In [2]:
```
import tensorflow as tf
dataset_dir = "C:\Plant_leave_diseases_dataset_without_augmentation"

batch_size = 32
img_size = (224, 224)

dataset = tf.keras.utils.image_dataset_from_directory(
    dataset_dir,
    labels='inferred',
    label_mode='categorical',
    color_mode='rgb',
    batch_size=batch_size,
    image_size=img_size,
    shuffle=True
)

for images, labels in dataset.take(1):
    print("Image batch shape:", images.shape)
    print("Label batch shape:", labels.shape)
```

```
Found 24164 files belonging to 17 classes.
Image batch shape: (32, 224, 224, 3)
Label batch shape: (32, 17)
```

In [3]:
```
class_names = dataset.class_names
print("Class names:", class_names)
```

```
Class names: ['Corn___Cercospora_leaf_spot Gray_leaf_spot', 'Corn___Common_rust', 'Cor
n___Northern_Leaf_Blight', 'Corn___healthy', 'Potato___Early_blight', 'Potato___Late_b
light', 'Potato___healthy', 'Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomat
o___Late_blight', 'Tomato___Leaf_Mold', 'Tomato___Septoria_leaf_spot', 'Tomato___Spide
r_mites Two-spotted_spider_mite', 'Tomato___Target_Spot', 'Tomato___Tomato_Yellow_Leaf
_Curl_Virus', 'Tomato___Tomato_mosaic_virus', 'Tomato___healthy']
```

In [4]:
```
dataset_size = tf.data.experimental.cardinality(dataset).numpy()


train_size = int(0.8 * dataset_size)
val_size = int(0.1 * dataset_size)
test_size = dataset_size - train_size - val_size


train_dataset = dataset.take(train_size)
remaining_dataset = dataset.skip(train_size)

val_dataset = remaining_dataset.take(val_size)
test_dataset = remaining_dataset.skip(val_size)


train_dataset = train_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
val_dataset = val_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)


print(f"Total dataset size: {dataset_size}")
print(f"Train dataset size: {train_size}")
print(f"Validation dataset size: {val_size}")
print(f"Test dataset size: {test_size}")
```

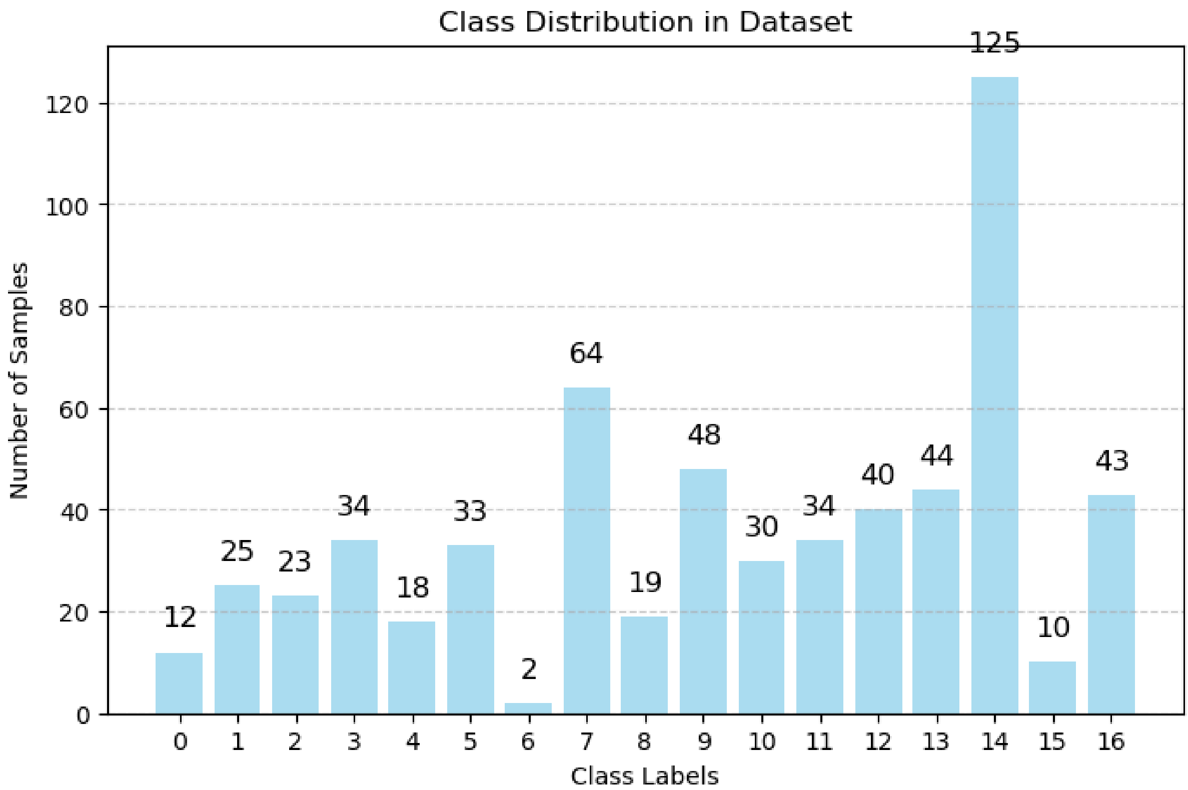
```
Total dataset size: 756
Train dataset size: 604
Validation dataset size: 75
Test dataset size: 77
```

class balancing

In [5]:
```
import matplotlib.pyplot as plt
import numpy as np
from collections import Counter

def get_class_distribution(dataset):
    labels = []
    for _, label in dataset:
        label = label.numpy()
        if label.ndim > 0:
            label = label.argmax()
        labels.append(label)

    class_counts = Counter(labels)
    return class_counts


class_counts = get_class_distribution(train_dataset)


print("Class Distribution:", class_counts)
```

```
Class Distribution: Counter({14: 125, 7: 64, 9: 48, 13: 44, 16: 43, 12: 40, 11: 34, 3:
34, 5: 33, 10: 30, 1: 25, 2: 23, 8: 19, 4: 18, 0: 12, 15: 10, 6: 2})
```

In [6]:
```
def plot_class_distribution(class_counts):
    classes = list(class_counts.keys())
    counts = list(class_counts.values())

    plt.figure(figsize=(8, 5))
    plt.bar(classes, counts, color='skyblue', alpha=0.7)
    plt.xlabel("Class Labels")
    plt.ylabel("Number of Samples")
    plt.title("Class Distribution in Dataset")
    plt.xticks(classes)
    plt.grid(axis="y", linestyle="--", alpha=0.6)

    for i, count in enumerate(counts):
        plt.text(classes[i], count + 5, str(count), ha="center", fontsize=12)

    plt.show()

plot_class_distribution(class_counts)
```



In [7]:
```
from sklearn.utils.class_weight import compute_class_weight
import numpy as np

all_labels = []
for _, labels in train_dataset:
    all_labels.extend(np.argmax(labels.numpy(), axis=1))

all_labels = np.array(all_labels)

class_weights_values = compute_class_weight(
    class_weight="balanced",
    classes=np.unique(all_labels),
    y=all_labels
)

class_weights = {i: class_weights_values[i] for i in np.unique(all_labels)}

print("Computed Class Weights:", class_weights)
```

```
Computed Class Weights: {0: 2.7730272596843615, 1: 1.2031123560535326, 2: 1.4319158393
836124, 3: 1.2107999749420535, 4: 1.3848248190871963, 5: 1.4557505460570912, 6: 9.1688
80455407969, 7: 0.6617818256522632, 8: 1.43372153401083, 9: 0.7470047151580738, 10: 1.
4727217311795184, 11: 0.7895424836601307, 12: 0.8317053229484918, 13: 1.02612019537056
7, 14: 0.26651223077135217, 15: 3.8024788510722014, 16: 0.9250945292681759}
```

data augmentation

In [8]:
```
from tensorflow.keras.layers import RandomFlip, RandomRotation, RandomZoom

data_augmentation = tf.keras.Sequential([
    RandomFlip("horizontal"),
    RandomRotation(0.2),
    RandomZoom(0.2),
])
```

In [9]:
```
train_dataset = train_dataset.map(lambda x, y: (data_augmentation(x, training=True), y)
```

vgg19

In [13]:
```
from tensorflow.keras.applications import VGG19
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import AdamW
import tensorflow as tf
```

In [14]:
```
base_model_v = VGG19(weights="imagenet", include_top=False, input_shape=(224, 224, 3))

base_model_v.trainable = False

x = GlobalAveragePooling2D()(base_model_v.output)
x = Dense(256, activation="relu", kernel_regularizer=l2(0.001))(x)
x = Dropout(0.5)(x)
x = Dense(128, activation="relu", kernel_regularizer=l2(0.001))(x)
x = Dropout(0.3)(x)
output = Dense(17, activation="softmax")(x)

model_v = Model(inputs=base_model_v.input, outputs=output)
```

In [17]:
```
model_v.compile(optimizer=AdamW(learning_rate=1e-3),
              loss="categorical_crossentropy",
              metrics=["accuracy"])

early_stopping = tf.keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, resto
lr_scheduler = tf.keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=3, fac
```

In [12]:
```
history = model_v.fit(train_dataset,
                    validation_data=val_dataset,
                    epochs=30,
                    class_weight=class_weights,
                    callbacks=[early_stopping, lr_scheduler])
```

```
Epoch 1/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 8152s 13s/step - accuracy: 0.3474 - loss: 2.8529 - val_ac
curacy: 0.7504 - val_loss: 1.1389 - learning_rate: 0.0010
Epoch 2/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7897s 13s/step - accuracy: 0.6543 - loss: 1.3760 - val_ac
curacy: 0.7887 - val_loss: 0.9182 - learning_rate: 0.0010
Epoch 3/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7761s 13s/step - accuracy: 0.7125 - loss: 1.1573 - val_ac
curacy: 0.8058 - val_loss: 0.8405 - learning_rate: 0.0010
Epoch 4/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7719s 13s/step - accuracy: 0.7502 - loss: 1.0203 - val_ac
curacy: 0.8317 - val_loss: 0.7416 - learning_rate: 0.0010
Epoch 5/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7731s 13s/step - accuracy: 0.7517 - loss: 0.9646 - val_ac
curacy: 0.8392 - val_loss: 0.6908 - learning_rate: 0.0010
Epoch 6/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7759s 13s/step - accuracy: 0.7721 - loss: 0.8976 - val_ac
curacy: 0.8554 - val_loss: 0.6493 - learning_rate: 0.0010
Epoch 7/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7753s 13s/step - accuracy: 0.7799 - loss: 0.8616 - val_ac
curacy: 0.8433 - val_loss: 0.6706 - learning_rate: 0.0010
Epoch 8/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7712s 13s/step - accuracy: 0.7812 - loss: 0.8573 - val_ac
curacy: 0.8438 - val_loss: 0.6379 - learning_rate: 0.0010
Epoch 9/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7735s 13s/step - accuracy: 0.7931 - loss: 0.8310 - val_ac
curacy: 0.8496 - val_loss: 0.6541 - learning_rate: 0.0010
Epoch 10/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7739s 13s/step - accuracy: 0.7929 - loss: 0.8222 - val_ac
curacy: 0.8633 - val_loss: 0.6212 - learning_rate: 0.0010
Epoch 11/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7736s 13s/step - accuracy: 0.7891 - loss: 0.8209 - val_ac
curacy: 0.8375 - val_loss: 0.7070 - learning_rate: 0.0010
Epoch 12/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7852s 13s/step - accuracy: 0.7979 - loss: 0.8047 - val_ac
curacy: 0.8313 - val_loss: 0.7080 - learning_rate: 0.0010
Epoch 13/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 8135s 13s/step - accuracy: 0.7980 - loss: 0.8122 - val_ac
curacy: 0.8617 - val_loss: 0.6148 - learning_rate: 0.0010
Epoch 14/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7798s 13s/step - accuracy: 0.8007 - loss: 0.7800 - val_ac
curacy: 0.8763 - val_loss: 0.5705 - learning_rate: 0.0010
Epoch 15/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7809s 13s/step - accuracy: 0.8021 - loss: 0.7916 - val_ac
curacy: 0.8554 - val_loss: 0.6347 - learning_rate: 0.0010
Epoch 16/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7795s 13s/step - accuracy: 0.8056 - loss: 0.7940 - val_ac
curacy: 0.8346 - val_loss: 0.6774 - learning_rate: 0.0010
Epoch 17/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7771s 13s/step - accuracy: 0.7972 - loss: 0.8014 - val_ac
curacy: 0.8708 - val_loss: 0.5781 - learning_rate: 0.0010
Epoch 18/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7768s 13s/step - accuracy: 0.8269 - loss: 0.7143 - val_ac
curacy: 0.8746 - val_loss: 0.5686 - learning_rate: 5.0000e-04
Epoch 19/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7768s 13s/step - accuracy: 0.8345 - loss: 0.6792 - val_ac
curacy: 0.8642 - val_loss: 0.5842 - learning_rate: 5.0000e-04
Epoch 20/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7791s 13s/step - accuracy: 0.8317 - loss: 0.6709 - val_ac
curacy: 0.8767 - val_loss: 0.5374 - learning_rate: 5.0000e-04
Epoch 21/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7673s 13s/step - accuracy: 0.8368 - loss: 0.6505 - val_ac
curacy: 0.8896 - val_loss: 0.4952 - learning_rate: 5.0000e-04
Epoch 22/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7632s 13s/step - accuracy: 0.8344 - loss: 0.6499 - val_ac
curacy: 0.8679 - val_loss: 0.5378 - learning_rate: 5.0000e-04
Epoch 23/30
```

```
604/604 ━━━━━━━━━━━━━━━━━━━━ 7820s 13s/step - accuracy: 0.8424 - loss: 0.6395 - val_ac
curacy: 0.8825 - val_loss: 0.5144 - learning_rate: 5.0000e-04
Epoch 24/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7809s 13s/step - accuracy: 0.8407 - loss: 0.6341 - val_ac
curacy: 0.8792 - val_loss: 0.5245 - learning_rate: 5.0000e-04
Epoch 25/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7831s 13s/step - accuracy: 0.8484 - loss: 0.5926 - val_ac
curacy: 0.8883 - val_loss: 0.4891 - learning_rate: 2.5000e-04
Epoch 26/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 8868s 15s/step - accuracy: 0.8526 - loss: 0.5703 - val_ac
curacy: 0.8858 - val_loss: 0.4797 - learning_rate: 2.5000e-04
Epoch 27/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7691s 13s/step - accuracy: 0.8593 - loss: 0.5734 - val_ac
curacy: 0.8850 - val_loss: 0.4910 - learning_rate: 2.5000e-04
Epoch 28/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7701s 13s/step - accuracy: 0.8593 - loss: 0.5624 - val_ac
curacy: 0.8933 - val_loss: 0.4591 - learning_rate: 2.5000e-04
Epoch 29/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7710s 13s/step - accuracy: 0.8593 - loss: 0.5537 - val_ac
curacy: 0.8967 - val_loss: 0.4548 - learning_rate: 2.5000e-04
Epoch 30/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 7757s 13s/step - accuracy: 0.8569 - loss: 0.5565 - val_ac
curacy: 0.8988 - val_loss: 0.4529 - learning_rate: 2.5000e-04
```

In [21]:

```
# Unfreeze Last 80 Layers for Fine-Tuning
for layer in base_model_v.layers[-80:]:
    layer.trainable = True

# Compile again with a lower learning rate
model_v.compile(optimizer=AdamW(learning_rate=1e-5),
              loss="categorical_crossentropy",
              metrics=["accuracy"])
```

In [22]:
```
fine_tune_history = model_v.fit(train_dataset,
                                validation_data=val_dataset,
                                epochs=30,
                                class_weight=class_weights,
                                callbacks=[early_stopping, lr_scheduler])
```

```
Epoch 1/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37441s 62s/step - accuracy: 0.8809 - loss: 0.5124 - val_a
ccuracy: 0.9133 - val_loss: 0.3677 - learning_rate: 1.0000e-05
Epoch 2/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37324s 62s/step - accuracy: 0.9098 - loss: 0.4103 - val_a
ccuracy: 0.9429 - val_loss: 0.3114 - learning_rate: 1.0000e-05
Epoch 3/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37414s 62s/step - accuracy: 0.9303 - loss: 0.3437 - val_a
ccuracy: 0.9371 - val_loss: 0.3127 - learning_rate: 1.0000e-05
Epoch 4/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37556s 62s/step - accuracy: 0.9411 - loss: 0.3054 - val_a
ccuracy: 0.9625 - val_loss: 0.2394 - learning_rate: 1.0000e-05
Epoch 5/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 36889s 61s/step - accuracy: 0.9458 - loss: 0.2999 - val_a
ccuracy: 0.9442 - val_loss: 0.2923 - learning_rate: 1.0000e-05
Epoch 6/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 36366s 60s/step - accuracy: 0.9531 - loss: 0.2786 - val_a
ccuracy: 0.9400 - val_loss: 0.3015 - learning_rate: 1.0000e-05
Epoch 7/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 36180s 60s/step - accuracy: 0.9576 - loss: 0.2709 - val_a
ccuracy: 0.9613 - val_loss: 0.2429 - learning_rate: 1.0000e-05
Epoch 8/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 36988s 61s/step - accuracy: 0.9660 - loss: 0.2348 - val_a
ccuracy: 0.9683 - val_loss: 0.2249 - learning_rate: 5.0000e-06
Epoch 9/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37462s 62s/step - accuracy: 0.9716 - loss: 0.2165 - val_a
ccuracy: 0.9667 - val_loss: 0.2261 - learning_rate: 5.0000e-06
Epoch 10/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37451s 62s/step - accuracy: 0.9712 - loss: 0.2140 - val_a
ccuracy: 0.9588 - val_loss: 0.2621 - learning_rate: 5.0000e-06
Epoch 11/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37444s 62s/step - accuracy: 0.9719 - loss: 0.2195 - val_a
ccuracy: 0.9646 - val_loss: 0.2421 - learning_rate: 5.0000e-06
Epoch 12/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37423s 62s/step - accuracy: 0.9808 - loss: 0.1861 - val_a
ccuracy: 0.9563 - val_loss: 0.2818 - learning_rate: 2.5000e-06
Epoch 13/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37414s 62s/step - accuracy: 0.9812 - loss: 0.1901 - val_a
ccuracy: 0.9717 - val_loss: 0.2155 - learning_rate: 2.5000e-06
Epoch 14/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37425s 62s/step - accuracy: 0.9828 - loss: 0.1756 - val_a
ccuracy: 0.9675 - val_loss: 0.2487 - learning_rate: 2.5000e-06
Epoch 15/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37422s 62s/step - accuracy: 0.9815 - loss: 0.1773 - val_a
ccuracy: 0.9667 - val_loss: 0.2384 - learning_rate: 2.5000e-06
Epoch 16/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37423s 62s/step - accuracy: 0.9817 - loss: 0.1797 - val_a
ccuracy: 0.9700 - val_loss: 0.2195 - learning_rate: 2.5000e-06
Epoch 17/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37407s 62s/step - accuracy: 0.9847 - loss: 0.1697 - val_a
ccuracy: 0.9754 - val_loss: 0.2090 - learning_rate: 1.2500e-06
Epoch 18/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37431s 62s/step - accuracy: 0.9887 - loss: 0.1591 - val_a
ccuracy: 0.9842 - val_loss: 0.1801 - learning_rate: 1.2500e-06
Epoch 19/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 41204s 68s/step - accuracy: 0.9873 - loss: 0.1653 - val_a
ccuracy: 0.9787 - val_loss: 0.1946 - learning_rate: 1.2500e-06
Epoch 20/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37535s 62s/step - accuracy: 0.9877 - loss: 0.1631 - val_a
ccuracy: 0.9771 - val_loss: 0.2084 - learning_rate: 1.2500e-06
Epoch 21/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37473s 62s/step - accuracy: 0.9874 - loss: 0.1638 - val_a
ccuracy: 0.9729 - val_loss: 0.2279 - learning_rate: 1.2500e-06
Epoch 22/30
604/604 ━━━━━━━━━━━━━━━━━━━━ 37429s 62s/step - accuracy: 0.9875 - loss: 0.1630 - val_a
ccuracy: 0.9750 - val_loss: 0.2127 - learning_rate: 1.0000e-06
Epoch 23/30
```

```
604/604 ━━━━━━━━━━━━━━━━━━━━ 37456s 62s/step - accuracy: 0.9890 - loss: 0.1561 - val_accuracy: 0.9775 - val_loss: 0.2082 - learning_rate: 1.0000e-06
```

In [15]:
```
model_v.save("vgg19.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

In [17]:
```
model_v.save('my_model.keras')
```

In [20]:
```
from tensorflow.keras.models import load_model

# Load the model
model_v = load_model("my_model.keras")

# Check model summary
model_v.summary()
```

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 256)	131,328
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 17)	2,193

Total params: 40,381,604 (154.04 MB)

Trainable params: 20,190,801 (77.02 MB)

Non-trainable params: 0 (0.00 B)

```
Optimizer params: 20,190,803 (77.02 MB)
```

In [24]:
```
model_v.save("vgg19.keras")
```

In [25]:
```
test_loss, test_acc = model_v.evaluate(test_dataset)
print(f"Test Accuracy: {test_acc:.4f}")
```

```
77/77 ━━━━━━━━━━━━━━━━━━━━ 909s 11s/step - accuracy: 0.9769 - loss: 0.1941
Test Accuracy: 0.9811
```

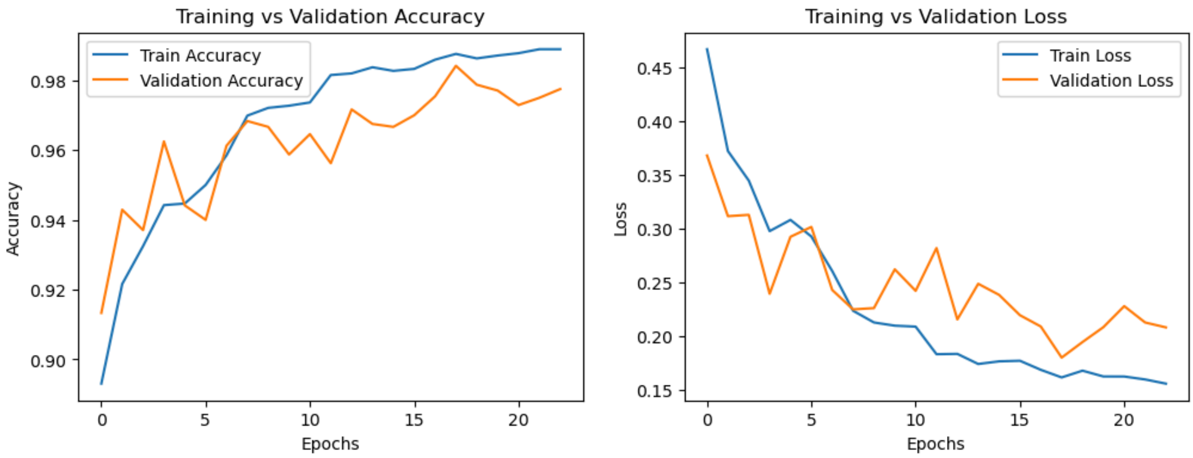
In [26]:
```
train_loss, train_acc = model_v.evaluate(train_dataset)
print(f"Train Accuracy: {train_acc:.4f}")
```

```
604/604 ━━━━━━━━━━━━━━━━━━━━ 6955s 12s/step - accuracy: 0.9954 - loss: 0.1367
Train Accuracy: 0.9960
```

In [27]:
```
val_loss, val_acc = model_v.evaluate(val_dataset)
print(f"Validation Accuracy: {val_acc:.4f}")
```

```
75/75 ━━━━━━━━━━━━━━━━━━━━ 878s 11s/step - accuracy: 0.9849 - loss: 0.1799
Validation Accuracy: 0.9821
```

In [28]:
```python
def plot_metrics(history):
    plt.figure(figsize=(12, 4))

    # Accuracy Plot
    plt.subplot(1, 2, 1)
    plt.plot(history.history["accuracy"], label="Train Accuracy")
    plt.plot(history.history["val_accuracy"], label="Validation Accuracy")
    plt.xlabel("Epochs")
    plt.ylabel("Accuracy")
    plt.title("Training vs Validation Accuracy")
    plt.legend()

    # Loss Plot
    plt.subplot(1, 2, 2)
    plt.plot(history.history["loss"], label="Train Loss")
    plt.plot(history.history["val_loss"], label="Validation Loss")
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.title("Training vs Validation Loss")
    plt.legend()

    plt.show()

plot_metrics(fine_tune_history)
```



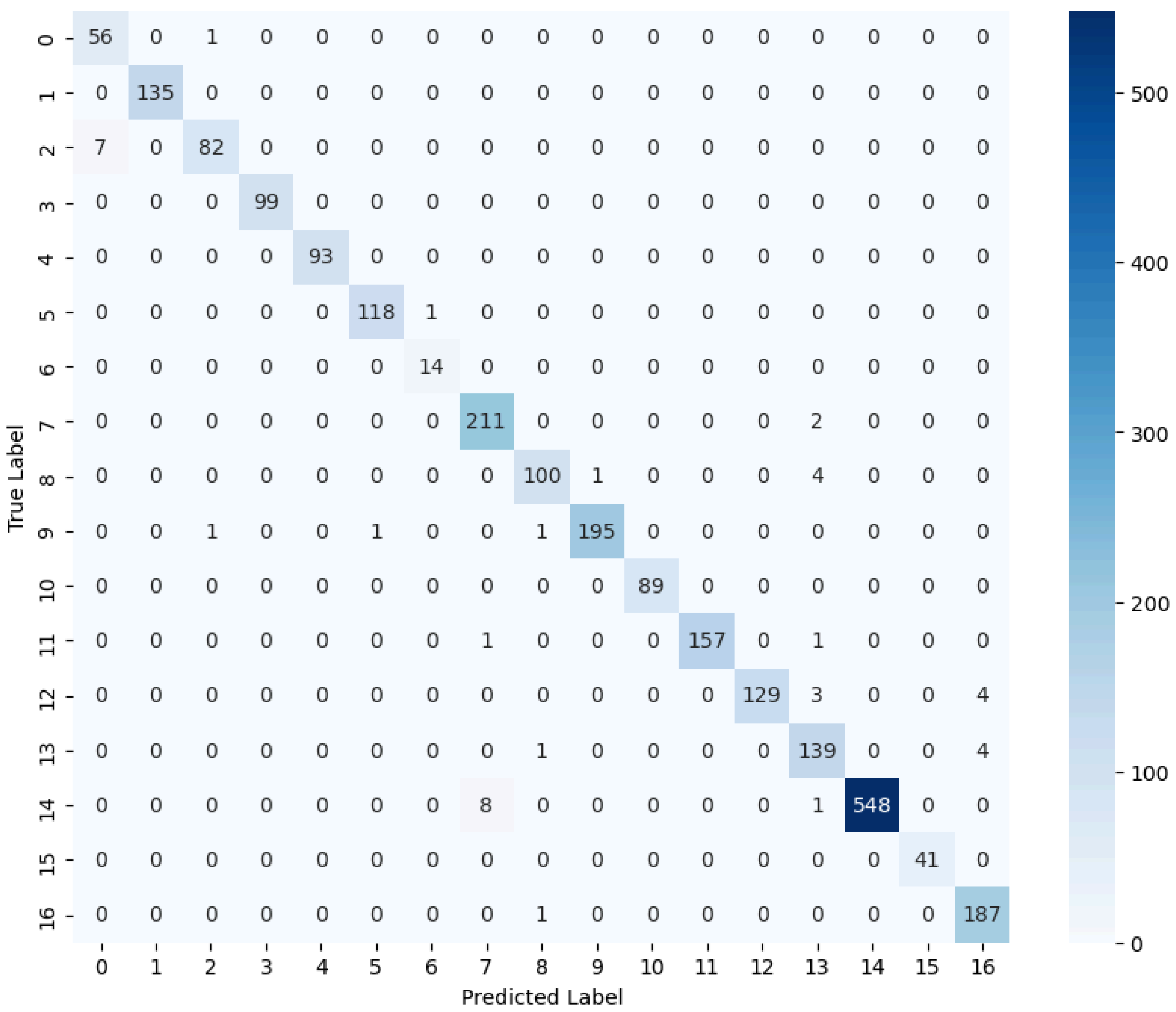
In [29]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report

true_labels = []
pred_labels = []

for images, labels in test_dataset:
    preds = model_v.predict(images)
    pred_classes = np.argmax(preds, axis=1)
    true_classes = np.argmax(labels.numpy(), axis=1)

    true_labels.extend(true_classes)
    pred_labels.extend(pred_classes)

true_labels = np.array(true_labels)
pred_labels = np.array(pred_labels)

cm = confusion_matrix(true_labels, pred_labels)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 12s 12s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 12s 12s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 12s 12s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 12s 12s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 12s 12s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 12s 12s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 12s 12s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 12s 12s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 11s 11s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```

In [30]:

```
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=range(17), yticklabels=r
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix for ResNet50")
plt.show()
```



In [31]:
```
#classification report
class_names = [f"Class {i}" for i in range(17)]
report = classification_report(true_labels, pred_labels, target_names=class_names)
print("Classification Report:\n", report)
```

```
Classification Report:
               precision    recall  f1-score   support

     Class 0       0.89      0.98      0.93        57
     Class 1       1.00      1.00      1.00       135
     Class 2       0.98      0.92      0.95        89
     Class 3       1.00      1.00      1.00        99
     Class 4       1.00      1.00      1.00        93
     Class 5       0.99      0.99      0.99       119
     Class 6       0.93      1.00      0.97        14
     Class 7       0.96      0.99      0.97       213
     Class 8       0.97      0.95      0.96       105
     Class 9       0.99      0.98      0.99       198
    Class 10       1.00      1.00      1.00        89
    Class 11       1.00      0.99      0.99       159
    Class 12       1.00      0.95      0.97       136
    Class 13       0.93      0.97      0.95       144
    Class 14       1.00      0.98      0.99       557
    Class 15       1.00      1.00      1.00        41
    Class 16       0.96      0.99      0.98       188

    accuracy                           0.98      2436
   macro avg       0.98      0.98      0.98      2436
weighted avg       0.98      0.98      0.98      2436
```

In [ ]: