colorImages

May 19, 2024

1 Color 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)

[8]:
```
import sys
sys.path.append('/content/project')

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

[9]:
```python
repo_url = "https://github.com/digitalepidemiologylab/
↪plantvillage_deeplearning_paper_dataset.git"
clone_dir = "plantvillage_deeplearning_paper_dataset"
extracted_folder = "raw/color"

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

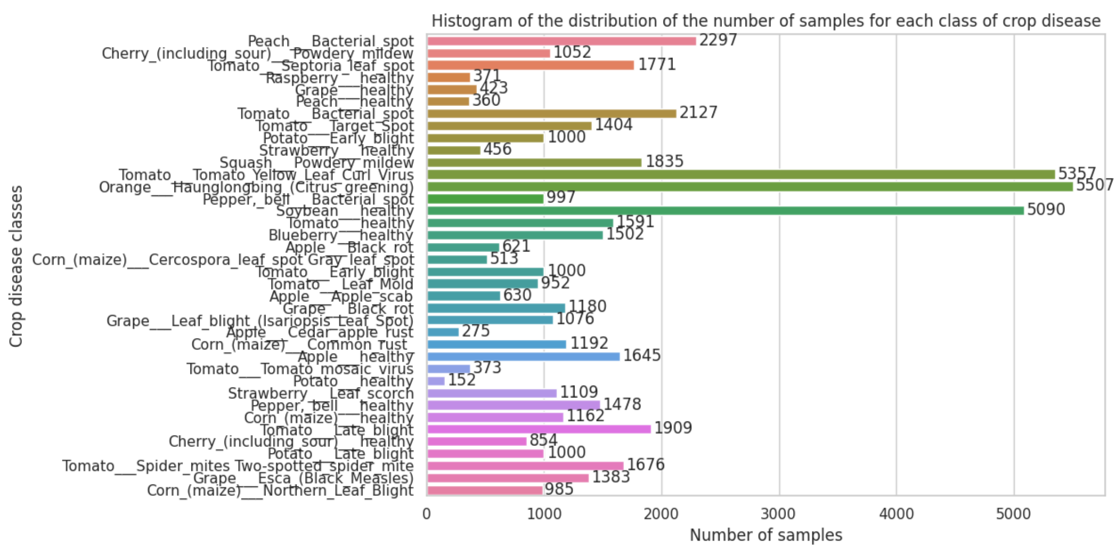
```
Folder 'raw/color' extracted successfully.
Directory 'Peach___Bacterial_spot' contains 2297 files.
Directory 'Cherry_(including_sour)___Powdery_mildew' contains 1052 files.
Directory 'Tomato___Septoria_leaf_spot' contains 1771 files.
Directory 'Raspberry___healthy' contains 371 files.
Directory 'Grape___healthy' 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 1383 files.
```

```
Directory 'Corn_(maize)___Northern_Leaf_Blight' contains 985 files.
Total number of files: 54305

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)': 1383,
'Corn_(maize)___Northern_Leaf_Blight': 985}
```

[9]:
```
{'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)': 1383,
'Corn_(maize)___Northern_Leaf_Blight': 985}
```

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



[10]:
```
# Spliiting the dataset
train_set = CropDiseaseDataset(root_dir=extracted_folder, train=True, validation=False, gray_scale=False, segmented=False)
validation_set = CropDiseaseDataset(root_dir=extracted_folder, train=False, validation=True, gray_scale=False, segmented=False)
test_set = CropDiseaseDataset(root_dir=extracted_folder, train=False, validation=False, gray_scale=False, segmented=False)
```

```python
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: '34321' images, Validation set: '7920' images, Test set: '10562'
images
```

[ ]:
```python
# 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')
```

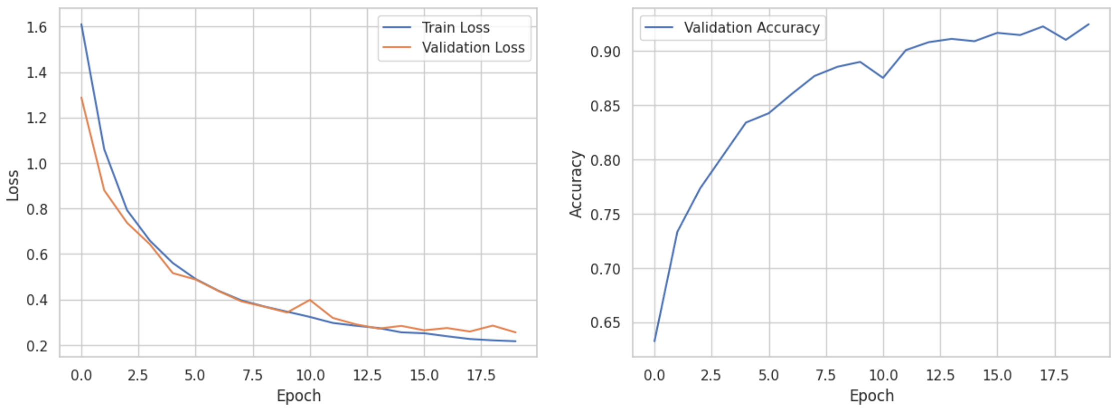
```
The model has 6,846,758 trainable parameters
```

[ ]:
```python
# 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: 1.2879, Test Accuracy: 63.24%, Train Loss: 1.6101,
Train Accuracy: 55.28%
| Epoch Time: 0m 41s
Epoch [2/20], Test Loss: 0.8799, Test Accuracy: 73.33%, Train Loss: 1.0603,
Train Accuracy: 68.44%
| Epoch Time: 0m 43s
Epoch [3/20], Test Loss: 0.7367, Test Accuracy: 77.35%, Train Loss: 0.7923,
Train Accuracy: 75.53%
| Epoch Time: 0m 42s
Epoch [4/20], Test Loss: 0.6423, Test Accuracy: 80.38%, Train Loss: 0.6583,
Train Accuracy: 79.37%
| Epoch Time: 0m 37s
Epoch [5/20], Test Loss: 0.5156, Test Accuracy: 83.41%, Train Loss: 0.5601,
Train Accuracy: 82.32%
| Epoch Time: 0m 40s
Epoch [6/20], Test Loss: 0.4875, Test Accuracy: 84.27%, Train Loss: 0.4897,
Train Accuracy: 84.44%
| Epoch Time: 0m 38s
```

```
Epoch [7/20], Test Loss: 0.4369, Test Accuracy: 86.04%, Train Loss: 0.4380,
Train Accuracy: 85.71%
| Epoch Time: 0m 40s
Epoch [8/20], Test Loss: 0.3915, Test Accuracy: 87.70%, Train Loss: 0.3958,
Train Accuracy: 87.02%
| Epoch Time: 0m 38s
Epoch [9/20], Test Loss: 0.3680, Test Accuracy: 88.55%, Train Loss: 0.3690,
Train Accuracy: 87.95%
| Epoch Time: 0m 41s
Epoch [10/20], Test Loss: 0.3425, Test Accuracy: 89.00%, Train Loss: 0.3461,
Train Accuracy: 88.62%
| Epoch Time: 0m 39s
Epoch [11/20], Test Loss: 0.3978, Test Accuracy: 87.53%, Train Loss: 0.3230,
Train Accuracy: 89.47%
| Epoch Time: 0m 40s
Epoch [12/20], Test Loss: 0.3188, Test Accuracy: 90.09%, Train Loss: 0.2967,
Train Accuracy: 90.49%
| Epoch Time: 0m 40s
Epoch [13/20], Test Loss: 0.2907, Test Accuracy: 90.82%, Train Loss: 0.2844,
Train Accuracy: 90.66%
| Epoch Time: 0m 41s
Epoch [14/20], Test Loss: 0.2717, Test Accuracy: 91.12%, Train Loss: 0.2743,
Train Accuracy: 91.03%
| Epoch Time: 0m 38s
Epoch [15/20], Test Loss: 0.2836, Test Accuracy: 90.91%, Train Loss: 0.2553,
Train Accuracy: 91.78%
| Epoch Time: 0m 38s
Epoch [16/20], Test Loss: 0.2644, Test Accuracy: 91.68%, Train Loss: 0.2512,
Train Accuracy: 91.61%
| Epoch Time: 0m 41s
Epoch [17/20], Test Loss: 0.2743, Test Accuracy: 91.48%, Train Loss: 0.2381,
Train Accuracy: 92.20%
| Epoch Time: 0m 38s
Epoch [18/20], Test Loss: 0.2589, Test Accuracy: 92.27%, Train Loss: 0.2260,
Train Accuracy: 92.76%
| Epoch Time: 0m 41s
Epoch [19/20], Test Loss: 0.2848, Test Accuracy: 91.04%, Train Loss: 0.2201,
Train Accuracy: 92.94%
| Epoch Time: 0m 39s
Epoch [20/20], Test Loss: 0.2551, Test Accuracy: 92.47%, Train Loss: 0.2162,
Train Accuracy: 93.15%
| Epoch Time: 0m 40s
```

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



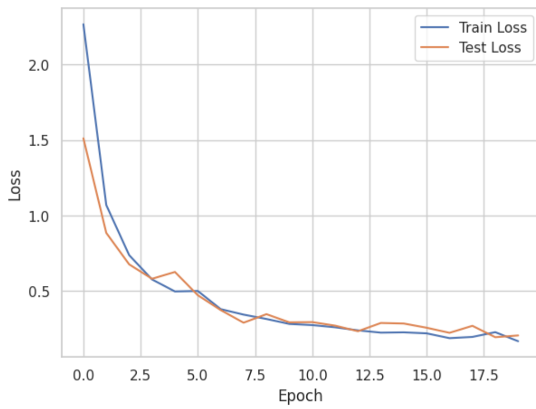
[ ]:
```
# Combining train and validation datasets for testing the model
combined_train_set = ConcatDataset([train_set, validation_set])
combined_train_loader = torch.utils.data.DataLoader(combined_train_set,␣
↪batch_size=64, shuffle=True, num_workers=2)
```

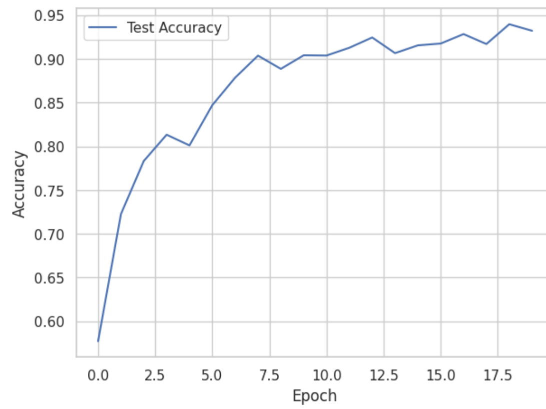
[ ]:
```
# Test the model
network = CNN(gray_scale=False).to(device)
network.apply(initialize_parameters)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(network.parameters(), lr=learning_rate)

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: 1.5123, Test Accuracy: 57.72%, Train Loss: 2.2678,
Train Accuracy: 43.27%
| Epoch Time: 0m 52s
Epoch [2/20], Test Loss: 0.8855, Test Accuracy: 72.25%, Train Loss: 1.0699,
Train Accuracy: 67.85%
| Epoch Time: 0m 51s
Epoch [3/20], Test Loss: 0.6771, Test Accuracy: 78.32%, Train Loss: 0.7381,
Train Accuracy: 76.90%
| Epoch Time: 0m 52s
Epoch [4/20], Test Loss: 0.5815, Test Accuracy: 81.32%, Train Loss: 0.5771,
Train Accuracy: 81.66%
| Epoch Time: 0m 50s
Epoch [5/20], Test Loss: 0.6267, Test Accuracy: 80.10%, Train Loss: 0.4976,
Train Accuracy: 84.09%
| Epoch Time: 0m 51s
Epoch [6/20], Test Loss: 0.4727, Test Accuracy: 84.70%, Train Loss: 0.5007,
Train Accuracy: 84.36%
| Epoch Time: 0m 51s
```

```
Epoch [7/20], Test Loss: 0.3743, Test Accuracy: 87.84%, Train Loss: 0.3804,
Train Accuracy: 87.63%
| Epoch Time: 0m 51s
Epoch [8/20], Test Loss: 0.2905, Test Accuracy: 90.36%, Train Loss: 0.3441,
Train Accuracy: 88.76%
| Epoch Time: 0m 52s
Epoch [9/20], Test Loss: 0.3471, Test Accuracy: 88.85%, Train Loss: 0.3152,
Train Accuracy: 89.90%
| Epoch Time: 0m 53s
Epoch [10/20], Test Loss: 0.2930, Test Accuracy: 90.40%, Train Loss: 0.2824,
Train Accuracy: 90.86%
| Epoch Time: 0m 51s
Epoch [11/20], Test Loss: 0.2946, Test Accuracy: 90.37%, Train Loss: 0.2747,
Train Accuracy: 91.09%
| Epoch Time: 0m 51s
Epoch [12/20], Test Loss: 0.2710, Test Accuracy: 91.26%, Train Loss: 0.2601,
Train Accuracy: 91.67%
| Epoch Time: 0m 49s
Epoch [13/20], Test Loss: 0.2340, Test Accuracy: 92.44%, Train Loss: 0.2399,
Train Accuracy: 92.08%
| Epoch Time: 0m 50s
Epoch [14/20], Test Loss: 0.2887, Test Accuracy: 90.65%, Train Loss: 0.2246,
Train Accuracy: 92.72%
| Epoch Time: 0m 49s
Epoch [15/20], Test Loss: 0.2851, Test Accuracy: 91.54%, Train Loss: 0.2266,
Train Accuracy: 92.58%
| Epoch Time: 0m 50s
Epoch [16/20], Test Loss: 0.2572, Test Accuracy: 91.74%, Train Loss: 0.2200,
Train Accuracy: 92.90%
| Epoch Time: 0m 49s
Epoch [17/20], Test Loss: 0.2235, Test Accuracy: 92.82%, Train Loss: 0.1882,
Train Accuracy: 93.98%
| Epoch Time: 0m 50s
Epoch [18/20], Test Loss: 0.2699, Test Accuracy: 91.69%, Train Loss: 0.1964,
Train Accuracy: 93.77%
| Epoch Time: 0m 49s
Epoch [19/20], Test Loss: 0.1944, Test Accuracy: 93.95%, Train Loss: 0.2278,
Train Accuracy: 92.86%
| Epoch Time: 0m 52s
Epoch [20/20], Test Loss: 0.2062, Test Accuracy: 93.20%, Train Loss: 0.1679,
Train Accuracy: 94.69%
| Epoch Time: 0m 49s
```

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

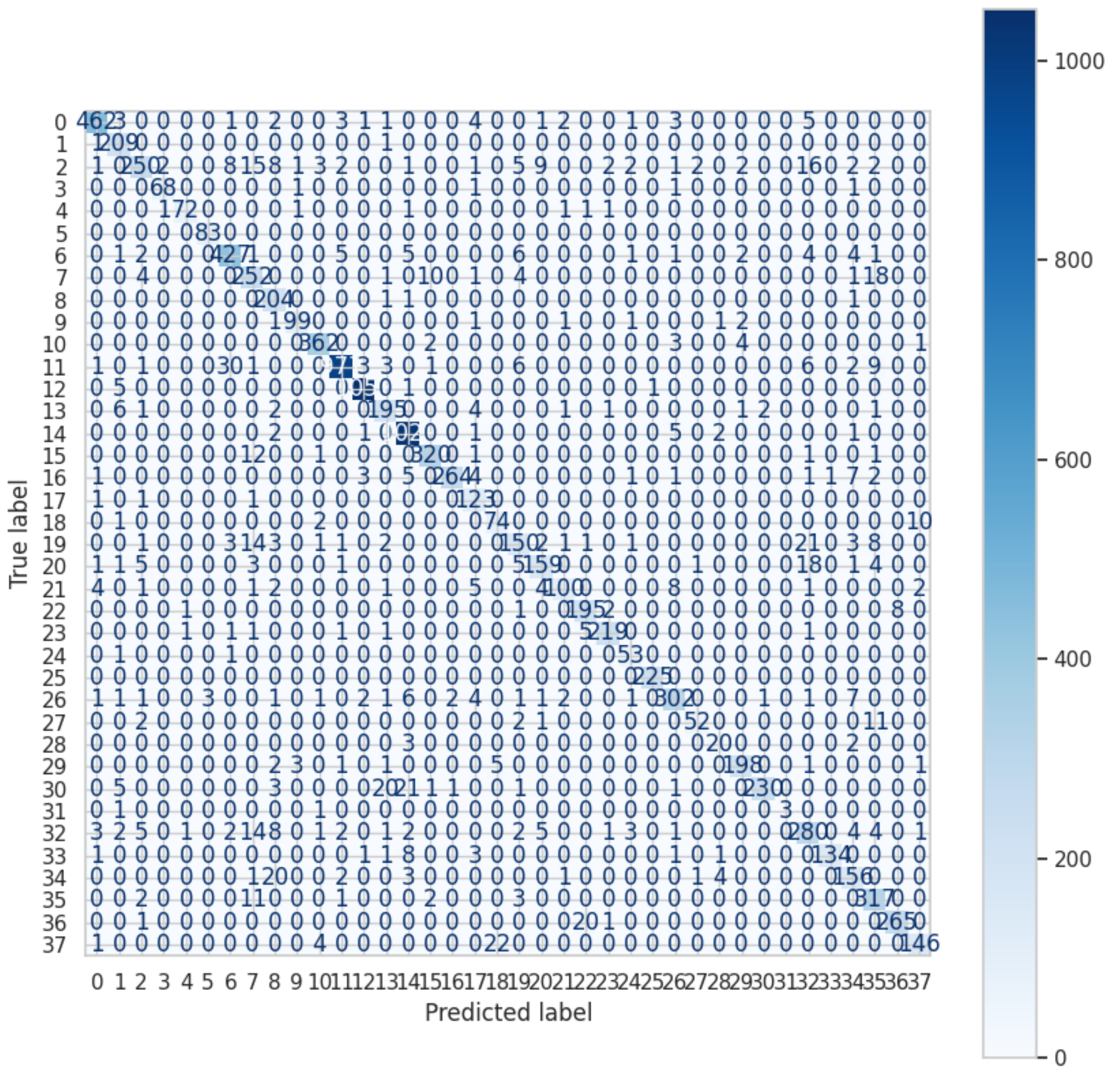




[83]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
network.load_state_dict(torch.load('model.pt'))
images, labels, probs, corrects = get_predictions(network, testloader, device)
pred_labels = torch.argmax(probs, 1)

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

```
There are 10562 correct predictions.
```

[84]:
```
plot_confusion_matrix(labels, pred_labels)
```

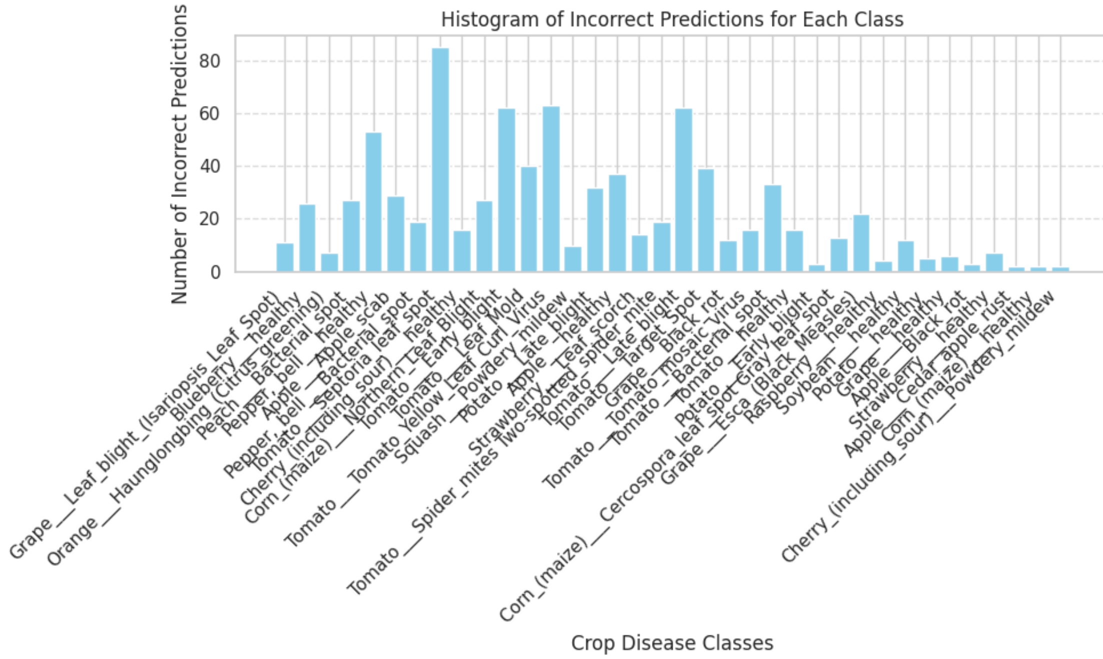



```
[85]: # Now, extract and sort incorrect examples
incorrect_examples = []
for image, label, prob, correct in zip(images, labels, probs, corrects):
    if not correct:
        incorrect_examples.append((image, label, prob))

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 836 incorrect predictions.
```

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

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

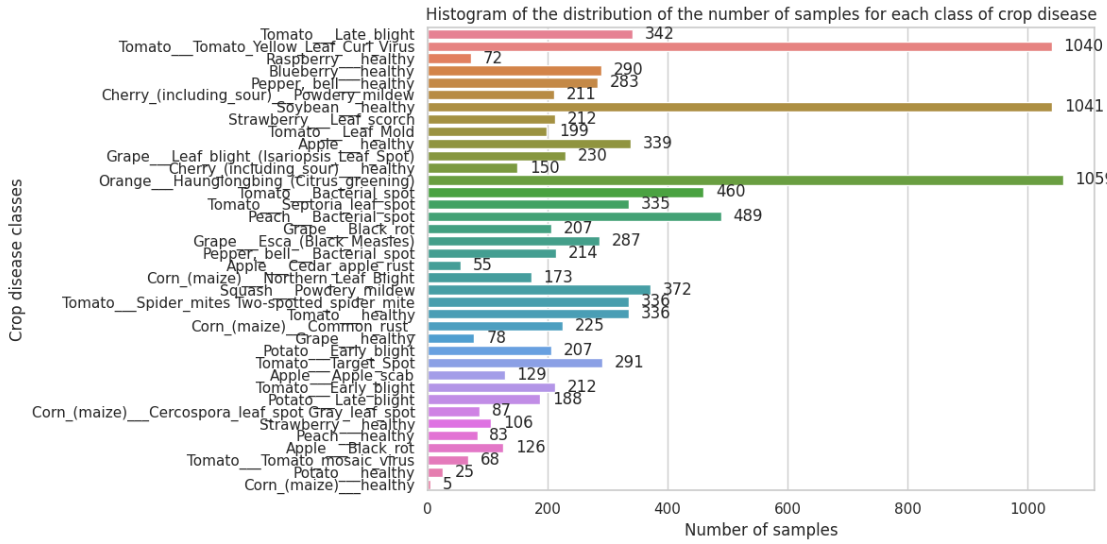


[87]:
```
data = {'Class Name': class_names, 'Incorrect Counts': list(incorrect_counts.
↪values())}
df1 = pd.DataFrame(data)
df1 = df1.sort_values(by='Class Name')
```

[88]:
```
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()
```

[89]:
```python
plot_class_histogram(class_counts_named)
```



[90]:
```python
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')
```

[91]:
```python
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
```

[91]:
```
                                           Class Name  Incorrect Counts  \
0                                  Apple___Apple_scab                29
1                                   Apple___Black_rot                 3
2                            Apple___Cedar_apple_rust                 2
3                                     Apple___healthy                37
4                                 Blueberry___healthy                26
5          Cherry_(including_sour)___Powdery_mildew                 2
6                   Cherry_(including_sour)___healthy                16
7     Corn_(maize)___Cercospora_leaf_spot Gray_leaf_...                13
8                 Corn_(maize)___Northern_Leaf_Blight                27
9                              Corn_(maize)___healthy                 2
10                                  Grape___Black_rot                12
```

11	Grape___Esca_(Black_Measles)	22
12	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	11
13	Grape___healthy	6
14	Orange___Haunglongbing_(Citrus_greening)	7
15	Peach___Bacterial_spot	27
16	Pepper,_bell___Bacterial_spot	19
17	Pepper,_bell___healthy	53
18	Potato___Early_blight	3
19	Potato___Late_blight	32
20	Potato___healthy	5
21	Raspberry___healthy	4
22	Soybean___healthy	12
23	Squash___Powdery_mildew	10
24	Strawberry___Leaf_scorch	14
25	Strawberry___healthy	7
26	Tomato___Bacterial_spot	33
27	Tomato___Early_blight	62
28	Tomato___Late_blight	62
29	Tomato___Leaf_Mold	40
30	Tomato___Septoria_leaf_spot	85
31	Tomato___Spider_mites Two-spotted_spider_mite	19
32	Tomato___Target_Spot	39
33	Tomato___Tomato_Yellow_Leaf_Curl_Virus	63
34	Tomato___Tomato_mosaic_virus	16
35	Tomato___healthy	16

	Counts	Success Rate (%)
0	129	77.519380
1	126	97.619048
2	55	96.363636
3	339	89.085546
4	290	91.034483
5	211	99.052133
6	150	89.333333
7	87	85.057471
8	173	84.393064
9	5	60.000000
10	207	94.202899
11	287	92.334495
12	230	95.217391
13	78	92.307692
14	1059	99.338999
15	489	94.478528
16	214	91.121495
17	283	81.272085
18	207	98.550725
19	188	82.978723

20	25	80.000000
21	72	94.444444
22	1041	98.847262
23	372	97.311828
24	212	93.396226
25	106	93.396226
26	460	92.826087
27	212	70.754717
28	342	81.871345
29	199	79.899497
30	335	74.626866
31	336	94.345238
32	291	86.597938
33	1040	93.942308
34	68	76.470588
35	336	95.238095

[92]:
```
N_IMAGES = 25
plot_most_incorrect(incorrect_examples, N_IMAGES)
```