식물 병해 이미지 분류

팀 : 1인이하

이호준

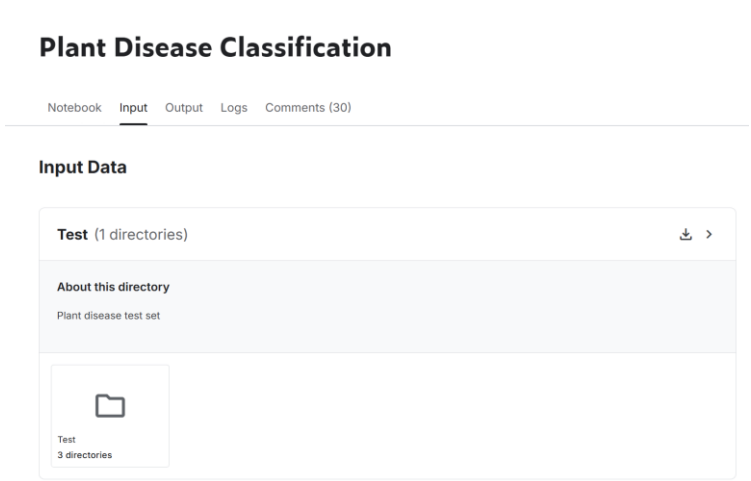
개발 동기

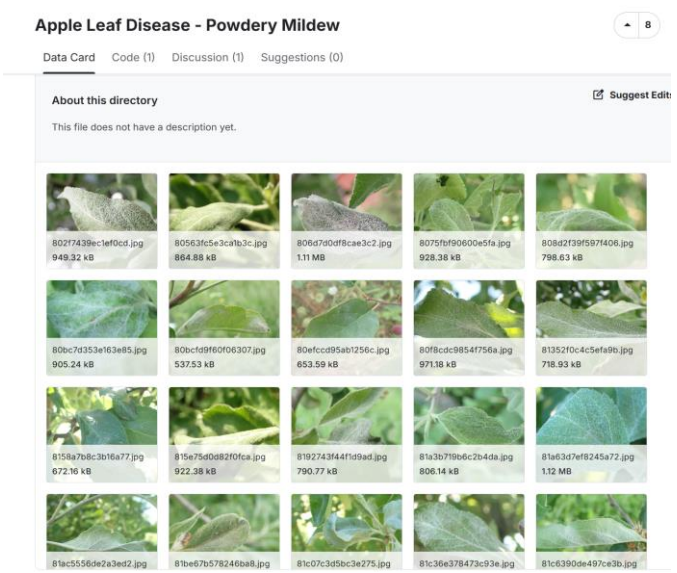
개발 동기

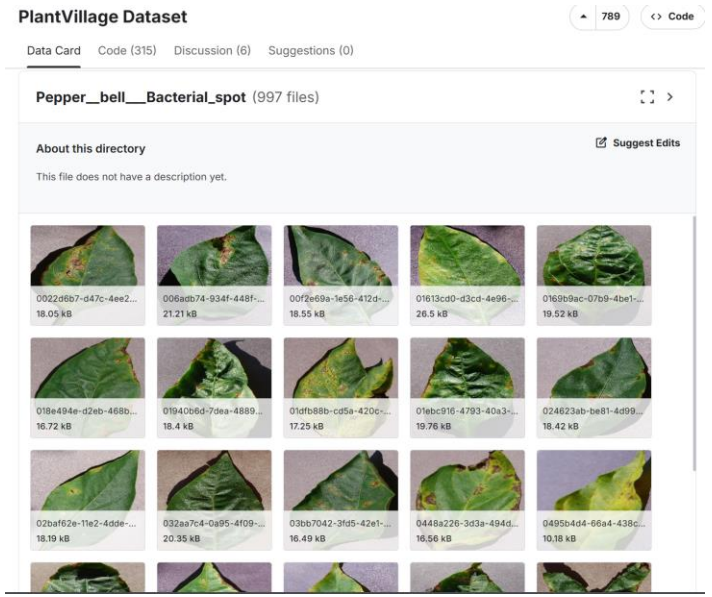
개발 동기

개발 도구

개발 과정(Dataset)



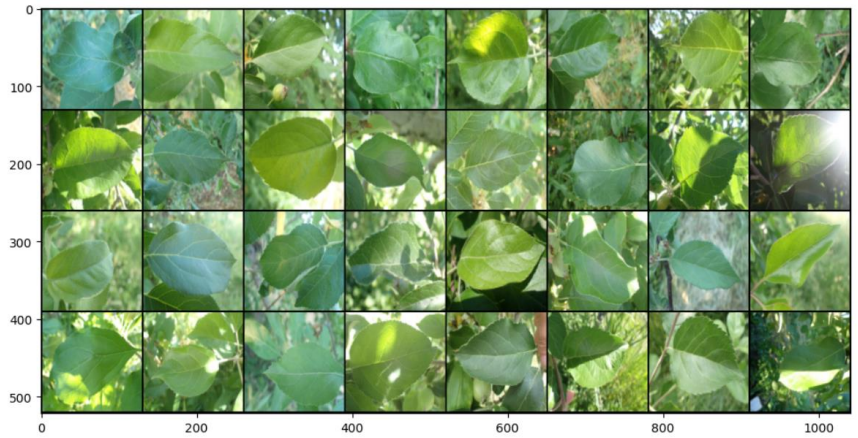




개발 과정(Code)

```python
from torchvision.utils import make_grid
import numpy as np

def show_images(images, labels):
    images = images * 0.5 + 0.5  # unnormalize
    npimg = make_grid(images.cpu()).numpy()
    plt.figure(figsize=(18, 6))
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    print("Labels:", [test_dataset.classes[l] for l in labels])

images, labels = next(iter(test_loader))
show_images(images, labels)
```

```
Labels: ['Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test', 'Test',
```




```python
num_epochs = 5

for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, predicted = torch.max(outputs, 1)
        correct += (predicted == labels).sum().item()
        total += labels.size(0)

    print(f"Epoch {epoch+1}/{num_epochs} | Loss: {running_loss/len(train_loader):.4f} | Accuracy: {100*correct/total:.2f}%")
```

```
Epoch 1/5 | Loss: 0.2382 | Accuracy: 80.00%
Epoch 2/5 | Loss: 0.0000 | Accuracy: 100.00%
Epoch 3/5 | Loss: 0.0000 | Accuracy: 100.00%
Epoch 4/5 | Loss: 0.0000 | Accuracy: 100.00%
Epoch 5/5 | Loss: 0.0000 | Accuracy: 100.00%
```

```python
test_dataset = datasets.ImageFolder("/content/mydata", transform=transform)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

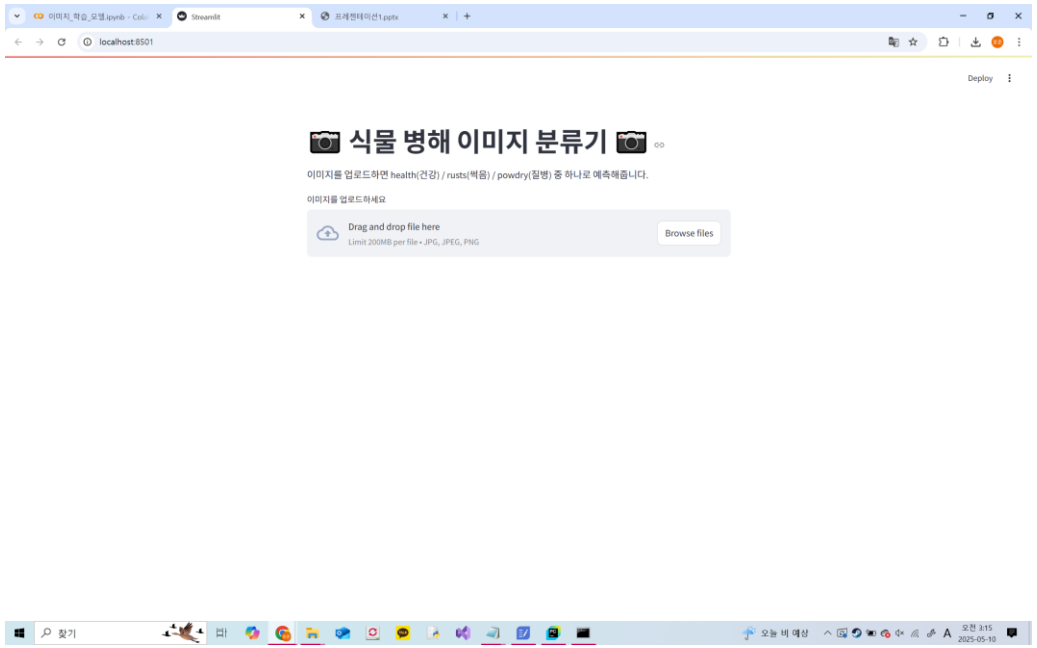
```python
model.eval()
correct = 0
total = 0

with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        correct += (predicted == labels).sum().item()
        total += labels.size(0)

print(f"Test Accuracy: {100 * correct / total:.2f}%")
# 모델 평가 정확도
```

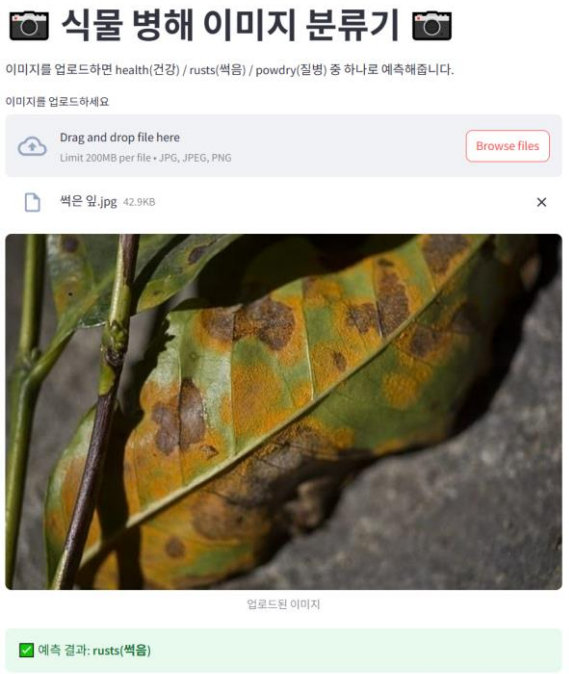
```
Test Accuracy: 100.00%
```

개발 과정(Code)

```
model = SimpleNN()
model.load_state_dict(torch.load("C:/Users/이호준/OneDrive/바탕 화면/app/simple_nn_model.pth", map_location=torch.device('cpu')))
model.eval()

def transform_image(image):  1개의 사용 위치
    transform = transforms.Compose([
        transforms.Resize((128, 128)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
    ])
    return transform(image).unsqueeze(0)

def predict(image):  1개의 사용 위치
    image_tensor = transform_image(image)
    with torch.no_grad():
        outputs = model(image_tensor)
        _, predicted = torch.max(outputs, 1)
    classes = ['health(건강)', 'rusts(썩음)', 'powdry(질병)']
    return classes[predicted.item()]


st.title("📷 식물 병해 이미지 분류기 📷")
st.write("이미지를 업로드하면 health(건강) / rusts(썩음) / powdry(질병) 중 하나로 예측해줍니다.")

uploaded_file = st.file_uploader("이미지를 업로드하세요", type=["jpg", "jpeg", "png"])

if uploaded_file is not None:
    image = Image.open(uploaded_file).convert("RGB")
    st.image(image, caption="업로드된 이미지", use_container_width=True)

    prediction = predict(image)
    st.success(f"✅ 예측 결과: **{prediction}**")
```

개발 과정(Web)


localhost:8501
Deploy
📷 식물 병해 이미지 분류기 📷
이미지를 업로드하면 health(건강) / rusts(썩음) / powdry(질병) 중 하나로 예측해줍니다.
이미지를 업로드하세요
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files


📷 식물 병해 이미지 분류기 📷
이미지를 업로드하면 health(건강) / rusts(썩음) / powdry(질병) 중 하나로 예측해줍니다.
이미지를 업로드하세요
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
썩은 잎.jpg 42.9KB
업로드된 이미지
✅ 예측 결과: rusts(썩음)

시사점

1. 사진을 올림으로써 식물 상태를 누구나 간편히 알 수 있다.
2. 스마트팜과 같은 고비용이 아닌 저렴하게 사용할 수 있다.
3. 식물의 다양한 질병에 대해 구현하지 못하였다.
4. 두가지 이상(썩고 병듦)의 상태에서 구별하지 못하는 상황이 발생

감사합니다.