CNN-based Disease Prediction for Crops

Shreyas Anil, Ryan Bernstein, Aryan Mistry, Justin Yang

The Problem

- Globally, more than 20% of crop yields are lost to plant disease every year.
- Plant disease contributes to food insecurity, poverty, and famine, especially in lesser-developed countries.
- There is a pressing need for a reliable early-detection system, which would allow farmers to save blighted crops.

Objectives

- We aim to implement a computer-vision deep learning model that classifies a plant image by species, and predicts whether it is diseased or not.
- We leverage the PlantDoc dataset (~2600 images from 13 different species and 17 different disease types) for model training.

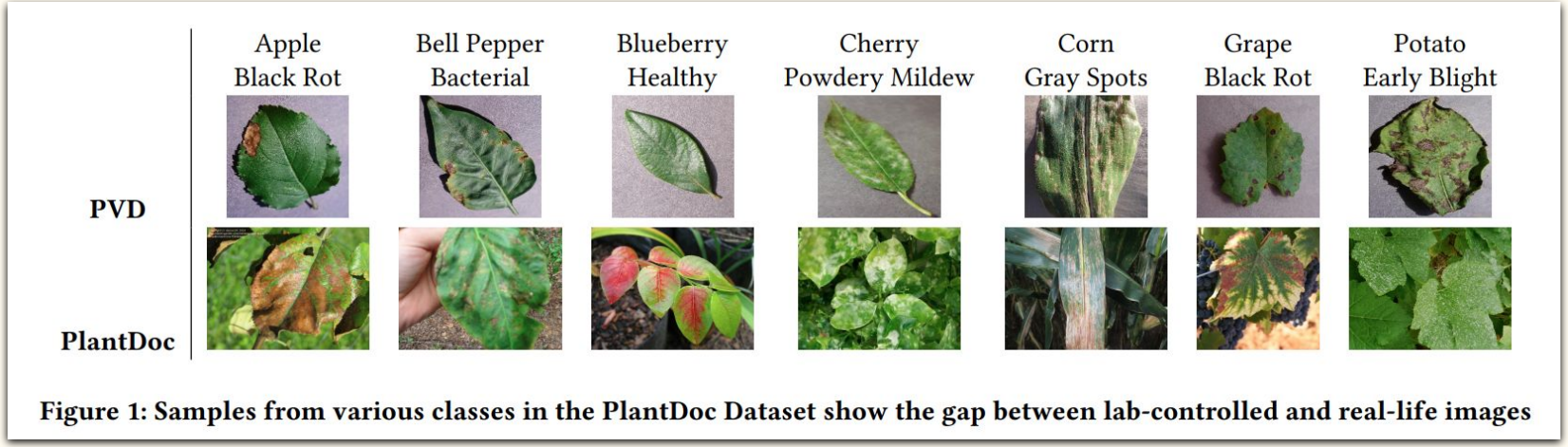
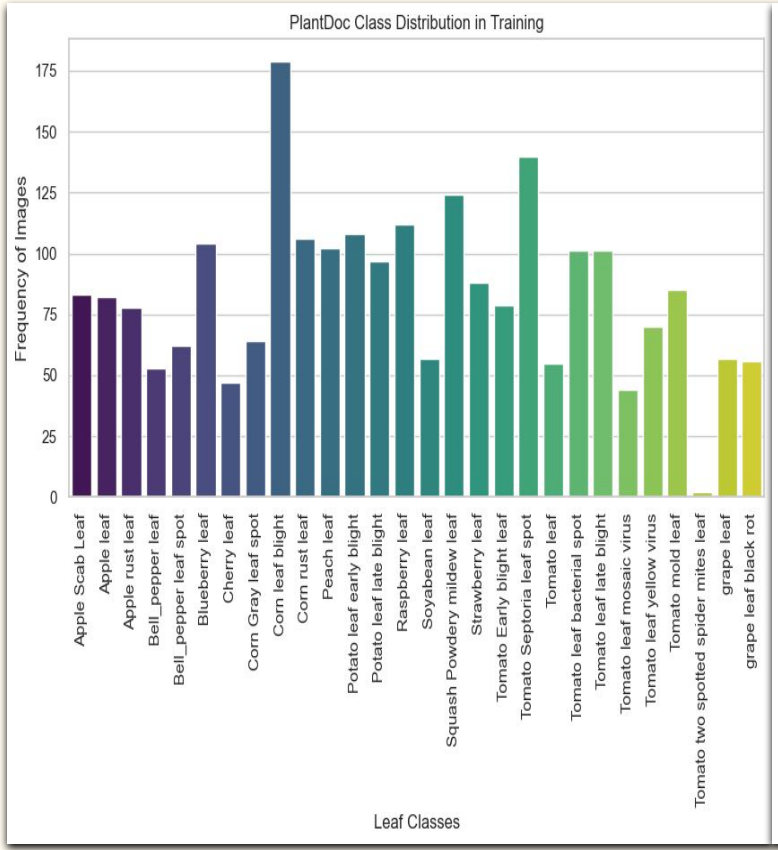
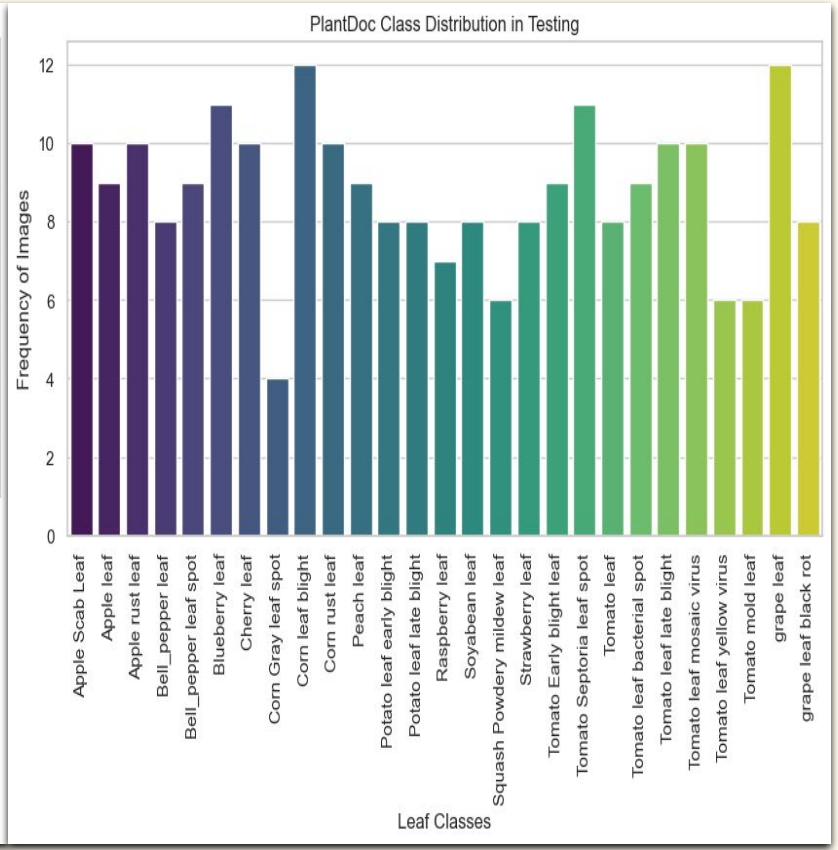


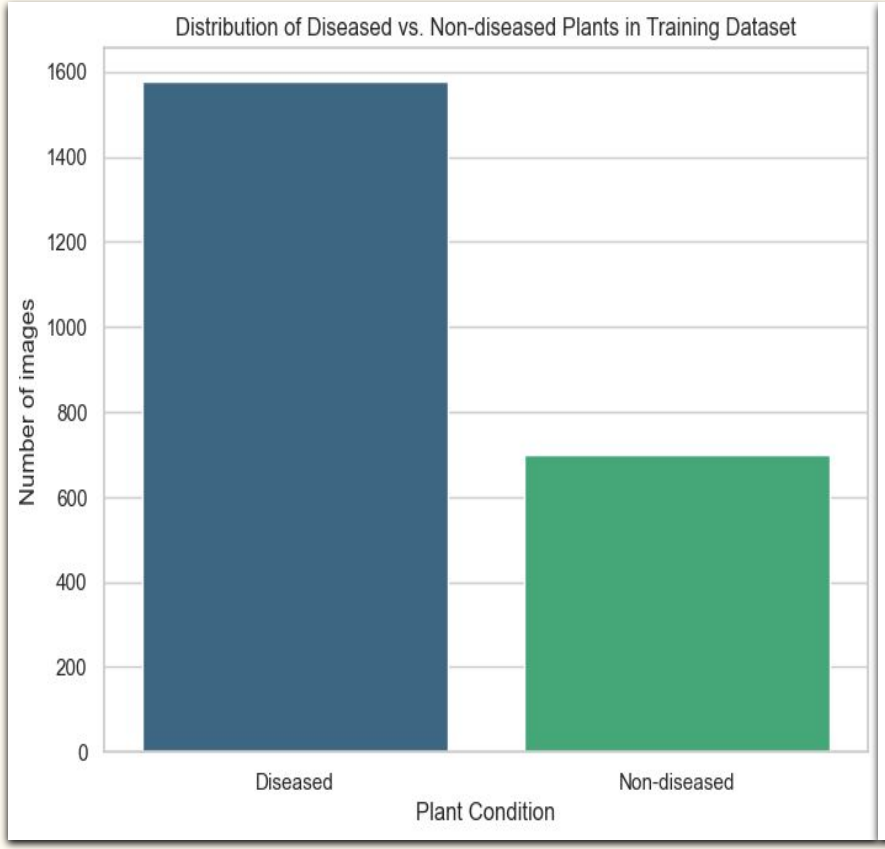
Figure 1: Samples from various classes in the PlantDoc Dataset show the gap between lab-controlled and real-life images

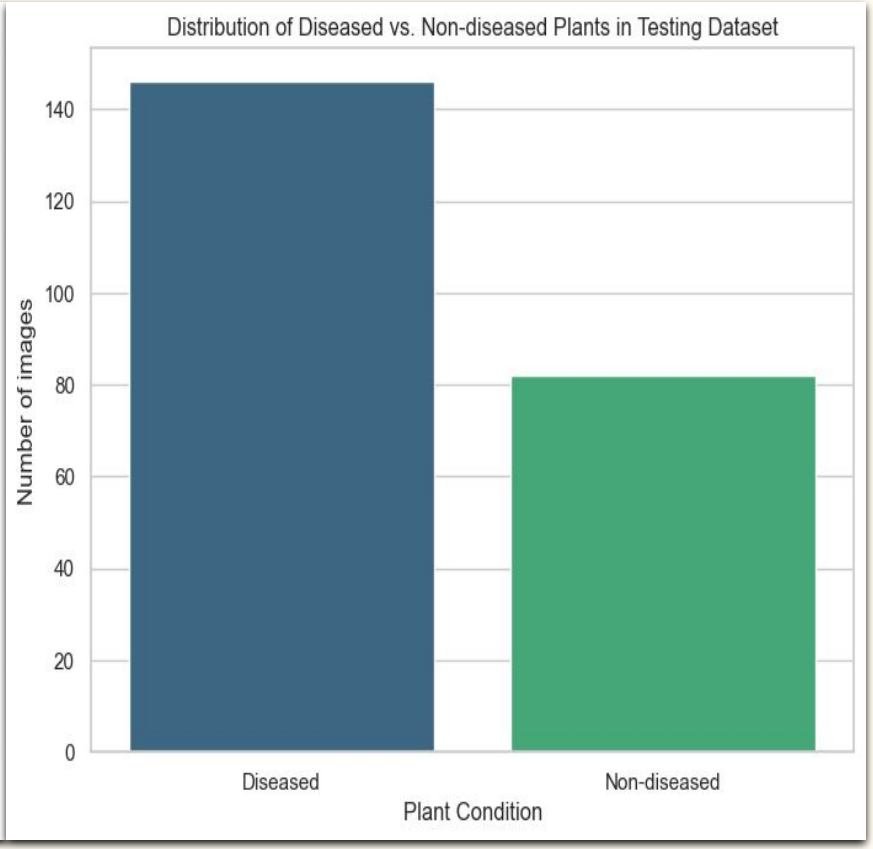
Data Structure - Species





Data Structure - Diseases





Methodology

Data Preprocessing/Augmentation:

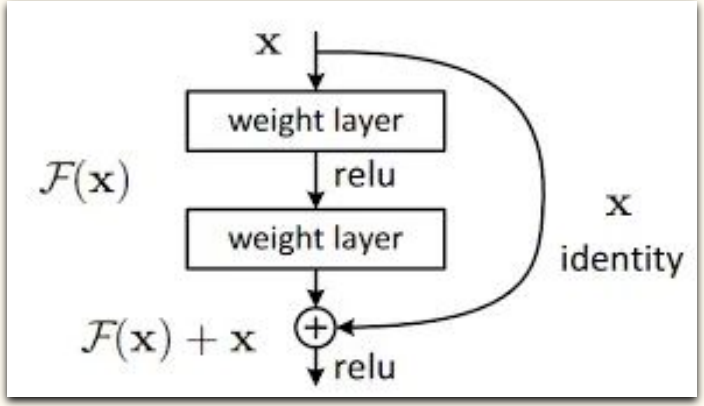
1. Resized and cropped input images
2. Randomly introduced horizontally flipped images into training data
3. Converted images to tensors and normalized color via ImageNet
4. Used a DataLoader with reshuffling for developing batches

Modeling

- A Simple CNN with 9 layers
  - Vanilla
  - w/Regularization
  - w/Stochastic Gradient Descent, Regularization and Scheduler
- ResNet 18 and 50
- EfficientNet 0 and B4
- All trained for only 10 epochs

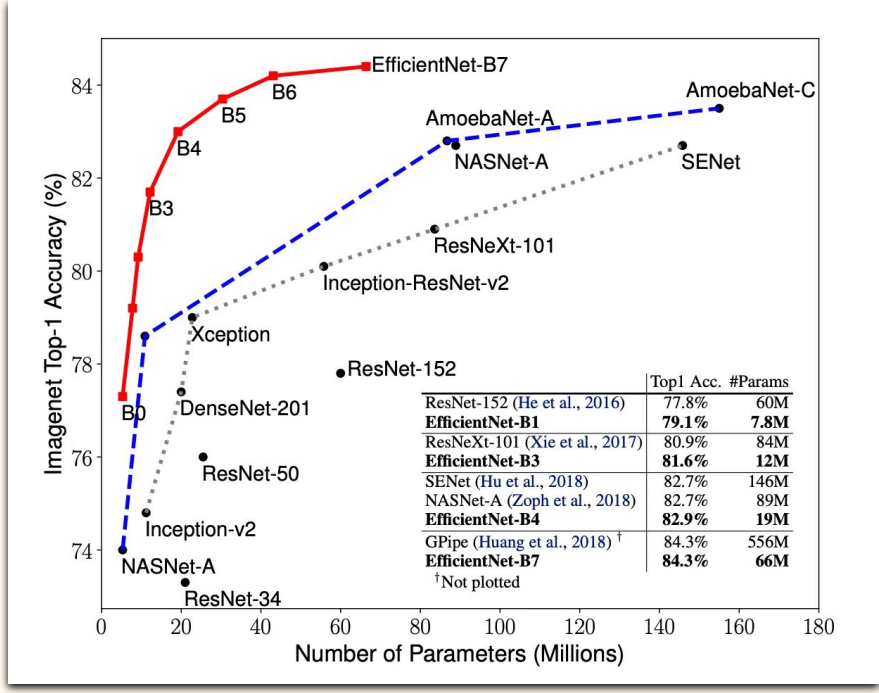
ResNet

- Developed by Microsoft Research
- Builds on existing CNNs with a New Layer: Residual Blocks
  - Contains “Skip Connection”
- We utilized ResNets already pretrained on ImageNet with 18 and 50 layers
- Helps solve the Vanishing/Exploding Gradient Problem, while also easily adaptable



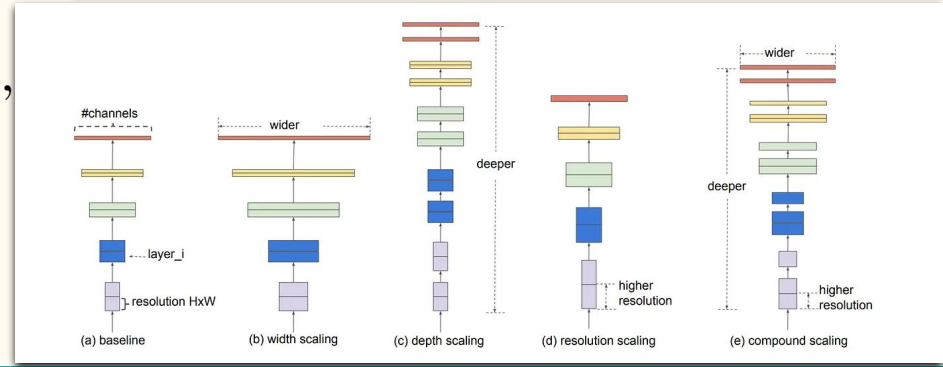
EfficientNet

- Developed by Google AI
- Employs **compound scaling**, which uniformly scales all dimensions of depth, width, and resolution
- We utilized EfficientNet B0 and B4, as depicted in the graph above
  - Pretrained on ImageNet
- Helps scale parameters simultaneously, allowing for better utilization of computational resources



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted



Code Overview

```
[ ] res_50_model = create_pretrained_model(num_classes)

    # Set up loss and optimizer
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(res_50_model.parameters(), lr=0.001)

    # Training and testing
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth
100%|██████████| 97.8M/97.8M [00:00<00:00, 113MB/s]
```

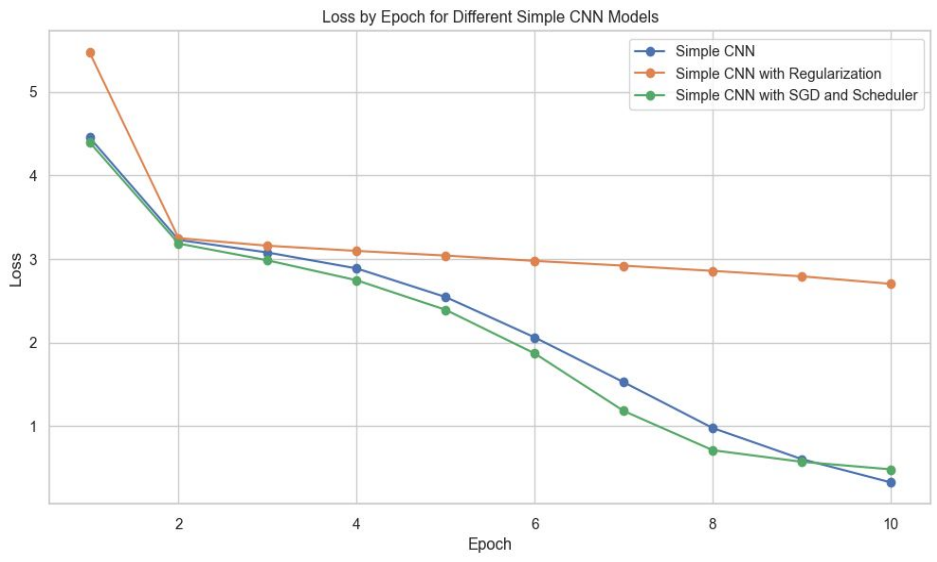
```
[ ] train_model(res_50_model, train_loader, criterion, optimizer, num_epochs=10, device=device)
```

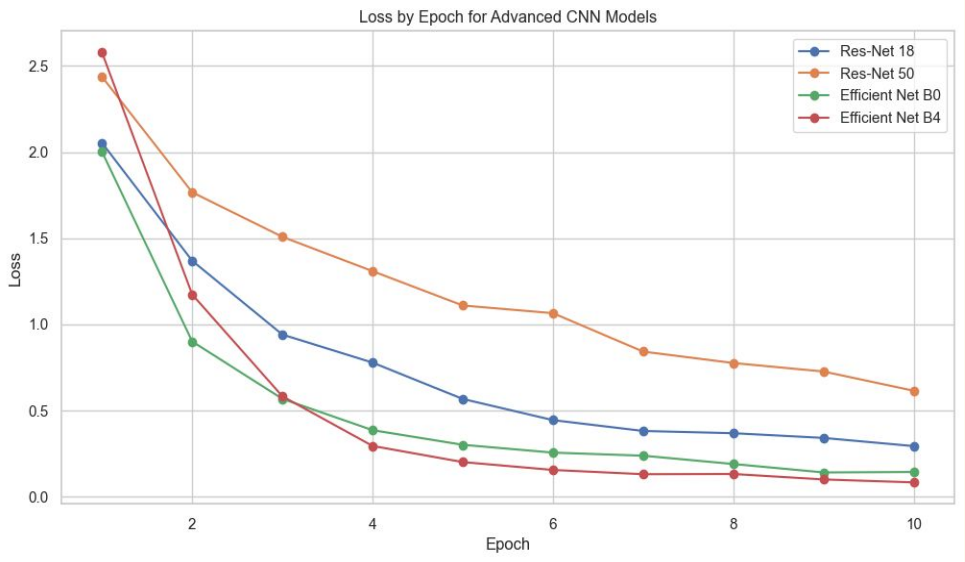
```
Epoch 1, Loss: 2.43699514543688
Epoch 2, Loss: 1.7665844930184853
Epoch 3, Loss: 1.5082206436105676
Epoch 4, Loss: 1.3089014195107125
Epoch 5, Loss: 1.1097415427903872
Epoch 6, Loss: 1.0651415248174925
Epoch 7, Loss: 0.8421571931323489
Epoch 8, Loss: 0.7758747323139293
Epoch 9, Loss: 0.7265614617515255
Epoch 10, Loss: 0.6148334806029861
```

```
[ ] test_model(res_50_model, test_loader, device=device)
```

```
Accuracy of the network on the test images: 39.40677966101695%
F1 Score (Weighted): 0.37661299605204734
```

Results - Model Losses over Epochs





Results-Accuracy Table

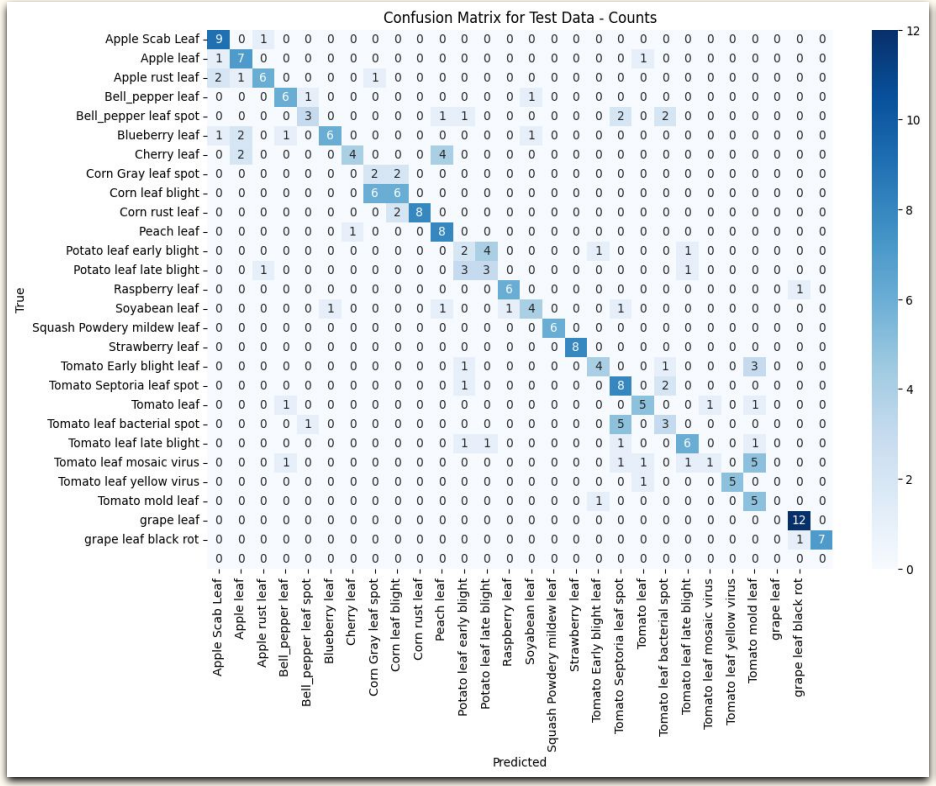
Accuracy and F1 Score for Simple CNNs

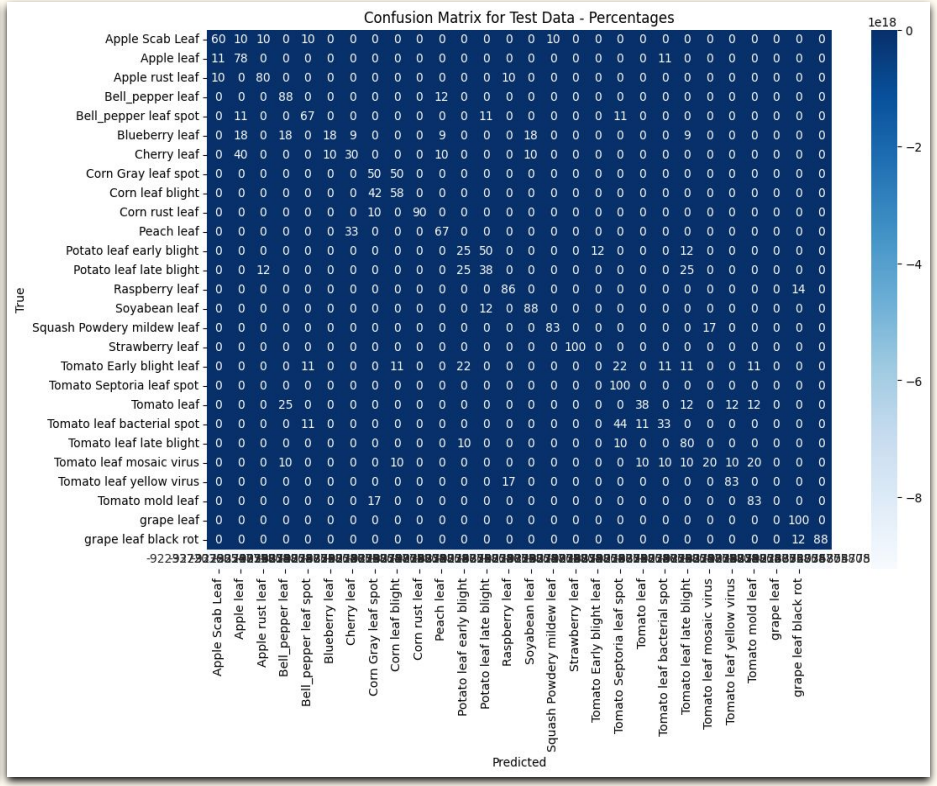
Model	Accuracy (%)	F1 Score (Weighted)
Simple CNN	18.220338983050848	0.15146084728608208
Simple CNN with Regularization	11.016949152542374	0.08775200407716652
Simple CNN with SGD and Scheduler	16.10169491525424	0.1480259592552006

Accuracy and F1 Score for Advanced CNNs

Model	Accuracy (%)	F1 Score (Weighted)
Res-Net 18	40.25423728813559	0.3666380717501854
Res-Net 50	39.40677966101695	0.37661299605204734
Efficient Net B0	47.45762711864407	0.47404671235547036
Efficient Net B4	52.54237288135593	0.5177276451181709

Results - Confusion Matrix for EfficientNet B4





Conclusion and Next Steps

- EfficientNet B4 was our best model; it achieved impressive accuracy despite only being trained for 10 epochs.
- Complex CNNs with pre-trained weights perform significantly better than simple CNNs without pre-trained weights
- In the future we may want to look into doing bounding box implementation
- We could also do a binary classification task to see how well a model can distinguish between diseased plants and non-diseased plants

Sources and Extra Links

[1] Savary, S., Willocquet, L., Pethybridge, S.J. *et al.* The global burden of pathogens and pests on major food crops. *Nat Ecol Evol* 3, 430–439 (2019). https://doi.org/10.1038/s41559-018-0793-y

[2] Singh, Davinder, et al. "PlantDoc: A dataset for visual plant disease detection." *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*. 2020. 249-253.

[3] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[4] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International conference on machine learning*. PMLR, 2019.

Link to GitHub: https://github.com/jyang0620/Plant_Disease_Classifier

- Medium article coming soon, will be linked in the GitHub