segmentedImages

May 19, 2024

1 Segmented images

The aim of this project is to build a deep neural network capable of analyzing images captured by smartphones to quickly and accurately identify crop diseases.

We are analyzing 54,306 images of plant leaves, which are categorized into 38 class labels. Each class label corresponds to a crop-disease pair, and our goal is to predict this pair from the image of the plant leaf. These images are sourced from the dataset available at the following repository: https://github.com/digitalepidemiologylab/plantvillage_deeplearning_paper_dataset.

To train our AI-based image recognition system, we will utilize this dataset. In all our experiments, we utilize three different versions of the PlantVillage dataset. We begin with the original dataset in color, then we explore a grayscale version, and finally, we conduct our experiments on a version where the leaves are segmented. This approach allows us to assess the performance and robustness of our image recognition system in various contexts. We analyze how variations such as color, grayscale, and leaf segmentation can impact the model's results. By understanding how our system behaves under these different conditions, we can better evaluate its ability to generalize and operate effectively in real-world environments. These three versions of the data are already available via the above-mentioned link.

The different of crop disease types used in this project :

0: Grape___healthy

1: Peach___Bacterial_spot

2: Apple___healthy

3: Orange___Haunglongbing_(Citrus_greening)

4: Corn_(maize)___healthy

5: Tomato___Septoria_leaf_spot

6: Tomato___healthy

7: Corn_(maize)__Common_rust$

8: Tomato___Early_blight

9: Potato___Late_blight

10: Peach___healthy

11: Corn_(maize)___Northern_Leaf_Blight

12: Blueberry___healthy

13: Grape___Leaf_blight_(Isariopsis_Leaf_Spot)

14: Tomato___Leaf_Mold

15: Soybean___healthy

16: Cherry_(including_sour)___healthy

17: Tomato___Spider_mites Two-spotted_spider_mite

18: Potato___healthy

19: Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot

20: Cherry_(including_sour)___Powdery_mildew

21: Apple___Cedar_apple_rust

22: Squash___Powdery_mildew

23: Tomato___Late_blight

24: Grape___Black_rot

25: Pepper,_bell___healthy

26: Tomato___Target_Spot

27: Apple___Black_rot

28: Tomato___Bacterial_spot

29: Strawberry___healthy

30: Pepper,_bell___Bacterial_spot

31: Raspberry___healthy

32: Tomato___Tomato_Yellow_Leaf_Curl_Virus

33: Apple___Apple_scab

34: Potato___Early_blight

35: Tomato___Tomato_mosaic_virus

36: Strawberry___Leaf_scorch

37: Grape___Esca_(Black_Measles)

[ ]:
```
import sys
sys.path.append('/content/project/src')

from utils import *
from crop_disease_dataset import *
from model import *
from train import *
from evaluation import *
```

[ ]:
```
repo_url = "https://github.com/digitalepidemiologylab/
 ↪plantvillage_deeplearning_paper_dataset.git"
clone_dir = "plantvillage_deeplearning_paper_dataset"
extracted_folder = "raw/segmented"

clone_repo(repo_url, clone_dir)
classes = extract_folder(repo_url, clone_dir, extracted_folder)
classes
```

```
Folder 'raw/segmented' extracted successfully.
Directory 'Peach___Bacterial_spot' contains 2297 files.
Directory 'Cherry_(including_sour)___Powdery_mildew' contains 1052 files.
Directory 'Tomato___Septoria_leaf_spot' contains 1771 files.
Directory 'Raspberry___healthy' contains 371 files.
Directory 'Grape___healthy' contains 423 files.
Directory 'Peach___healthy' contains 360 files.
Directory 'Tomato___Bacterial_spot' contains 2127 files.
Directory 'Tomato___Target_Spot' contains 1404 files.
Directory 'Potato___Early_blight' contains 1000 files.
Directory 'Strawberry___healthy' contains 456 files.
Directory 'Squash___Powdery_mildew' contains 1835 files.
Directory 'Tomato___Tomato_Yellow_Leaf_Curl_Virus' contains 5357 files.
Directory 'Orange___Haunglongbing_(Citrus_greening)' contains 5507 files.
Directory 'Pepper,_bell___Bacterial_spot' contains 997 files.
Directory 'Soybean___healthy' contains 5090 files.
Directory 'Tomato___healthy' contains 1591 files.
Directory 'Blueberry___healthy' contains 1502 files.
Directory 'Apple___Black_rot' contains 621 files.
Directory 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot' contains 513
files.
Directory 'Tomato___Early_blight' contains 1000 files.
Directory 'Tomato___Leaf_Mold' contains 952 files.
Directory 'Apple___Apple_scab' contains 630 files.
Directory 'Grape___Black_rot' contains 1180 files.
Directory 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)' contains 1076 files.
Directory 'Apple___Cedar_apple_rust' contains 275 files.
Directory 'Corn_(maize)___Common_rust_' contains 1192 files.
Directory 'Apple___healthy' contains 1645 files.
Directory 'Tomato___Tomato_mosaic_virus' contains 373 files.
Directory 'Potato___healthy' contains 152 files.
Directory 'Strawberry___Leaf_scorch' contains 1109 files.
Directory 'Pepper,_bell___healthy' contains 1478 files.
Directory 'Corn_(maize)___healthy' contains 1162 files.
Directory 'Tomato___Late_blight' contains 1909 files.
Directory 'Cherry_(including_sour)___healthy' contains 854 files.
Directory 'Potato___Late_blight' contains 1000 files.
Directory 'Tomato___Spider_mites Two-spotted_spider_mite' contains 1676 files.
Directory 'Grape___Esca_(Black_Measles)' contains 1384 files.
```

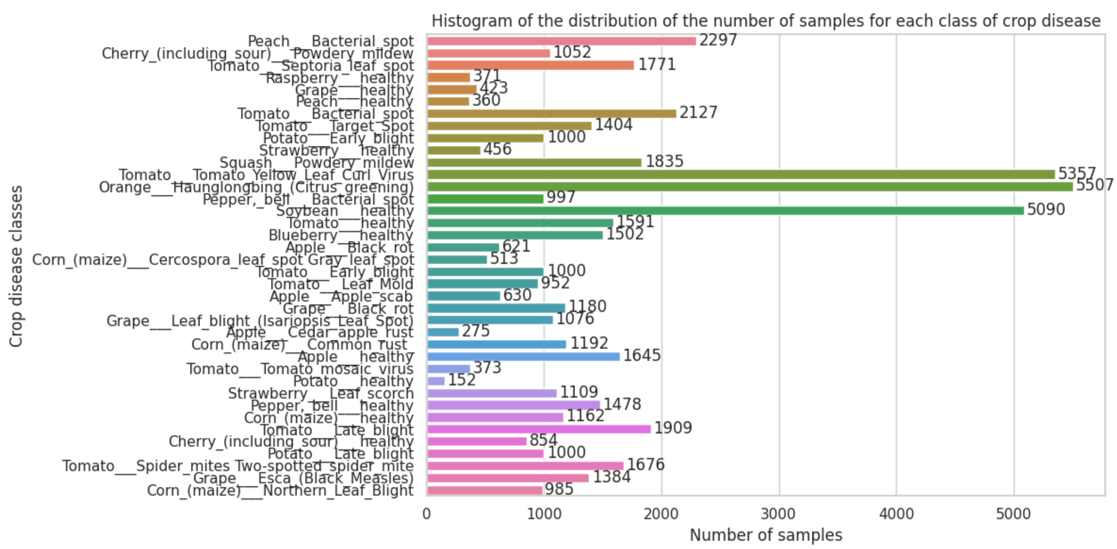
```
Directory 'Corn_(maize)___Northern_Leaf_Blight' contains 985 files.
Total number of files: 54306

Total number of classes : '38'.
{'Peach___Bacterial_spot': 2297, 'Cherry_(including_sour)___Powdery_mildew':
1052, 'Tomato___Septoria_leaf_spot': 1771, 'Raspberry___healthy': 371,
'Grape___healthy': 423, 'Peach___healthy': 360, 'Tomato___Bacterial_spot': 2127,
'Tomato___Target_Spot': 1404, 'Potato___Early_blight': 1000,
'Strawberry___healthy': 456, 'Squash___Powdery_mildew': 1835,
'Tomato___Tomato_Yellow_Leaf_Curl_Virus': 5357,
'Orange___Haunglongbing_(Citrus_greening)': 5507,
'Pepper,_bell___Bacterial_spot': 997, 'Soybean___healthy': 5090,
'Tomato___healthy': 1591, 'Blueberry___healthy': 1502, 'Apple___Black_rot': 621,
'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot': 513,
'Tomato___Early_blight': 1000, 'Tomato___Leaf_Mold': 952, 'Apple___Apple_scab':
630, 'Grape___Black_rot': 1180, 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)':
1076, 'Apple___Cedar_apple_rust': 275, 'Corn_(maize)___Common_rust_': 1192,
'Apple___healthy': 1645, 'Tomato___Tomato_mosaic_virus': 373,
'Potato___healthy': 152, 'Strawberry___Leaf_scorch': 1109,
'Pepper,_bell___healthy': 1478, 'Corn_(maize)___healthy': 1162,
'Tomato___Late_blight': 1909, 'Cherry_(including_sour)___healthy': 854,
'Potato___Late_blight': 1000, 'Tomato___Spider_mites Two-spotted_spider_mite':
1676, 'Grape___Esca_(Black_Measles)': 1384,
'Corn_(maize)___Northern_Leaf_Blight': 985}
```

```
[ ]: {'Peach___Bacterial_spot': 2297,
      'Cherry_(including_sour)___Powdery_mildew': 1052,
      'Tomato___Septoria_leaf_spot': 1771,
      'Raspberry___healthy': 371,
      'Grape___healthy': 423,
      'Peach___healthy': 360,
      'Tomato___Bacterial_spot': 2127,
      'Tomato___Target_Spot': 1404,
      'Potato___Early_blight': 1000,
      'Strawberry___healthy': 456,
      'Squash___Powdery_mildew': 1835,
      'Tomato___Tomato_Yellow_Leaf_Curl_Virus': 5357,
      'Orange___Haunglongbing_(Citrus_greening)': 5507,
      'Pepper,_bell___Bacterial_spot': 997,
      'Soybean___healthy': 5090,
      'Tomato___healthy': 1591,
      'Blueberry___healthy': 1502,
      'Apple___Black_rot': 621,
      'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot': 513,
      'Tomato___Early_blight': 1000,
      'Tomato___Leaf_Mold': 952,
      'Apple___Apple_scab': 630,
```

```
'Grape___Black_rot': 1180,
'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)': 1076,
'Apple___Cedar_apple_rust': 275,
'Corn_(maize)___Common_rust_': 1192,
'Apple___healthy': 1645,
'Tomato___Tomato_mosaic_virus': 373,
'Potato___healthy': 152,
'Strawberry___Leaf_scorch': 1109,
'Pepper,_bell___healthy': 1478,
'Corn_(maize)___healthy': 1162,
'Tomato___Late_blight': 1909,
'Cherry_(including_sour)___healthy': 854,
'Potato___Late_blight': 1000,
'Tomato___Spider_mites Two-spotted_spider_mite': 1676,
'Grape___Esca_(Black_Measles)': 1384,
'Corn_(maize)___Northern_Leaf_Blight': 985}
```

[ ]:
```
plot_class_histogram(classes)
```



[ ]:
```
# Spliiting the dataset
train_set = CropDiseaseDataset(root_dir=extracted_folder, train=True,␣
↪validation=False, gray_scale=False, segmented=True)
validation_set = CropDiseaseDataset(root_dir=extracted_folder, train=False,␣
↪validation=True, gray_scale=False, segmented=True)
test_set = CropDiseaseDataset(root_dir=extracted_folder, train=False,␣
↪validation=False, gray_scale=False, segmented=True)
```

```
trainloader = torch.utils.data.DataLoader(train_set, batch_size=64,␣
↪shuffle=True, num_workers=2)
validationloader = torch.utils.data.DataLoader(validation_set, batch_size=64,␣
↪shuffle=False, num_workers=2)
testloader = torch.utils.data.DataLoader(test_set, batch_size=64, shuffle=False,␣
↪num_workers=2)

print(f"Train set: '{len(train_set)}' images,", f"Validation set:␣
↪'{len(validation_set)}' images,", f"Test set: '{len(test_set)}' images")
```

```
Train set: '35297' images, Validation set: '8145' images, Test set: '10863'
images
```

[ ]:
```
# Define hyperparameters
learning_rate = 1e-3
num_epochs = 20

network = CNN(gray_scale=False).to(device)
network.apply(initialize_parameters)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(network.parameters(), lr=learning_rate)

print(f'The model has {count_parameters(network):,} trainable parameters')
```

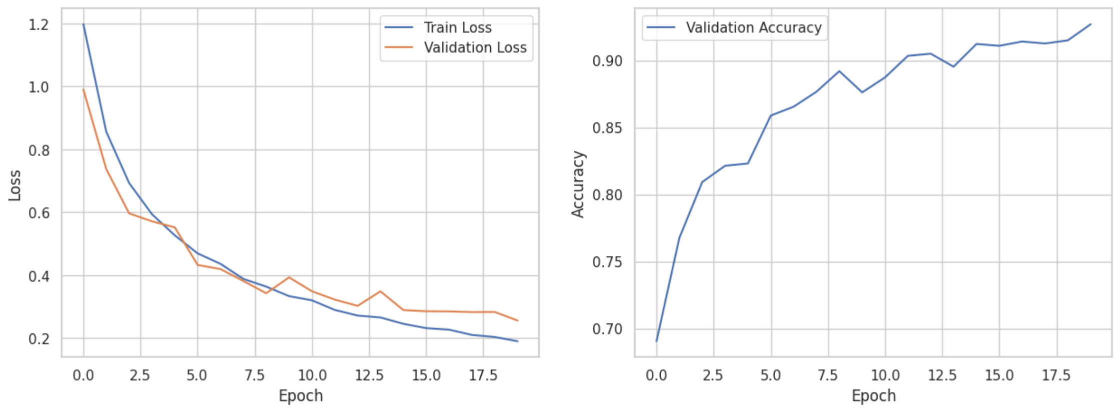
```
The model has 6,846,758 trainable parameters
```

[ ]:
```
# Train the model
train_avg_loss, validation_avg_loss, validation_accuracy, train_accuracy =␣
↪train(network, num_epochs, trainloader, validationloader, criterion,␣
↪optimizer,validation_phase=True)
```

```
Epoch [1/20], Test Loss: 0.9900, Test Accuracy: 69.05%, Train Loss: 1.1972,
Train Accuracy: 63.64%
| Epoch Time: 0m 32s
Epoch [2/20], Test Loss: 0.7375, Test Accuracy: 76.76%, Train Loss: 0.8561,
Train Accuracy: 73.02%
| Epoch Time: 0m 32s
Epoch [3/20], Test Loss: 0.5967, Test Accuracy: 80.92%, Train Loss: 0.6932,
Train Accuracy: 77.87%
| Epoch Time: 0m 32s
Epoch [4/20], Test Loss: 0.5714, Test Accuracy: 82.14%, Train Loss: 0.5940,
Train Accuracy: 80.87%
| Epoch Time: 0m 32s
Epoch [5/20], Test Loss: 0.5521, Test Accuracy: 82.31%, Train Loss: 0.5268,
Train Accuracy: 83.15%
| Epoch Time: 0m 34s
Epoch [6/20], Test Loss: 0.4328, Test Accuracy: 85.88%, Train Loss: 0.4694,
Train Accuracy: 84.75%
| Epoch Time: 0m 32s
```

```
Epoch [7/20], Test Loss: 0.4197, Test Accuracy: 86.54%, Train Loss: 0.4363,
Train Accuracy: 85.85%
| Epoch Time: 0m 32s
Epoch [8/20], Test Loss: 0.3818, Test Accuracy: 87.66%, Train Loss: 0.3885,
Train Accuracy: 87.04%
| Epoch Time: 0m 32s
Epoch [9/20], Test Loss: 0.3428, Test Accuracy: 89.20%, Train Loss: 0.3639,
Train Accuracy: 88.10%
| Epoch Time: 0m 32s
Epoch [10/20], Test Loss: 0.3933, Test Accuracy: 87.61%, Train Loss: 0.3338,
Train Accuracy: 88.88%
| Epoch Time: 0m 32s
Epoch [11/20], Test Loss: 0.3492, Test Accuracy: 88.73%, Train Loss: 0.3205,
Train Accuracy: 89.51%
| Epoch Time: 0m 32s
Epoch [12/20], Test Loss: 0.3227, Test Accuracy: 90.34%, Train Loss: 0.2899,
Train Accuracy: 90.44%
| Epoch Time: 0m 31s
Epoch [13/20], Test Loss: 0.3028, Test Accuracy: 90.50%, Train Loss: 0.2720,
Train Accuracy: 90.92%
| Epoch Time: 0m 32s
Epoch [14/20], Test Loss: 0.3490, Test Accuracy: 89.53%, Train Loss: 0.2660,
Train Accuracy: 91.23%
| Epoch Time: 0m 32s
Epoch [15/20], Test Loss: 0.2894, Test Accuracy: 91.22%, Train Loss: 0.2458,
Train Accuracy: 91.99%
| Epoch Time: 0m 34s
Epoch [16/20], Test Loss: 0.2858, Test Accuracy: 91.09%, Train Loss: 0.2321,
Train Accuracy: 92.37%
| Epoch Time: 0m 36s
Epoch [17/20], Test Loss: 0.2852, Test Accuracy: 91.41%, Train Loss: 0.2273,
Train Accuracy: 92.66%
| Epoch Time: 0m 34s
Epoch [18/20], Test Loss: 0.2830, Test Accuracy: 91.26%, Train Loss: 0.2106,
Train Accuracy: 93.26%
| Epoch Time: 0m 32s
Epoch [19/20], Test Loss: 0.2835, Test Accuracy: 91.49%, Train Loss: 0.2040,
Train Accuracy: 93.36%
| Epoch Time: 0m 33s
Epoch [20/20], Test Loss: 0.2562, Test Accuracy: 92.69%, Train Loss: 0.1905,
Train Accuracy: 93.79%
| Epoch Time: 0m 32s
```

[ ]:
```
plot_training_results(train_avg_loss, validation_avg_loss, validation_accuracy,␣
↪is_validation=True)
```




```
[ ]: # Combining train and validation datasets for testing the model
combined_train_set = ConcatDataset([train_set, validation_set])
combined_train_loader = torch.utils.data.DataLoader(combined_train_set,␣
 ↪batch_size=64, shuffle=True, num_workers=2)
```

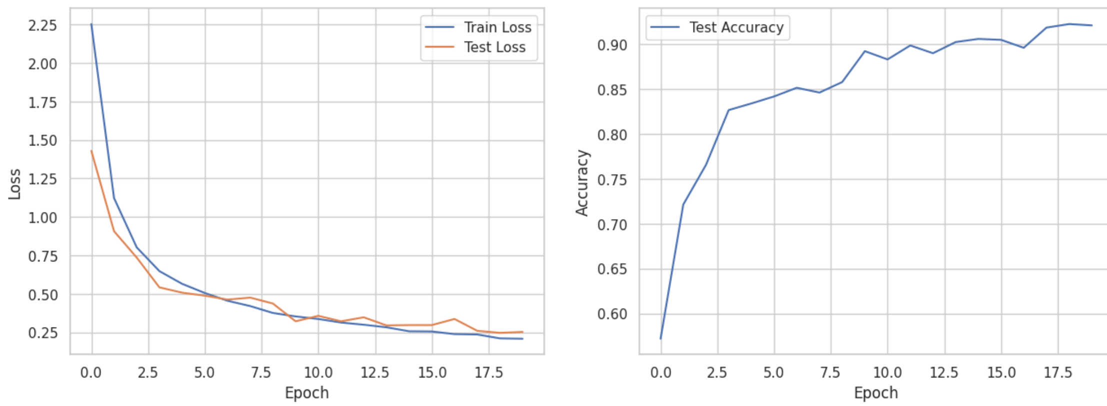
```
[ ]: # Test the model
network = CNN(gray_scale=False).to(device)
network.apply(initialize_parameters)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(network.parameters(), lr=learning_rate)

train_avg_loss, test_avg_loss, test_accuracy, train_accuracy = train(network,␣
 ↪num_epochs, combined_train_loader, testloader, criterion,␣
 ↪optimizer,validation_phase=False)
```

```
Epoch [1/20], Test Loss: 1.4289, Test Accuracy: 57.18%, Train Loss: 2.2526,
Train Accuracy: 42.14%
| Epoch Time: 0m 44s
Epoch [2/20], Test Loss: 0.9070, Test Accuracy: 72.13%, Train Loss: 1.1223,
Train Accuracy: 66.01%
| Epoch Time: 0m 46s
Epoch [3/20], Test Loss: 0.7356, Test Accuracy: 76.55%, Train Loss: 0.8018,
Train Accuracy: 74.96%
| Epoch Time: 0m 43s
Epoch [4/20], Test Loss: 0.5420, Test Accuracy: 82.66%, Train Loss: 0.6476,
Train Accuracy: 79.25%
| Epoch Time: 0m 43s
Epoch [5/20], Test Loss: 0.5077, Test Accuracy: 83.39%, Train Loss: 0.5650,
Train Accuracy: 82.03%
| Epoch Time: 0m 43s
Epoch [6/20], Test Loss: 0.4882, Test Accuracy: 84.18%, Train Loss: 0.5055,
Train Accuracy: 83.73%
| Epoch Time: 0m 43s
```

```
Epoch [7/20], Test Loss: 0.4627, Test Accuracy: 85.14%, Train Loss: 0.4551,
Train Accuracy: 85.17%
| Epoch Time: 0m 46s
Epoch [8/20], Test Loss: 0.4756, Test Accuracy: 84.61%, Train Loss: 0.4206,
Train Accuracy: 86.40%
| Epoch Time: 0m 41s
Epoch [9/20], Test Loss: 0.4370, Test Accuracy: 85.78%, Train Loss: 0.3758,
Train Accuracy: 87.75%
| Epoch Time: 0m 43s
Epoch [10/20], Test Loss: 0.3217, Test Accuracy: 89.22%, Train Loss: 0.3534,
Train Accuracy: 88.28%
| Epoch Time: 0m 42s
Epoch [11/20], Test Loss: 0.3572, Test Accuracy: 88.30%, Train Loss: 0.3366,
Train Accuracy: 89.07%
| Epoch Time: 0m 45s
Epoch [12/20], Test Loss: 0.3217, Test Accuracy: 89.86%, Train Loss: 0.3136,
Train Accuracy: 89.71%
| Epoch Time: 0m 43s
Epoch [13/20], Test Loss: 0.3475, Test Accuracy: 88.99%, Train Loss: 0.2995,
Train Accuracy: 90.25%
| Epoch Time: 0m 42s
Epoch [14/20], Test Loss: 0.2951, Test Accuracy: 90.23%, Train Loss: 0.2827,
Train Accuracy: 90.85%
| Epoch Time: 0m 43s
Epoch [15/20], Test Loss: 0.2971, Test Accuracy: 90.58%, Train Loss: 0.2569,
Train Accuracy: 91.55%
| Epoch Time: 0m 45s
Epoch [16/20], Test Loss: 0.2970, Test Accuracy: 90.48%, Train Loss: 0.2556,
Train Accuracy: 91.60%
| Epoch Time: 0m 43s
Epoch [17/20], Test Loss: 0.3367, Test Accuracy: 89.60%, Train Loss: 0.2385,
Train Accuracy: 92.20%
| Epoch Time: 0m 43s
Epoch [18/20], Test Loss: 0.2601, Test Accuracy: 91.84%, Train Loss: 0.2360,
Train Accuracy: 92.37%
| Epoch Time: 0m 45s
Epoch [19/20], Test Loss: 0.2466, Test Accuracy: 92.24%, Train Loss: 0.2108,
Train Accuracy: 93.12%
| Epoch Time: 0m 43s
Epoch [20/20], Test Loss: 0.2526, Test Accuracy: 92.09%, Train Loss: 0.2084,
Train Accuracy: 93.25%
| Epoch Time: 0m 42s
```

[ ]:
```
plot_training_results(train_avg_loss, test_avg_loss, test_accuracy,
 ↪is_validation=False)
```



[ ]:

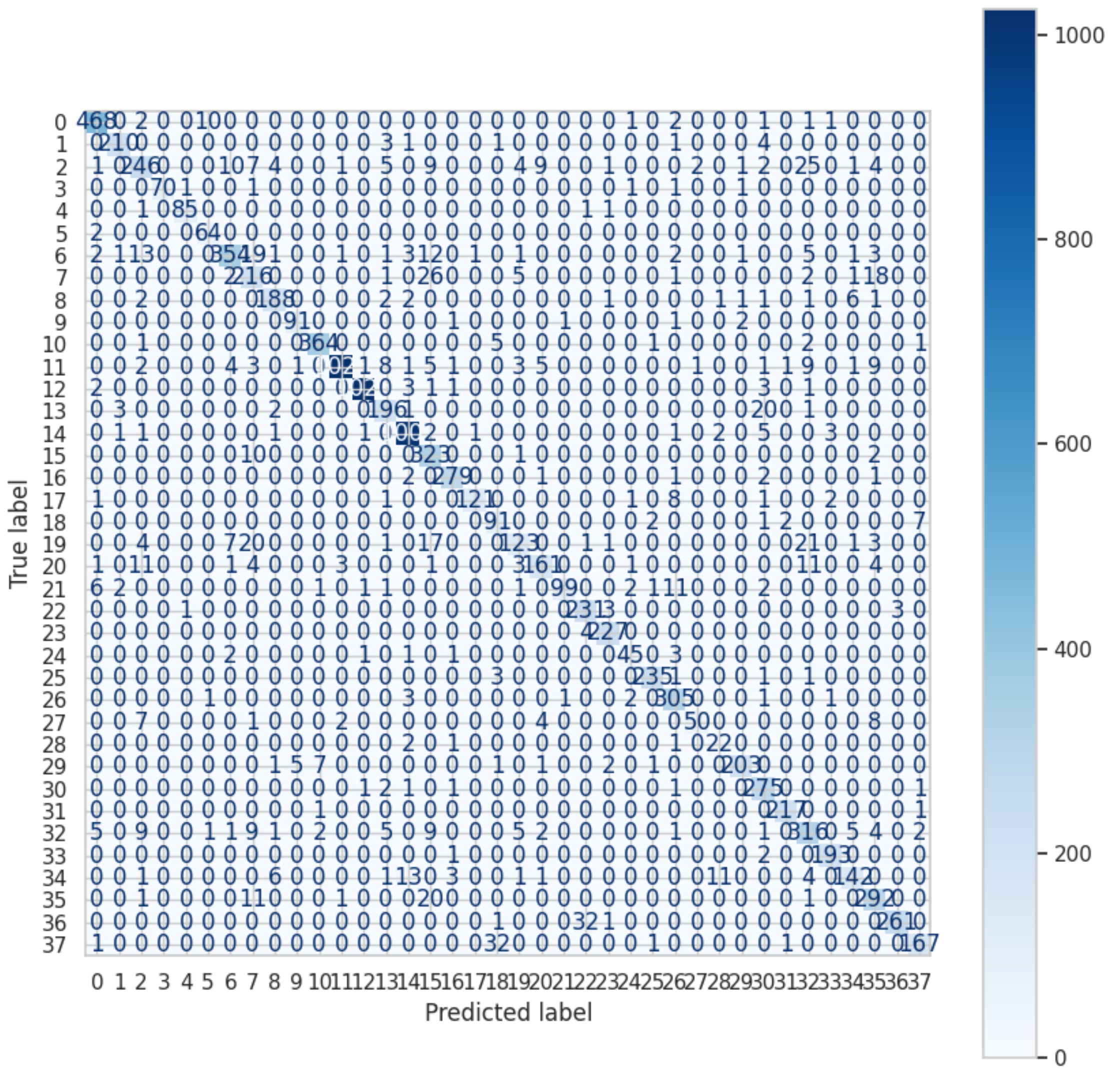
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
network.load_state_dict(torch.load('model.pt'))
images, labels, probs, corrects = get_predictions(network, testloader, device)
pred_labels = torch.argmax(probs, 1)

print(f"There are {len(corrects)} correct predictions.")
```

```
There are 10863 correct predictions.
```

[ ]:

```
plot_confusion_matrix(labels, pred_labels)
```



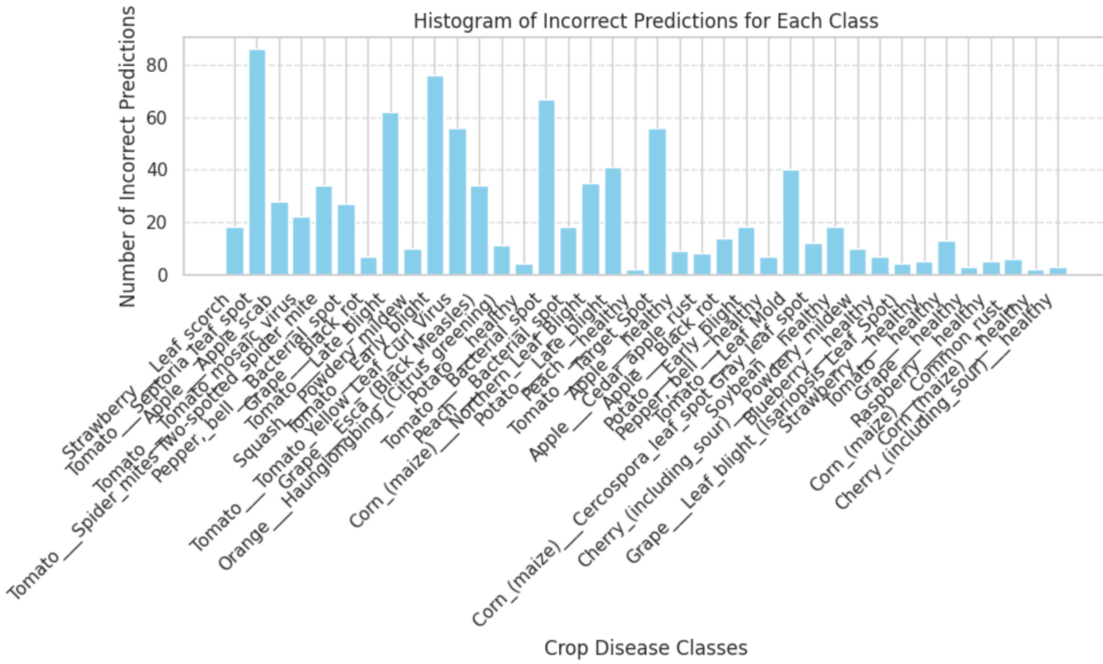
```
[ ]: # Now, extract and sort incorrect examples
incorrect_examples = []
for image, label, prob, correct in zip(images, labels, probs, corrects):
    if not correct:
        incorrect_examples.append((image, label, prob))

incorrect_examples.sort(reverse=True, key=lambda x: torch.max(x[2], dim=0).
 ↪values)
print(f"There are {len(incorrect_examples)} incorrect predictions.")
```

There are 878 incorrect predictions.

```
[ ]: incorrect_counts = count_incorrect_predictions(incorrect_examples)
class_names = [train_set.classes[label] for label in incorrect_counts.keys()]
```

```python
plot_incorrect_predictions_histogram(incorrect_counts, class_names)
```



[ ]:

```python
data = {'Class Name': class_names, 'Incorrect Counts': list(incorrect_counts.
 ↪values())}
df1 = pd.DataFrame(data)
df1 = df1.sort_values(by='Class Name')
```

[ ]:

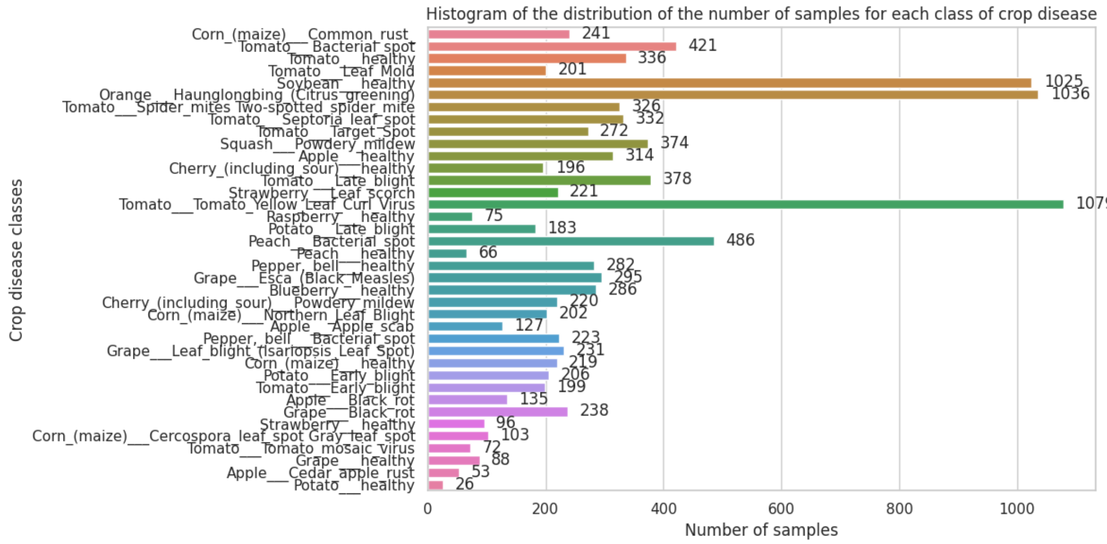
```python
class_counts = defaultdict(int)

for images, labels in testloader:
    for label in labels:
        class_counts[label.item()] += 1

class_names_dict = {label: train_set.classes[label] for label in class_counts.
 ↪keys()}
class_counts_named = {class_names_dict[label]: count for label, count in␣
 ↪class_counts.items()}
```

```
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork()
was called. os.fork() is incompatible with multithreaded code, and JAX is
multithreaded, so this will likely lead to a deadlock.
  self.pid = os.fork()
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork()
was called. os.fork() is incompatible with multithreaded code, and JAX is
multithreaded, so this will likely lead to a deadlock.
  self.pid = os.fork()
```

```
[ ]: plot_class_histogram(class_counts_named)
```




```
[ ]: class_counts_named = {class_names_dict[label]: count for label, count in␣
     ↪class_counts.items()}

     df2 = pd.DataFrame(list(class_counts_named.items()), columns=['Class Name',␣
     ↪'Counts'])
     df2 = df2.sort_values(by='Class Name')
```

```
[ ]: merged_df = pd.merge(df1, df2, on='Class Name', suffixes=('_incorrect', '_test'))
     # Calculate the rate of success for each class in percentage
     merged_df['Success Rate (%)'] = ((merged_df['Counts'] - merged_df['Incorrect␣
     ↪Counts']) / merged_df['Counts']) * 100
     merged_df = merged_df.sort_values(by='Class Name')
     merged_df
```

```
[ ]:                                        Class Name  Incorrect Counts  \
     0                           Apple___Apple_scab                28
     1                            Apple___Black_rot                14
     2                     Apple___Cedar_apple_rust                 8
     3                              Apple___healthy                 9
     4                          Blueberry___healthy                 7
     5       Cherry_(including_sour)___Powdery_mildew              10
     6              Cherry_(including_sour)___healthy                 3
     7   Corn_(maize)___Cercospora_leaf_spot Gray_leaf_...         12
     8                    Corn_(maize)___Common_rust_                 6
     9           Corn_(maize)___Northern_Leaf_Blight               35
     10                      Corn_(maize)___healthy                 2
```

11	Grape___Black_rot	7
12	Grape___Esca_(Black_Measles)	34
13	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	4
14	Grape___healthy	3
15	Orange___Haunglongbing_(Citrus_greening)	11
16	Peach___Bacterial_spot	18
17	Peach___healthy	2
18	Pepper,_bell___Bacterial_spot	27
19	Pepper,_bell___healthy	7
20	Potato___Early_blight	18
21	Potato___Late_blight	41
22	Potato___healthy	4
23	Raspberry___healthy	5
24	Soybean___healthy	18
25	Squash___Powdery_mildew	10
26	Strawberry___Leaf_scorch	18
27	Strawberry___healthy	5
28	Tomato___Bacterial_spot	67
29	Tomato___Early_blight	76
30	Tomato___Late_blight	62
31	Tomato___Leaf_Mold	40
32	Tomato___Septoria_leaf_spot	86
33	Tomato___Spider_mites Two-spotted_spider_mite	34
34	Tomato___Target_Spot	56
35	Tomato___Tomato_Yellow_Leaf_Curl_Virus	56
36	Tomato___Tomato_mosaic_virus	22
37	Tomato___healthy	13

	Counts	Success Rate (%)
0	127	77.952756
1	135	89.629630
2	53	84.905660
3	314	97.133758
4	286	97.552448
5	220	95.454545
6	196	98.469388
7	103	88.349515
8	241	97.510373
9	202	82.673267
10	219	99.086758
11	238	97.058824
12	295	88.474576
13	231	98.268398
14	88	96.590909
15	1036	98.938224
16	486	96.296296
17	66	96.969697

18	223	87.892377
19	282	97.517730
20	206	91.262136
21	183	77.595628
22	26	84.615385
23	75	93.333333
24	1025	98.243902
25	374	97.326203
26	221	91.855204
27	96	94.791667
28	421	84.085511
29	199	61.809045
30	378	83.597884
31	201	80.099502
32	332	74.096386
33	326	89.570552
34	272	79.411765
35	1079	94.810009
36	72	69.444444
37	336	96.130952

[ ]:
```
N_IMAGES = 25
plot_most_incorrect(incorrect_examples, N_IMAGES)
```