# Knowledge Informed Neural Network (KINN)-Inspired Lightweight Architecture for Plant Leaf Disease Classification


```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
dataset_path = '/kaggle/input/plant-disease-detection-dataset/train/'
image_paths = []
labels = []

for folder in os.listdir(dataset_path):
    folder_path = os.path.join(dataset_path, folder)
    if os.path.isdir(folder_path):
        for img in os.listdir(folder_path):
            img_path = os.path.join(folder_path, img)
            if img.endswith(('.jpg', '.jpeg', '.png')):
                image_paths.append(img_path)
                labels.append(folder)

df = pd.DataFrame({'image_path': image_paths, 'label': labels})

df.head()
```

| | image_path | label |
|---|---|---|
| 0 | /kaggle/input/plant-disease-detection-dataset/... | corn leaf blight |
| 1 | /kaggle/input/plant-disease-detection-dataset/... | corn leaf blight |
| 2 | /kaggle/input/plant-disease-detection-dataset/... | corn leaf blight |
| 3 | /kaggle/input/plant-disease-detection-dataset/... | corn leaf blight |
| 4 | /kaggle/input/plant-disease-detection-dataset/... | corn leaf blight |

```
df.tail()
```

| | image_path | label |
|---|---|---|
| 1093 | /kaggle/input/plant-disease-detection-dataset/... | apple leaf healthy |
| 1094 | /kaggle/input/plant-disease-detection-dataset/... | apple leaf healthy |
| 1095 | /kaggle/input/plant-disease-detection-dataset/... | apple leaf healthy |
| 1096 | /kaggle/input/plant-disease-detection-dataset/... | apple leaf healthy |
| 1097 | /kaggle/input/plant-disease-detection-dataset/... | apple leaf healthy |

```
df.shape
```

```
(1098, 2)
```

```
df.columns
```

```
Index(['image_path', 'label'], dtype='object')
```

```
df.duplicated().sum()
```

```
0
```

```
df.isnull().sum()
```

```
image_path    0
label         0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1098 entries, 0 to 1097
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   image_path  1098 non-null   object
 1   label       1098 non-null   object
dtypes: object(2)
memory usage: 17.3+ KB
```

```
df['label'].unique()
```

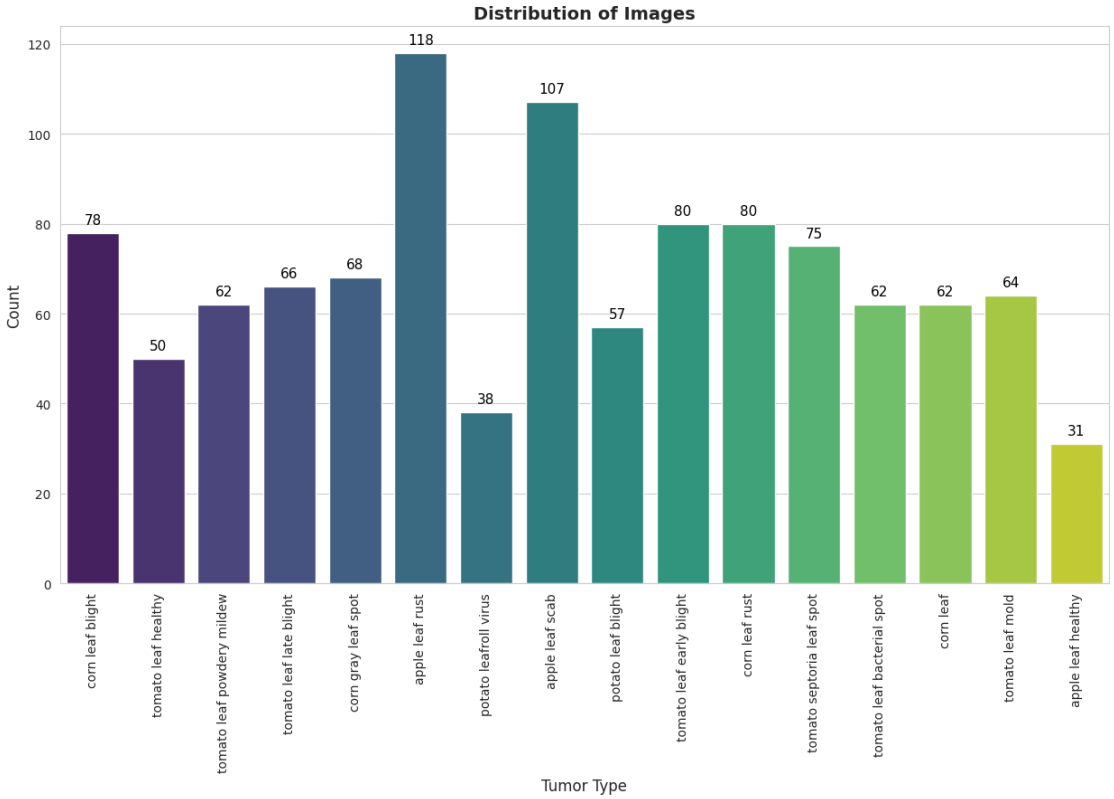
```
array(['corn leaf blight', 'tomato leaf healthy',
       'tomato leaf powdery mildew', 'tomato leaf late blight',
       'corn gray leaf spot', 'apple leaf rust', 'potato leafroll virus',
       'apple leaf scab', 'potato leaf blight',
       'tomato leaf early blight', 'corn leaf rust',
       'tomato septoria leaf spot', 'tomato leaf bacterial spot',
       'corn leaf', 'tomato leaf mold', 'apple leaf healthy'],
      dtype=object)
```

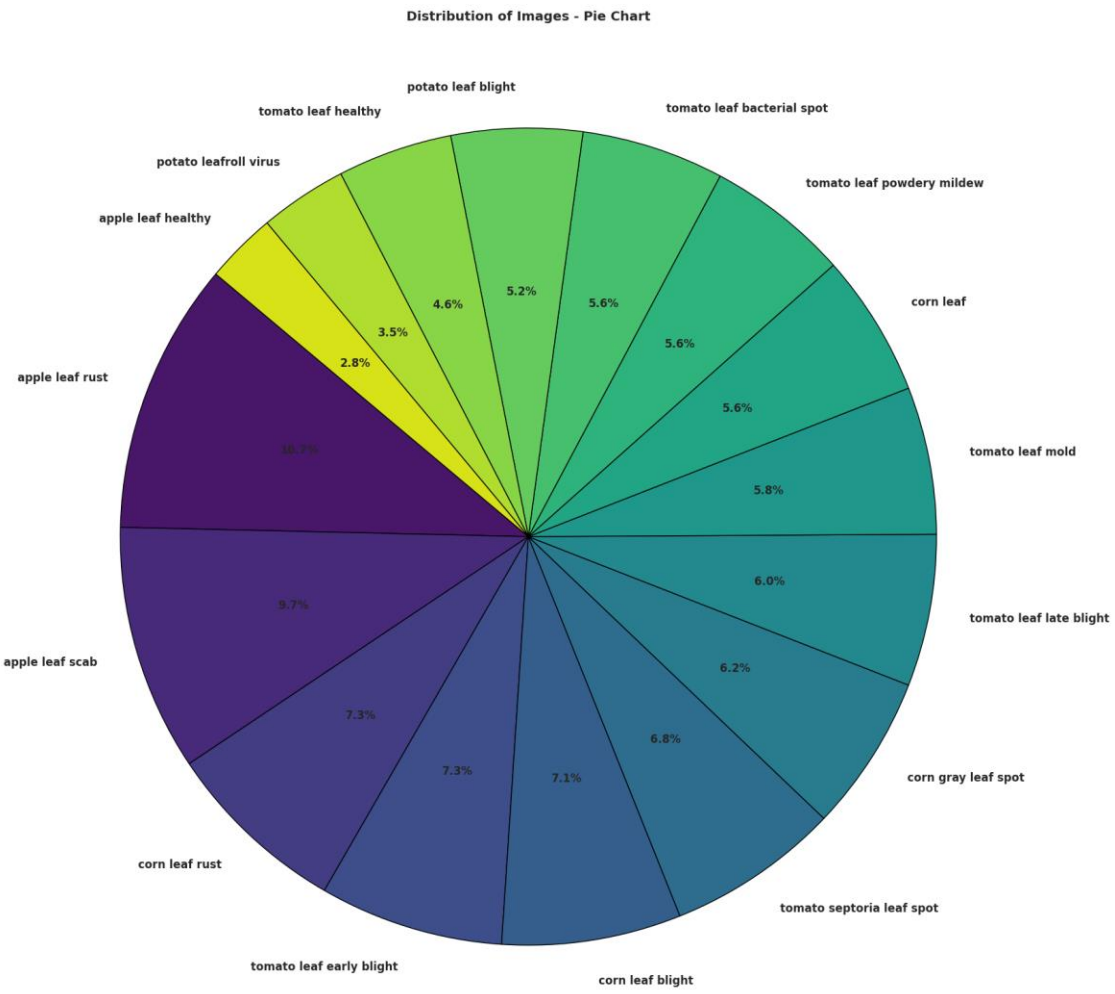
```
df['label'].value_counts()
```

```
label
apple leaf rust               118
apple leaf scab               107
corn leaf rust                 80
tomato leaf early blight       80
corn leaf blight               78
tomato septoria leaf spot      75
corn gray leaf spot            68
tomato leaf late blight        66
tomato leaf mold               64
corn leaf                      62
tomato leaf powdery mildew     62
tomato leaf bacterial spot     62
potato leaf blight             57
tomato leaf healthy            50
potato leafroll virus          38
apple leaf healthy             31
Name: count, dtype: int64
```

```
sns.set_style("whitegrid")

fig, ax = plt.subplots(figsize=(15, 8))
sns.countplot(data=df, x="label", palette="viridis", ax=ax)

ax.set_title("Distribution of Images", fontsize=14, fontweight='bold')
ax.set_xlabel("Tumor Type", fontsize=12)
ax.set_ylabel("Count", fontsize=12)
```

```
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='bottom', fontsize=11, color='black',
                xytext=(0, 5), textcoords='offset points')

plt.xticks(rotation = 90)
plt.show()

label_counts = df["label"].value_counts()

fig, ax = plt.subplots(figsize=(20, 20))
colors = sns.color_palette("viridis", len(label_counts))

ax.pie(label_counts, labels=label_counts.index, autopct='%1.1f%%',
       startangle=140, colors=colors, textprops={'fontsize': 12, 'weight':
'bold'},
       wedgeprops={'edgecolor': 'black', 'linewidth': 1})

ax.set_title("Distribution of Images - Pie Chart", fontsize=14,
fontweight='bold')

plt.show()
```






```python
from PIL import Image

for label in df['label'].unique():
    label_images = df[df['label'] == label]['image_path'].head(5).tolist()
    fig, axes = plt.subplots(1, 5, figsize=(15, 3))
    fig.suptitle(label)
    for i, img_path in enumerate(label_images):
        img = Image.open(img_path)
        axes[i].imshow(img)
        axes[i].axis('off')
    plt.show()
```

corn leaf blight





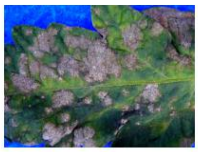

tomato leaf healthy





tomato leaf powdery mildew





tomato leaf late blight





corn gray leaf spot





apple leaf rust





potato leafroll virus





apple leaf scab





potato leaf blight



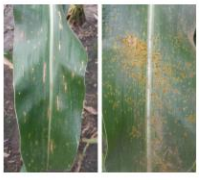


tomato leaf early blight





corn leaf rust





tomato septoria leaf spot





tomato leaf bacterial spot

corn leaf

tomato leaf mold

apple leaf healthy

```
from sklearn.utils import resample

max_count = df['label'].value_counts().max()

dfs = []
for category in df['label'].unique():
    class_subset = df[df['label'] == category]
    class_upsampled = resample(class_subset,
                               replace=True,
                               n_samples=max_count,
                               random_state=42)
    dfs.append(class_upsampled)

df_balanced = pd.concat(dfs).sample(frac=1,
random_state=42).reset_index(drop=True)
```

```
df = df_balanced
```

```
df
```

```
                                             image_path  \
0     /kaggle/input/plant-disease-detection-dataset/...
1     /kaggle/input/plant-disease-detection-dataset/...
2     /kaggle/input/plant-disease-detection-dataset/...
3     /kaggle/input/plant-disease-detection-dataset/...
4     /kaggle/input/plant-disease-detection-dataset/...
...                                                 ...
1883  /kaggle/input/plant-disease-detection-dataset/...
1884  /kaggle/input/plant-disease-detection-dataset/...
1885  /kaggle/input/plant-disease-detection-dataset/...
1886  /kaggle/input/plant-disease-detection-dataset/...
1887  /kaggle/input/plant-disease-detection-dataset/...

                           label
0                 corn leaf rust
1              potato leaf blight
2                tomato leaf mold
3     tomato leaf bacterial spot
4     tomato leaf bacterial spot
...                          ...
1883    tomato leaf early blight
1884               corn leaf rust
1885              apple leaf scab
1886  tomato leaf bacterial spot
1887    tomato leaf early blight

[1888 rows x 2 columns]
```

```
df['label'].value_counts()
```

```
label
corn leaf rust                118
potato leaf blight            118
tomato leaf mold              118
tomato leaf bacterial spot    118
apple leaf rust               118
apple leaf scab               118
tomato leaf healthy           118
tomato septoria leaf spot     118
potato leafroll virus         118
tomato leaf powdery mildew    118
corn gray leaf spot           118
apple leaf healthy            118
tomato leaf late blight       118
corn leaf blight              118
corn leaf                     118
```

```
tomato leaf early blight      118
Name: count, dtype: int64
```

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
import torchvision.transforms as transforms
from torchvision.models import resnet18
from PIL import Image
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"Using device: {device}")

print(f"Dataset shape: {df.shape}")
print(df['label'].value_counts())

class PlantDiseaseDataset(Dataset):
    def __init__(self, df, transform=None):
        self.df = df.reset_index(drop=True)
        self.transform = transform
        self.label_to_idx = {label: idx for idx, label in
enumerate(sorted(self.df['label'].unique()))}

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        img_path = self.df.iloc[idx]['image_path']
        label = self.df.iloc[idx]['label']
        try:
            image = Image.open(img_path).convert('RGB')
        except FileNotFoundError:
            print(f"Warning: Could not load {img_path}")
            image = Image.new('RGB', (224, 224), color=(128, 128, 128))
        if self.transform:
            image = self.transform(image)
        label_idx = self.label_to_idx[label]
        return image, label_idx

train_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(),
```

```
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225])
])

val_test_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225])
])

train_df, temp_df = train_test_split(df, test_size=0.2, stratify=df['label'],
random_state=42)
val_df, test_df = train_test_split(temp_df, test_size=0.5,
stratify=temp_df['label'], random_state=42)

train_dataset = PlantDiseaseDataset(train_df, train_transform)
val_dataset = PlantDiseaseDataset(val_df, val_test_transform)
test_dataset = PlantDiseaseDataset(test_df, val_test_transform)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True,
num_workers=2)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False,
num_workers=2)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False,
num_workers=2)

num_classes = len(train_dataset.label_to_idx)
idx_to_label = {i: label for label, i in train_dataset.label_to_idx.items()}
labels_list = list(idx_to_label.values())

class CompressionBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()
        self.dwconv = nn.Conv2d(in_channels, in_channels, 3, padding=1,
groups=in_channels)
        self.pwconv = nn.Conv2d(in_channels, out_channels, 1)
        self.bn = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        return self.relu(self.bn(self.pwconv(self.dwconv(x))))

class AggregationModule(nn.Module):
    def __init__(self, channels):
        super().__init__()
        self.attn = nn.Sequential(
            nn.AdaptiveAvgPool2d(1),
            nn.Conv2d(channels, channels // 8, 1),
```

```
            nn.ReLU(),
            nn.Conv2d(channels // 8, channels, 1),
            nn.Sigmoid()
        )

    def forward(self, x, skip):
        attn = self.attn(x)
        return x * attn + skip

class DistillationHead(nn.Module):
    def __init__(self, in_features, num_classes, temp=4):
        super().__init__()
        self.fc = nn.Linear(in_features, num_classes)
        self.temp = temp

    def forward(self, x):
        return self.fc(x) / self.temp

class KINN(nn.Module):
    def __init__(self, num_classes):
        super().__init__()
        base = resnet18(pretrained=True)
        self.stem = nn.Sequential(
            base.conv1,
            base.bn1,
            base.relu,
            base.maxpool
        )
        self.layer1 = base.layer1
        self.compress1 = CompressionBlock(64, 64)
        self.layer2 = base.layer2
        self.compress2 = CompressionBlock(128, 128)
        self.layer3 = base.layer3
        self.layer4 = base.layer4
        self.avgpool = base.avgpool
        self.agg = AggregationModule(512)
        self.head1 = DistillationHead(512, num_classes)
        self.head2 = DistillationHead(512, num_classes)
        self.final_head = nn.Linear(512, num_classes)

    def forward(self, x):
        x = self.stem(x)
        x = self.layer1(x)
        feat_early = self.compress1(x)
        x = self.layer2(feat_early)
        feat_mid = self.compress2(x)
        x = self.layer3(feat_mid)
        x = self.layer4(x)
        x = self.avgpool(x)
```

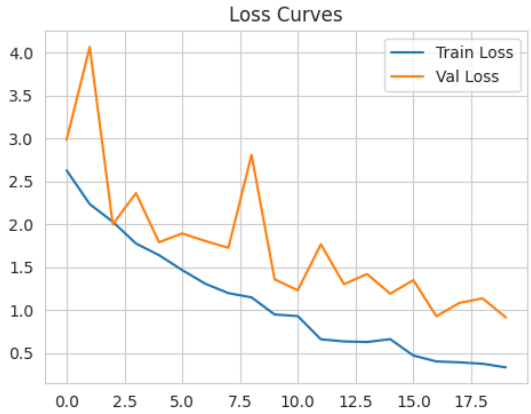
```
        feat_late = x.flatten(1)
        feat_early_pooled = nn.AdaptiveAvgPool2d(1)(feat_early).flatten(1)
        proj_early = nn.Linear(64, 512).to(x.device)(feat_early_pooled)
        agg_feat = self.agg(feat_late.unsqueeze(-1).unsqueeze(-1),
proj_early.unsqueeze(-1).unsqueeze(-1)).flatten(1)
        agg_feat = feat_late + 0.1 * feat_late
        logits1 = self.head1(feat_late)
        logits2 = self.head2(feat_late)
        final_logits = self.final_head(agg_feat)
        return final_logits, logits1, logits2

def distillation_loss(logits_student, logits_teacher, temp=4, alpha=0.5):
    soft_teacher = nn.functional.softmax(logits_teacher / temp, dim=1)
    soft_student = nn.LogSoftmax(dim=1)(logits_student)
    return nn.KLDivLoss(reduction='batchmean')(soft_student, soft_teacher) *
(temp ** 2) * alpha

model = KINN(num_classes).to(device)
optimizer = optim.Adam(model.parameters(), lr=1e-3)
criterion = nn.CrossEntropyLoss()
num_epochs = 20

train_losses, train_accs = [], []
val_losses, val_accs = [], []

for epoch in range(num_epochs):
    model.train()
    train_loss, train_correct = 0, 0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        logits, logits_s, logits_t = model(images)
        loss_ce = criterion(logits, labels)
        loss_dist = distillation_loss(logits_s, logits_t)
        loss = loss_ce + 0.1 * loss_dist
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
        train_correct += (logits.argmax(1) == labels).sum().item()

    train_losses.append(train_loss / len(train_loader))
    train_accs.append(train_correct / len(train_dataset))

    model.eval()
    val_loss, val_correct = 0, 0
    with torch.no_grad():
        for images, labels in val_loader:
            images, labels = images.to(device), labels.to(device)
            logits, _, _ = model(images)
```

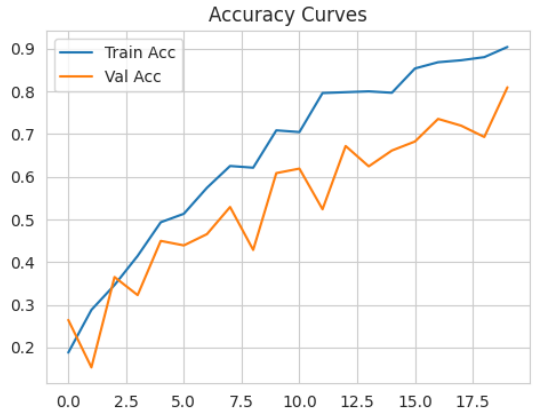
```
            val_loss += criterion(logits, labels).item()
            val_correct += (logits.argmax(1) == labels).sum().item()

    val_losses.append(val_loss / len(val_loader))
    val_accs.append(val_correct / len(val_dataset))

    print(f"Epoch {epoch+1}/{num_epochs}: Train Loss: {train_losses[-1]:.4f},
Acc: {train_accs[-1]:.4f} | Val Loss: {val_losses[-1]:.4f}, Acc: {val_accs[-
1]:.4f}")

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(train_losses, label='Train Loss')
ax1.plot(val_losses, label='Val Loss')
ax1.set_title('Loss Curves')
ax1.legend()

ax2.plot(train_accs, label='Train Acc')
ax2.plot(val_accs, label='Val Acc')
ax2.set_title('Accuracy Curves')
ax2.legend()
plt.savefig('/kaggle/working/training_plots.png')
plt.show()

plt.figure(figsize=(12, 6))
df['label'].value_counts().plot(kind='bar')
plt.title('Class Distribution')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.savefig('/kaggle/working/class_dist.png')
plt.show()

model.eval()
all_preds, all_labels = [], []
with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        logits, _, _ = model(images)
        preds = logits.argmax(1).cpu().numpy()
        all_preds.extend(preds)
        all_labels.extend(labels.numpy())

cm = confusion_matrix(all_labels, all_preds)
plt.figure(figsize=(14, 12))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels_list,
yticklabels=labels_list)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.xticks(rotation=45, ha='right')
```

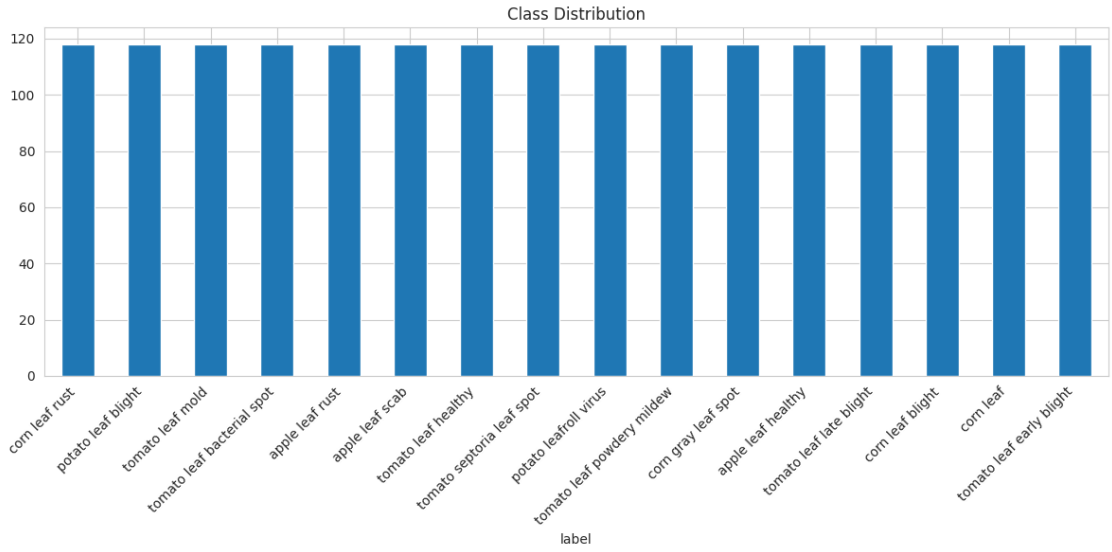
```
plt.yticks(rotation=0)
plt.tight_layout()
plt.savefig('/kaggle/working/confusion_matrix.png')
plt.show()

report = classification_report(all_labels, all_preds,
target_names=labels_list, digits=4)
print("Classification Report:\n", report)

with open('/kaggle/working/classification_report.txt', 'w') as f:
    f.write(report)

print("Fixed and running! Outputs in /kaggle/working/.")
```

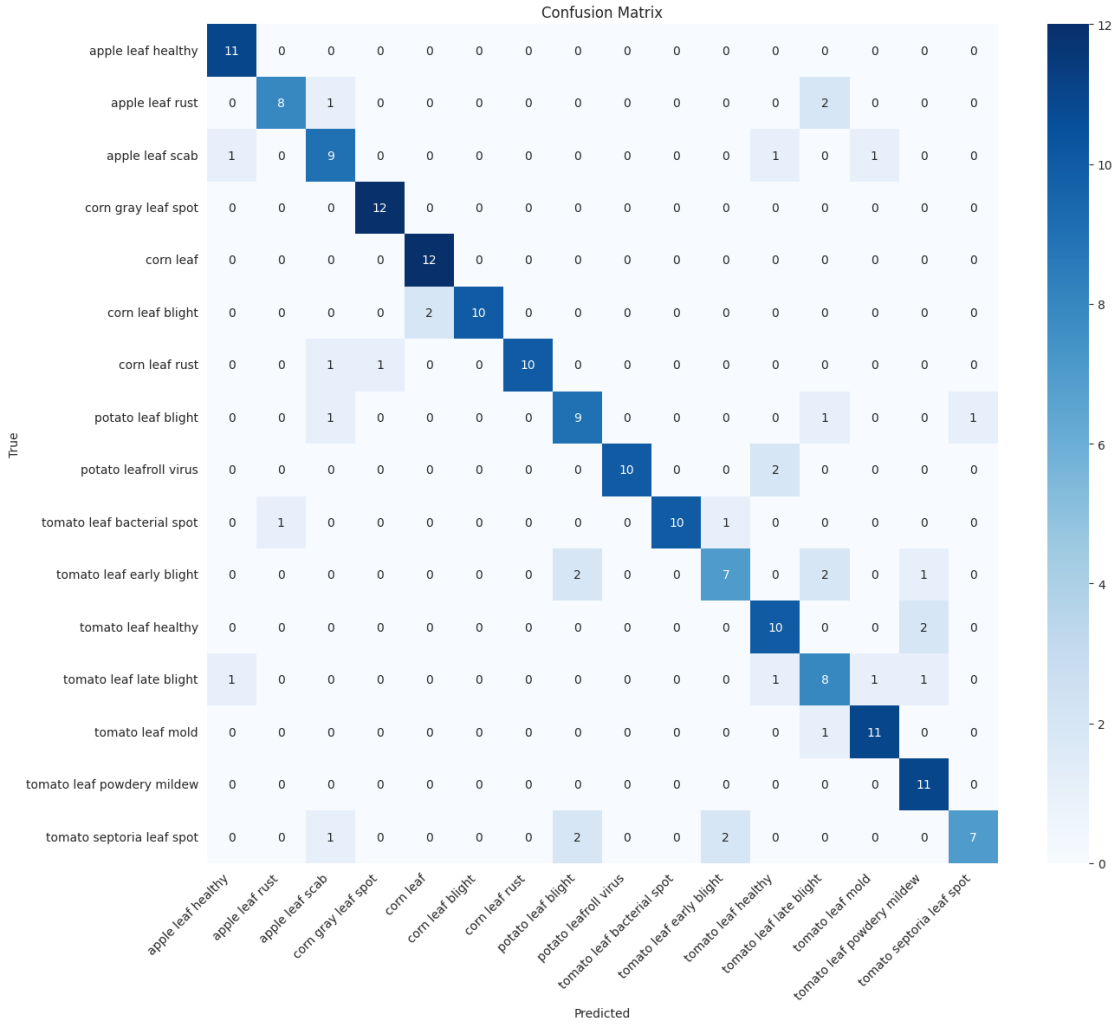
```
Using device: cuda
Dataset shape: (1888, 2)
label
corn leaf rust                118
potato leaf blight            118
tomato leaf mold              118
tomato leaf bacterial spot    118
apple leaf rust               118
apple leaf scab               118
tomato leaf healthy           118
tomato septoria leaf spot     118
potato leafroll virus         118
tomato leaf powdery mildew    118
corn gray leaf spot           118
apple leaf healthy            118
tomato leaf late blight       118
corn leaf blight              118
corn leaf                     118
tomato leaf early blight      118
Name: count, dtype: int64
Epoch 1/20: Train Loss: 2.6306, Acc: 0.1881 | Val Loss: 2.9876, Acc: 0.2646
Epoch 2/20: Train Loss: 2.2369, Acc: 0.2881 | Val Loss: 4.0683, Acc: 0.1534
Epoch 3/20: Train Loss: 2.0306, Acc: 0.3464 | Val Loss: 2.0011, Acc: 0.3651
Epoch 4/20: Train Loss: 1.7783, Acc: 0.4146 | Val Loss: 2.3647, Acc: 0.3228
Epoch 5/20: Train Loss: 1.6422, Acc: 0.4934 | Val Loss: 1.7930, Acc: 0.4497
Epoch 6/20: Train Loss: 1.4686, Acc: 0.5132 | Val Loss: 1.8960, Acc: 0.4392
Epoch 7/20: Train Loss: 1.3093, Acc: 0.5742 | Val Loss: 1.8074, Acc: 0.4656
Epoch 8/20: Train Loss: 1.1997, Acc: 0.6252 | Val Loss: 1.7282, Acc: 0.5291
Epoch 9/20: Train Loss: 1.1506, Acc: 0.6212 | Val Loss: 2.8109, Acc: 0.4286
Epoch 10/20: Train Loss: 0.9507, Acc: 0.7086 | Val Loss: 1.3606, Acc: 0.6085
Epoch 11/20: Train Loss: 0.9322, Acc: 0.7046 | Val Loss: 1.2326, Acc: 0.6190
Epoch 12/20: Train Loss: 0.6621, Acc: 0.7960 | Val Loss: 1.7690, Acc: 0.5238
Epoch 13/20: Train Loss: 0.6372, Acc: 0.7980 | Val Loss: 1.3042, Acc: 0.6720
Epoch 14/20: Train Loss: 0.6305, Acc: 0.8000 | Val Loss: 1.4227, Acc: 0.6243
Epoch 15/20: Train Loss: 0.6636, Acc: 0.7967 | Val Loss: 1.1929, Acc: 0.6614
Epoch 16/20: Train Loss: 0.4732, Acc: 0.8536 | Val Loss: 1.3513, Acc: 0.6825
```

```
Epoch 17/20: Train Loss: 0.4038, Acc: 0.8682 | Val Loss: 0.9305, Acc: 0.7354
Epoch 18/20: Train Loss: 0.3935, Acc: 0.8728 | Val Loss: 1.0869, Acc: 0.7196
Epoch 19/20: Train Loss: 0.3770, Acc: 0.8801 | Val Loss: 1.1382, Acc: 0.6931
Epoch 20/20: Train Loss: 0.3347, Acc: 0.9040 | Val Loss: 0.9169, Acc: 0.8095
```






Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| apple leaf healthy | 0.8462 | 1.0000 | 0.9167 | 11 |
| apple leaf rust | 0.8889 | 0.7273 | 0.8000 | 11 |
| apple leaf scab | 0.6923 | 0.7500 | 0.7200 | 12 |
| corn gray leaf spot | 0.9231 | 1.0000 | 0.9600 | 12 |
| corn leaf | 0.8571 | 1.0000 | 0.9231 | 12 |
| corn leaf blight | 1.0000 | 0.8333 | 0.9091 | 12 |
| corn leaf rust | 1.0000 | 0.8333 | 0.9091 | 12 |
| potato leaf blight | 0.6923 | 0.7500 | 0.7200 | 12 |
| potato leafroll virus | 1.0000 | 0.8333 | 0.9091 | 12 |
| tomato leaf bacterial spot | 1.0000 | 0.8333 | 0.9091 | 12 |
| tomato leaf early blight | 0.7000 | 0.5833 | 0.6364 | 12 |
| tomato leaf healthy | 0.7143 | 0.8333 | 0.7692 | 12 |
| tomato leaf late blight | 0.5714 | 0.6667 | 0.6154 | 12 |
| tomato leaf mold | 0.8462 | 0.9167 | 0.8800 | 12 |
| tomato leaf powdery mildew | 0.7333 | 1.0000 | 0.8462 | 11 |
| tomato septoria leaf spot | 0.8750 | 0.5833 | 0.7000 | 12 |

```
                accuracy                           0.8201       189
               macro avg     0.8338    0.8215    0.8202       189
            weighted avg     0.8339    0.8201    0.8197       189

Fixed and running! Outputs in /kaggle/working/.
```