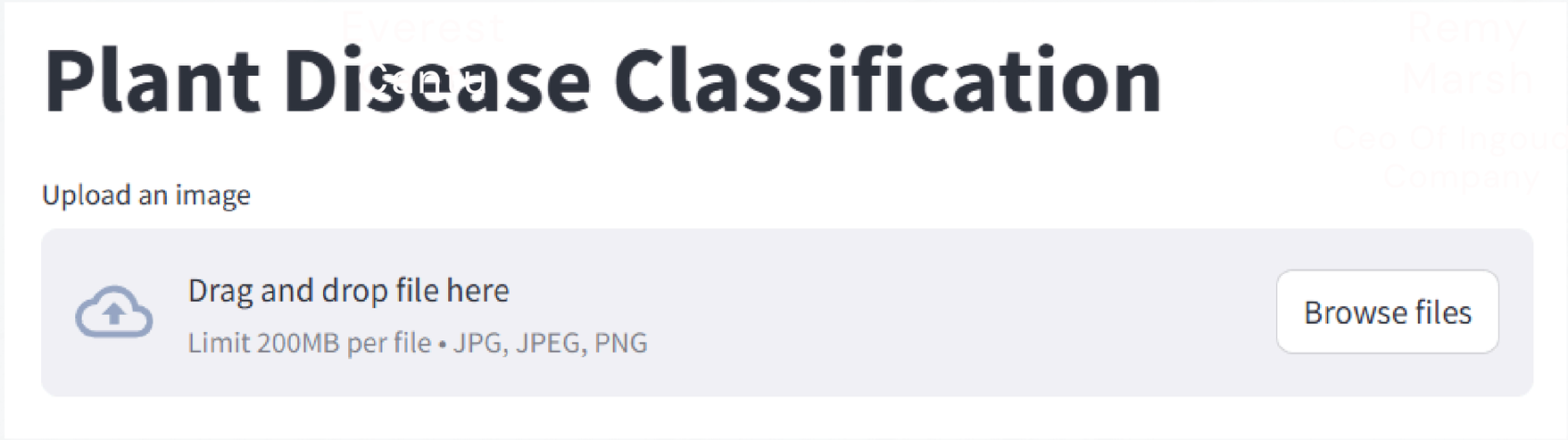
```
new_metrics = validate_model(trained_model, test_loader, nn.CrossEntropyLoss())
print("Metrics after adding new data:", new_metrics)
Executed at 2024.11.18 23:55:17 in 15s 879ms

 Metrics after adding new data: {'loss': 0.098912938279681 84, 'accuracy': 0
 .9732505373775974, 'f1': 0.9731485359974333, 'precision': 0.9738364963377927, 'recall':
 0.9732505373775974}
```

APP

- **To simplify model testing, a custom web interface system was developed using Streamlit.**

- **Streamlit allows you to quickly deploy an application to interact with the model, providing an easy-to-use interface for uploading images and getting predictions.**



THANK'S FOR WATCHING