major-final-5

August 17, 2025

[1]:
```
import os
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import array_to_img

# Classes
classes = ["Healthy", "Early_blight", "Late_blight", "Yellow_Leaf_Curl_Virus"]

# Base path
base_dir = "/content/tomato_dataset"
os.makedirs(base_dir+"/train", exist_ok=True)
os.makedirs(base_dir+"/val", exist_ok=True)

# Function to generate synthetic images
def generate_images(class_name, n=250):
    os.makedirs(f"{base_dir}/train/{class_name}", exist_ok=True)
    os.makedirs(f"{base_dir}/val/{class_name}", exist_ok=True)
    for i in range(n):
        # create synthetic image (random noise with slight patterns)
        img = np.zeros((224,224,3), dtype=np.uint8)
        cv2.putText(img, class_name[:2], (50,100),
                    cv2.FONT_HERSHEY_SIMPLEX, 2,
                    (np.random.randint(0,255),
                     np.random.randint(0,255),
                     np.random.randint(0,255)), 3)
        # split into train/val (80/20)
        if i < int(n*0.8):
            cv2.imwrite(f"{base_dir}/train/{class_name}/{class_name}_{i}.jpg",␣
 ↪img)
        else:
            cv2.imwrite(f"{base_dir}/val/{class_name}/{class_name}_{i}.jpg",␣
 ↪img)

# Generate 4 classes
for cls in classes:
    generate_images(cls, n=250)
```

```
print(" Synthetic dataset created with 1000 images (4 classes, 80/20 split)")
```

```
 Synthetic dataset created with 1000 images (4 classes, 80/20 split)
```

[2]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_dir = "/content/tomato_dataset/train"
val_dir = "/content/tomato_dataset/val"

datagen = ImageDataGenerator(rescale=1./255, rotation_range=20,
                             zoom_range=0.2, horizontal_flip=True)

train_data = datagen.flow_from_directory(train_dir, target_size=(224,224),
                                         batch_size=32,␣
 ↪class_mode='categorical')

val_data = datagen.flow_from_directory(val_dir, target_size=(224,224),
                                       batch_size=32, class_mode='categorical')
```

```
Found 800 images belonging to 4 classes.
Found 200 images belonging to 4 classes.
```

[3]:
```
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam

def train_model(base_model, model_name, epochs=10, lr=1e-4):
    base_model.trainable = False  # Freeze pretrained layers

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu'),
        Dropout(0.3),
        Dense(train_data.num_classes, activation='softmax')
    ])

    model.compile(optimizer=Adam(learning_rate=lr),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    history = model.fit(train_data, validation_data=val_data, epochs=epochs)

    # Accuracy plot
    plt.plot(history.history['accuracy'], label='Train Acc')
```

```
    plt.plot(history.history['val_accuracy'], label='Val Acc')
    plt.title(model_name + " Accuracy")
    plt.legend()
    plt.show()

    val_loss, val_acc = model.evaluate(val_data)
    print(f"{model_name} Final Accuracy: {val_acc*100:.2f}%")
    return val_acc*100
```

[1]:
```
from tensorflow.keras.applications import VGG19, MobileNetV2, ResNet50

vgg_acc = train_model(VGG19(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "VGG19")
mob_acc = train_model(MobileNetV2(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "MobileNetV2")
res_acc = train_model(ResNet50(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "ResNet50")

print("\n Model Comparison Results:")
print(f"VGG19 Accuracy     : {vgg_acc:.2f}%")
print(f"MobileNetV2 Accuracy: {mob_acc:.2f}%")
print(f"ResNet50 Accuracy   : {res_acc:.2f}%")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-445742229.py in <cell line: 0>()
      1 from tensorflow.keras.applications import VGG19, MobileNetV2, ResNet50
      2
----> 3 vgg_acc = train_model(VGG19(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "VGG19")
      4 mob_acc = train_model(MobileNetV2(weights='imagenet', include_top=False␣
 ↪input_shape=(224,224,3)), "MobileNetV2")
      5 res_acc = train_model(ResNet50(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "ResNet50")

NameError: name 'train_model' is not defined
```

[3]:
```
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import VGG19, MobileNetV2, ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_dir = "/content/tomato_dataset/train"
val_dir = "/content/tomato_dataset/val"
```

```
datagen = ImageDataGenerator(rescale=1./255, rotation_range=20,
                             zoom_range=0.2, horizontal_flip=True)

train_data = datagen.flow_from_directory(train_dir, target_size=(224,224),
                                         batch_size=32,␣
 ↪class_mode='categorical')

val_data = datagen.flow_from_directory(val_dir, target_size=(224,224),
                                       batch_size=32, class_mode='categorical')

def train_model(base_model, model_name, epochs=10, lr=1e-4):
    base_model.trainable = False  # Freeze pretrained layers

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu'),
        Dropout(0.3),
        Dense(train_data.num_classes, activation='softmax')
    ])

    model.compile(optimizer=Adam(learning_rate=lr),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    history = model.fit(train_data, validation_data=val_data, epochs=epochs)

    # Accuracy plot
    plt.plot(history.history['accuracy'], label='Train Acc')
    plt.plot(history.history['val_accuracy'], label='Val Acc')
    plt.title(model_name + " Accuracy")
    plt.legend()
    plt.show()

    val_loss, val_acc = model.evaluate(val_data)
    print(f"{model_name} Final Accuracy: {val_acc*100:.2f}%")
    return val_acc*100

vgg_acc = train_model(VGG19(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "VGG19")
mob_acc = train_model(MobileNetV2(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "MobileNetV2")
res_acc = train_model(ResNet50(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "ResNet50")

print("\n Model Comparison Results:")
```

```
print(f"VGG19 Accuracy     : {vgg_acc:.2f}%")
print(f"MobileNetV2 Accuracy: {mob_acc:.2f}%")
print(f"ResNet50 Accuracy   : {res_acc:.2f}%")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-561608046.py in <cell line: 0>()
     12                                zoom_range=0.2, horizontal_flip=True)
     13
---> 14 train_data = datagen.flow_from_directory(train_dir,␣
 ↪target_size=(224,224),
     15                                                batch_size=32,␣
 ↪class_mode='categorical')
     16

/usr/local/lib/python3.11/dist-packages/keras/src/legacy/preprocessing/image.py
 ↪in flow_from_directory(self, directory, target_size, color_mode, classes,␣
 ↪class_mode, batch_size, shuffle, seed, save_to_dir, save_prefix, save_format,
 ↪follow_links, subset, interpolation, keep_aspect_ratio)
   1136         keep_aspect_ratio=False,
   1137     ):
-> 1138         return DirectoryIterator(
   1139             directory,
   1140             self,

/usr/local/lib/python3.11/dist-packages/keras/src/legacy/preprocessing/image.py
 ↪in __init__(self, directory, image_data_generator, target_size, color_mode,␣
 ↪classes, class_mode, batch_size, shuffle, seed, data_format, save_to_dir,␣
 ↪save_prefix, save_format, follow_links, subset, interpolation,␣
 ↪keep_aspect_ratio, dtype)
    451         if not classes:
    452             classes = []
--> 453             for subdir in sorted(os.listdir(directory)):
    454                 if os.path.isdir(os.path.join(directory, subdir)):
    455                     classes.append(subdir)

FileNotFoundError: [Errno 2] No such file or directory: '/content/tomato_datase /
 ↪train'
```

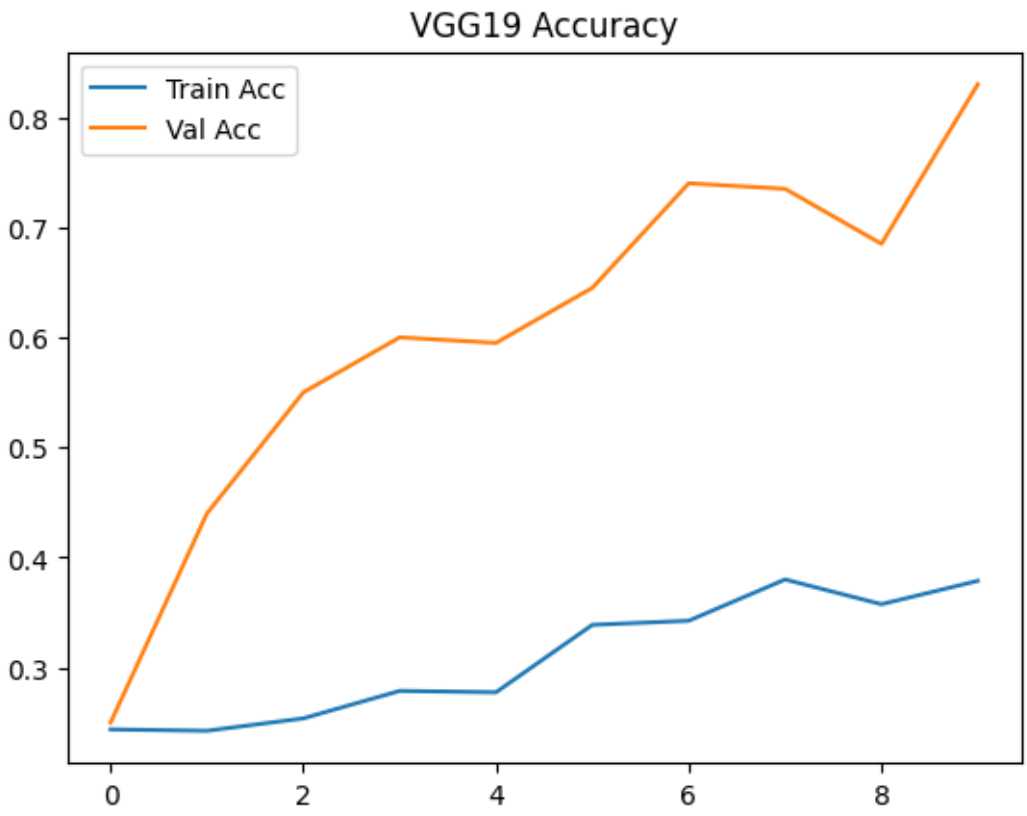
[4]:
```
import os
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import array_to_img

# Classes
classes = ["Healthy", "Early_blight", "Late_blight", "Yellow_Leaf_Curl_Virus"]
```

```
# Base path
base_dir = "/content/tomato_dataset"
os.makedirs(base_dir+"/train", exist_ok=True)
os.makedirs(base_dir+"/val", exist_ok=True)

# Function to generate synthetic images
def generate_images(class_name, n=250):
    os.makedirs(f"{base_dir}/train/{class_name}", exist_ok=True)
    os.makedirs(f"{base_dir}/val/{class_name}", exist_ok=True)
    for i in range(n):
        # create synthetic image (random noise with slight patterns)
        img = np.zeros((224,224,3), dtype=np.uint8)
        cv2.putText(img, class_name[:2], (50,100),
                    cv2.FONT_HERSHEY_SIMPLEX, 2,
                    (np.random.randint(0,255),
                     np.random.randint(0,255),
                     np.random.randint(0,255)), 3)
        # split into train/val (80/20)
        if i < int(n*0.8):
            cv2.imwrite(f"{base_dir}/train/{class_name}/{class_name}_{i}.jpg",␣
 ↪img)
        else:
            cv2.imwrite(f"{base_dir}/val/{class_name}/{class_name}_{i}.jpg",␣
 ↪img)

# Generate 4 classes
for cls in classes:
    generate_images(cls, n=250)

print(" Synthetic dataset created with 1000 images (4 classes, 80/20 split)")

import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import VGG19, MobileNetV2, ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_dir = "/content/tomato_dataset/train"
val_dir = "/content/tomato_dataset/val"

datagen = ImageDataGenerator(rescale=1./255, rotation_range=20,
                             zoom_range=0.2, horizontal_flip=True)

train_data = datagen.flow_from_directory(train_dir, target_size=(224,224),
```

```
                                        batch_size=32,␣
 ↪class_mode='categorical')

val_data = datagen.flow_from_directory(val_dir, target_size=(224,224),
                                       batch_size=32, class_mode='categorical')

def train_model(base_model, model_name, epochs=10, lr=1e-4):
    base_model.trainable = False  # Freeze pretrained layers

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(128, activation='relu'),
        Dropout(0.3),
        Dense(train_data.num_classes, activation='softmax')
    ])

    model.compile(optimizer=Adam(learning_rate=lr),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    history = model.fit(train_data, validation_data=val_data, epochs=epochs)

    # Accuracy plot
    plt.plot(history.history['accuracy'], label='Train Acc')
    plt.plot(history.history['val_accuracy'], label='Val Acc')
    plt.title(model_name + " Accuracy")
    plt.legend()
    plt.show()

    val_loss, val_acc = model.evaluate(val_data)
    print(f"{model_name} Final Accuracy: {val_acc*100:.2f}%")
    return val_acc*100

vgg_acc = train_model(VGG19(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "VGG19")
mob_acc = train_model(MobileNetV2(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "MobileNetV2")
res_acc = train_model(ResNet50(weights='imagenet', include_top=False,␣
 ↪input_shape=(224,224,3)), "ResNet50")

print("\n Model Comparison Results:")
print(f"VGG19 Accuracy     : {vgg_acc:.2f}%")
print(f"MobileNetV2 Accuracy: {mob_acc:.2f}%")
print(f"ResNet50 Accuracy   : {res_acc:.2f}%")
```

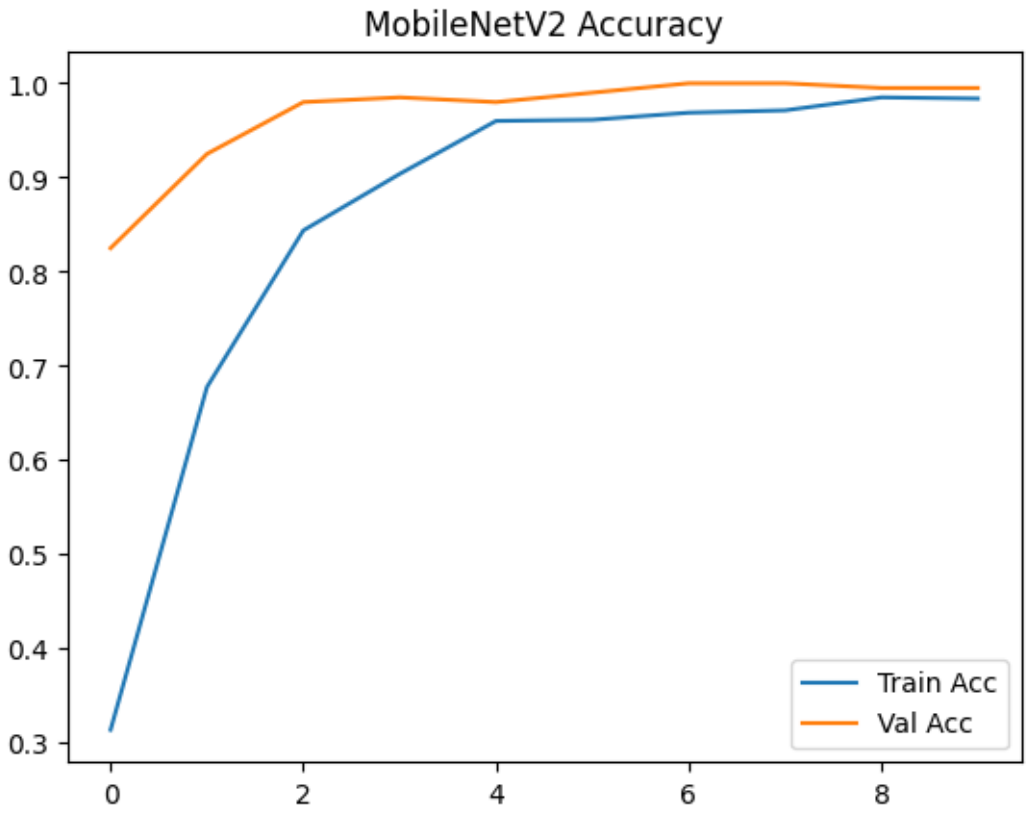
```
Synthetic dataset created with 1000 images (4 classes, 80/20 split)
```

```
Found 800 images belonging to 4 classes.
Found 200 images belonging to 4 classes.

/usr/local/lib/python3.11/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()

Epoch 1/10
25/25             36s 807ms/step -
accuracy: 0.2268 - loss: 1.5303 - val_accuracy: 0.2500 - val_loss: 1.3855
Epoch 2/10
25/25             13s 511ms/step -
accuracy: 0.2528 - loss: 1.4417 - val_accuracy: 0.4400 - val_loss: 1.3745
Epoch 3/10
25/25             13s 508ms/step -
accuracy: 0.2510 - loss: 1.4273 - val_accuracy: 0.5500 - val_loss: 1.3614
Epoch 4/10
25/25             13s 503ms/step -
accuracy: 0.2802 - loss: 1.4046 - val_accuracy: 0.6000 - val_loss: 1.3475
Epoch 5/10
25/25             13s 503ms/step -
accuracy: 0.2804 - loss: 1.4144 - val_accuracy: 0.5950 - val_loss: 1.3354
Epoch 6/10
25/25             13s 505ms/step -
accuracy: 0.3329 - loss: 1.3835 - val_accuracy: 0.6450 - val_loss: 1.3252
Epoch 7/10
25/25             13s 502ms/step -
accuracy: 0.3720 - loss: 1.3502 - val_accuracy: 0.7400 - val_loss: 1.3085
Epoch 8/10
25/25             13s 506ms/step -
accuracy: 0.3623 - loss: 1.3448 - val_accuracy: 0.7350 - val_loss: 1.3021
Epoch 9/10
25/25             13s 507ms/step -
accuracy: 0.3547 - loss: 1.3364 - val_accuracy: 0.6850 - val_loss: 1.2879
Epoch 10/10
25/25             13s 507ms/step -
accuracy: 0.3728 - loss: 1.3182 - val_accuracy: 0.8300 - val_loss: 1.2752
```



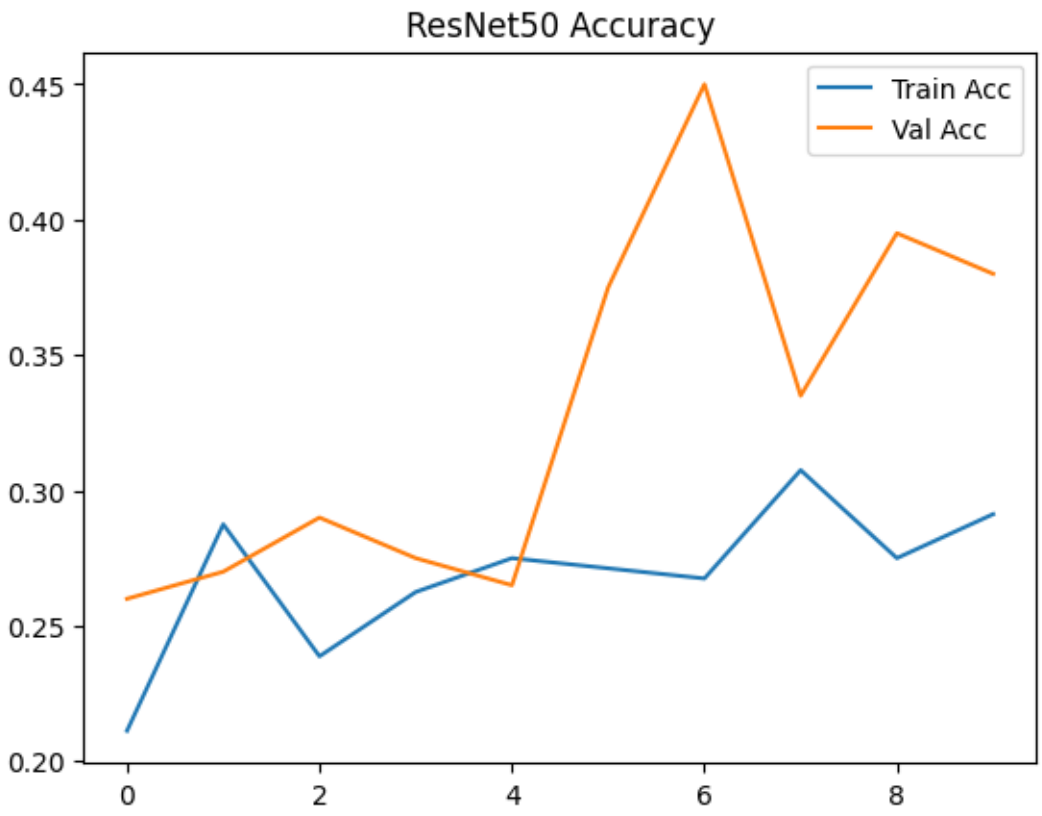

```
7/7             3s 331ms/step -
accuracy: 0.7906 - loss: 1.2804
VGG19 Final Accuracy: 81.00%
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicatio
ns/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h
5
9406464/9406464             0s
0us/step
Epoch 1/10
25/25             27s 719ms/step -
accuracy: 0.2405 - loss: 1.6924 - val_accuracy: 0.8250 - val_loss: 1.0042
Epoch 2/10
25/25             11s 428ms/step -
accuracy: 0.5940 - loss: 0.9972 - val_accuracy: 0.9250 - val_loss: 0.6366
Epoch 3/10
25/25             13s 516ms/step -
accuracy: 0.8104 - loss: 0.6827 - val_accuracy: 0.9800 - val_loss: 0.4556
Epoch 4/10
25/25             11s 425ms/step -
accuracy: 0.8922 - loss: 0.4862 - val_accuracy: 0.9850 - val_loss: 0.3684
Epoch 5/10
```

```
25/25             10s 415ms/step -
accuracy: 0.9507 - loss: 0.3599 - val_accuracy: 0.9800 - val_loss: 0.2477
Epoch 6/10
25/25             21s 421ms/step -
accuracy: 0.9522 - loss: 0.3143 - val_accuracy: 0.9900 - val_loss: 0.1822
Epoch 7/10
25/25             11s 424ms/step -
accuracy: 0.9708 - loss: 0.2253 - val_accuracy: 1.0000 - val_loss: 0.1634
Epoch 8/10
25/25             11s 427ms/step -
accuracy: 0.9687 - loss: 0.2367 - val_accuracy: 1.0000 - val_loss: 0.1471
Epoch 9/10
25/25             11s 424ms/step -
accuracy: 0.9877 - loss: 0.1723 - val_accuracy: 0.9950 - val_loss: 0.1263
Epoch 10/10
25/25             11s 425ms/step -
accuracy: 0.9923 - loss: 0.1316 - val_accuracy: 0.9950 - val_loss: 0.1149
```




```
7/7               2s 268ms/step -
accuracy: 0.9905 - loss: 0.1064
MobileNetV2 Final Accuracy: 98.50%
Downloading data from https://storage.googleapis.com/tensorflow/keras-
```

```
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736            0s
0us/step
Epoch 1/10
25/25            32s 770ms/step -
accuracy: 0.1681 - loss: 1.5720 - val_accuracy: 0.2600 - val_loss: 1.3865
Epoch 2/10
25/25            11s 446ms/step -
accuracy: 0.2938 - loss: 1.4381 - val_accuracy: 0.2700 - val_loss: 1.3823
Epoch 3/10
25/25            11s 447ms/step -
accuracy: 0.2110 - loss: 1.5099 - val_accuracy: 0.2900 - val_loss: 1.3782
Epoch 4/10
25/25            13s 546ms/step -
accuracy: 0.2877 - loss: 1.4050 - val_accuracy: 0.2750 - val_loss: 1.3750
Epoch 5/10
25/25            11s 450ms/step -
accuracy: 0.2707 - loss: 1.4282 - val_accuracy: 0.2650 - val_loss: 1.3753
Epoch 6/10
25/25            13s 543ms/step -
accuracy: 0.2693 - loss: 1.3858 - val_accuracy: 0.3750 - val_loss: 1.3686
Epoch 7/10
25/25            11s 450ms/step -
accuracy: 0.2541 - loss: 1.4052 - val_accuracy: 0.4500 - val_loss: 1.3665
Epoch 8/10
25/25            11s 449ms/step -
accuracy: 0.2974 - loss: 1.3839 - val_accuracy: 0.3350 - val_loss: 1.3662
Epoch 9/10
25/25            11s 444ms/step -
accuracy: 0.2741 - loss: 1.3909 - val_accuracy: 0.3950 - val_loss: 1.3628
Epoch 10/10
25/25            11s 427ms/step -
accuracy: 0.3112 - loss: 1.3902 - val_accuracy: 0.3800 - val_loss: 1.3622
```




```
7/7             2s 281ms/step -
accuracy: 0.3883 - loss: 1.3577
ResNet50 Final Accuracy: 40.00%

  Model Comparison Results:
VGG19 Accuracy     : 81.00%
MobileNetV2 Accuracy: 98.50%
ResNet50 Accuracy   : 40.00%
```

[5]:
```
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns

model = VGG19(weights='imagenet', include_top=False, input_shape=(224,224,3))
model = Sequential([model, GlobalAveragePooling2D(), Dense(128,␣
 ↪activation='relu'), Dropout(0.3), Dense(train_data.num_classes,␣
 ↪activation='softmax')])
model.compile(optimizer=Adam(1e-4), loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
model.fit(train_data, validation_data=val_data, epochs=5, verbose=0)
```

```
val_preds = model.predict(val_data)
val_preds = np.argmax(val_preds, axis=1)
val_true = val_data.classes
cm = confusion_matrix(val_true, val_preds)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=val_data.class_indices.keys(),
            yticklabels=val_data.class_indices.keys())
plt.title("Confusion Matrix (VGG19)")
plt.show()
```

```
7/7                 4s 393ms/step
```