MACHINE LEARNING ASSIGNMENT 08

IMAAD IMRAN HAJWANE
202101132 / 21
LY - 7th SEMESTER
TOPIC: IMPLEMENTATION OF TRANSFER LEARNING ALGORITHM

IMPORTING LIBRARIES

```
import os
import numpy as np
import tensorflow as tf
from keras.preprocessing import image
from keras.applications.vgg16 import VGG16
from keras.applications.resnet50 import ResNet50
from keras.applications.inception_v3 import InceptionV3
# from keras.applications.mobilenet_v2 import MobileNetV2
# from keras.applications.xception import Xception
# from keras.applications.efficientnet import EfficientNetB0
# from keras.applications.vgg19 import VGG19
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense, GlobalAveragePooling2D
from keras.models import Model
from sklearn.metrics import classification_report, accuracy_score,
f1_score, confusion_matrix
import seaborn as sns
from keras.optimizers import Adam
from keras import models as tf_models, layers
import matplotlib.pyplot as plt
from keras.layers import GlobalAveragePooling2D, Dense,
BatchNormalization, Dropout
import warnings
warnings.filterwarnings("ignore")
```

```
WARNING:tensorflow:From c:\Users\iamim\anaconda3\Lib\site-packages\
keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

```
models = [
    'VGG16',
    'ResNet50',
    'InceptionV3'
]

train_data_dir = r'Train_test\train'
validation_data_dir = r'Train_test\validation'
test_data_dir = r'Train_test\test'
```

```
bs = 32
iz = 224

train_generator = tf.keras.preprocessing.image_dataset_from_directory(
    train_data_dir,
    shuffle=True,
    seed=16,
    image_size=(iz, iz),
    batch_size=bs,
)

validation_generator =
tf.keras.preprocessing.image_dataset_from_directory(
    validation_data_dir,
    shuffle=True,
    seed=16,
    image_size=(iz, iz),
    batch_size=bs,
)

test_generator = tf.keras.preprocessing.image_dataset_from_directory(
    test_data_dir,
    shuffle=True,
    seed=16,
    image_size=(iz, iz),
    batch_size=bs,
)
```

```
Found 31500 files belonging to 3 classes.
Found 6750 files belonging to 3 classes.
Found 6750 files belonging to 3 classes.
```

```
class_names = train_generator.class_names
print(class_names)
```

```
['Arborio', 'Basmati', 'Ipsala']
```

VGG16 MODEL

```
model_name = "VGG16"
epochs = 5
desired_steps_per_epoch = 30

num_classes = len(train_generator.class_names)

base_model_vgg16 = VGG16(include_top=False, weights='imagenet',
input_shape=(iz, iz, 3))

base_model_vgg16.trainable = False

model_vgg16 = tf_models.Sequential(
```

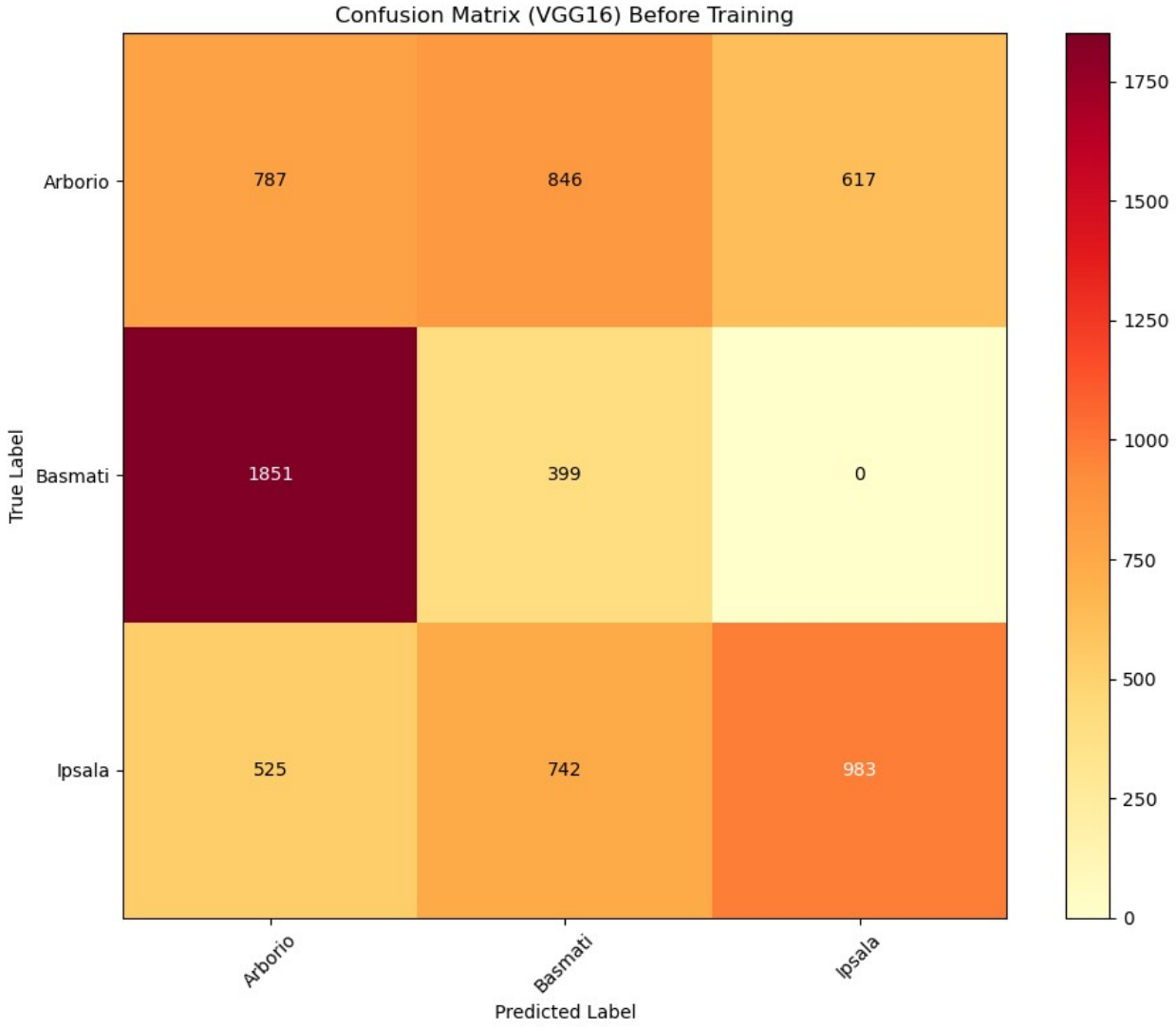
```
[
    base_model_vgg16,
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation='softmax')
])

model_vgg16.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
WARNING:tensorflow:From c:\Users\iamim\anaconda3\Lib\site-packages\
keras\src\backend.py:1398: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From c:\Users\iamim\anaconda3\Lib\site-packages\
keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool
is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From c:\Users\iamim\anaconda3\Lib\site-packages\
keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer is
deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

```
# *** Evaluate accuracy and confusion matrix BEFORE training ***
print("Evaluating before training...")

vgg16_y_true = []
vgg16_y_pred = []

for images, labels in test_generator:
    predictions = model_vgg16.predict(images, verbose=0)
    predicted_labels = tf.argmax(predictions, axis=1)
    vgg16_y_true.extend(labels.numpy())
    vgg16_y_pred.extend(predicted_labels.numpy())

# Confusion matrix and accuracy before training
vgg16_confusion_matrix_before = confusion_matrix(vgg16_y_true,
vgg16_y_pred, labels=range(num_classes))
vgg16_f1_score_before = f1_score(vgg16_y_true, vgg16_y_pred,
average='weighted')
vgg16_classification_report_before =
classification_report(vgg16_y_true, vgg16_y_pred,
target_names=class_names)

print("Confusion Matrix Before Training:\n",
vgg16_confusion_matrix_before)
print("F1 Score Before Training:", vgg16_f1_score_before)
```

```
print("Classification Report Before Training:\n",
vgg16_classification_report_before)

plt.figure(figsize=(10, 8))
plt.imshow(vgg16_confusion_matrix_before, interpolation='nearest',
cmap=plt.cm.YlOrRd)
plt.title('Confusion Matrix (VGG16) Before Training')
plt.colorbar()

tick_marks = np.arange(num_classes)
plt.xticks(tick_marks, class_names, rotation=45)
plt.yticks(tick_marks, class_names)

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.grid(False)

thresh = vgg16_confusion_matrix_before.max() / 2.
for i in range(num_classes):
    for j in range(num_classes):
        plt.text(j, i, format(vgg16_confusion_matrix_before[i, j],
'd'),
                 ha="center", va="center",
                 color="white" if vgg16_confusion_matrix_before[i, j]
> thresh else "black")

plt.tight_layout()
plt.show()
```

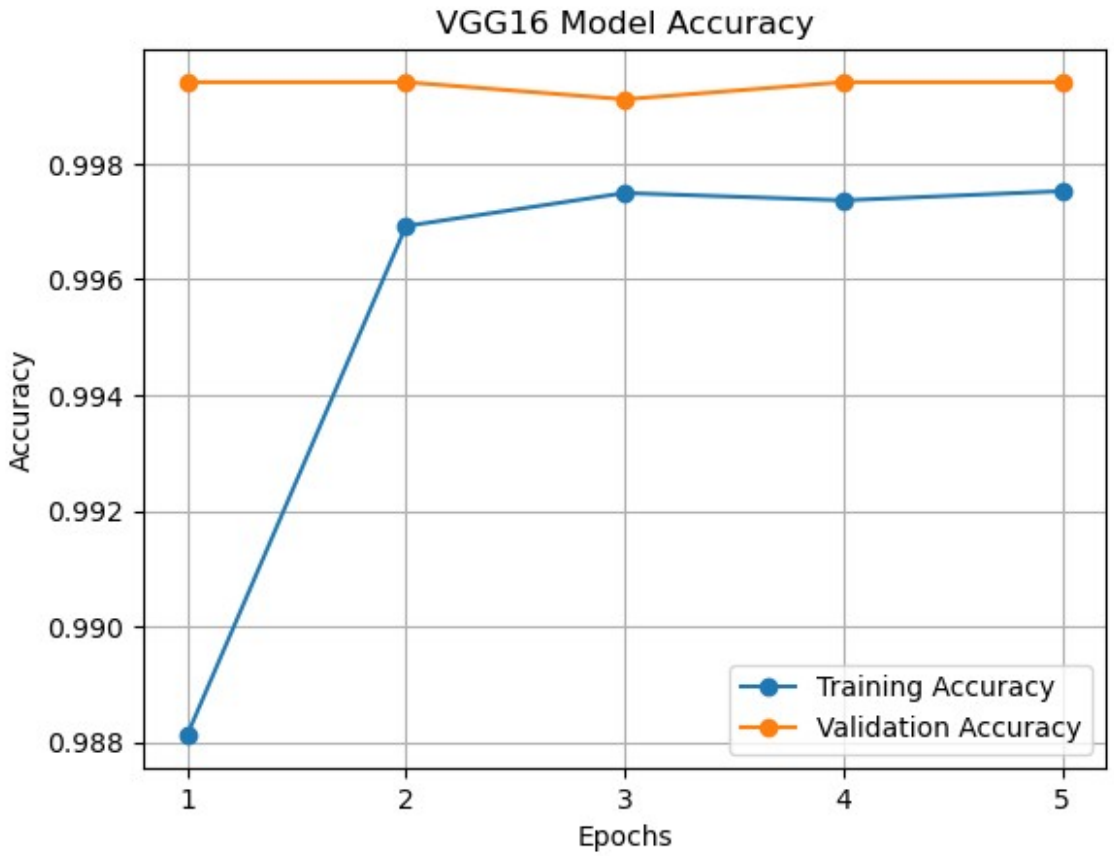
```
Evaluating before training...
Confusion Matrix Before Training:
 [[ 787  846  617]
 [1851  399    0]
 [ 525  742  983]]
F1 Score Before Training: 0.32992386996136486
Classification Report Before Training:
              precision    recall  f1-score   support

     Arborio       0.25      0.35      0.29      2250
     Basmati       0.20      0.18      0.19      2250
      Ipsala       0.61      0.44      0.51      2250

    accuracy                           0.32      6750
   macro avg       0.35      0.32      0.33      6750
weighted avg       0.35      0.32      0.33      6750
```




```
history_vgg16 = model_vgg16.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=epochs,
)
```

```
Epoch 1/5
WARNING:tensorflow:From c:\Users\iamim\anaconda3\Lib\site-packages\
keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue
is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue
instead.

WARNING:tensorflow:From c:\Users\iamim\anaconda3\Lib\site-packages\
keras\src\engine\base_layer_utils.py:384: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.
```

```
985/985 [==============================] - 2961s 3s/step - loss:
0.0434 - accuracy: 0.9881 - val_loss: 0.0021 - val_accuracy: 0.9994
Epoch 2/5
985/985 [==============================] - 4355s 4s/step - loss:
0.0093 - accuracy: 0.9969 - val_loss: 0.0016 - val_accuracy: 0.9994
Epoch 3/5
985/985 [==============================] - 2716s 3s/step - loss:
0.0090 - accuracy: 0.9975 - val_loss: 0.0041 - val_accuracy: 0.9991
Epoch 4/5
985/985 [==============================] - 3196s 3s/step - loss:
0.0090 - accuracy: 0.9974 - val_loss: 0.0023 - val_accuracy: 0.9994
Epoch 5/5
985/985 [==============================] - 3130s 3s/step - loss:
0.0088 - accuracy: 0.9975 - val_loss: 0.0034 - val_accuracy: 0.9994

train_accuracy_vgg16 = history_vgg16.history['accuracy']
val_accuracy_vgg16 = history_vgg16.history['val_accuracy']

epochs_range = range(1, len(train_accuracy_vgg16) + 1)

plt.plot(epochs_range, train_accuracy_vgg16, label='Training
Accuracy', marker='o')
plt.plot(epochs_range, val_accuracy_vgg16, label='Validation
Accuracy', marker='o')

plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title(f'{model_name} Model Accuracy')
plt.legend()

plt.xticks(range(1, len(train_accuracy_vgg16) + 1))

plt.grid(True)
plt.show()
```



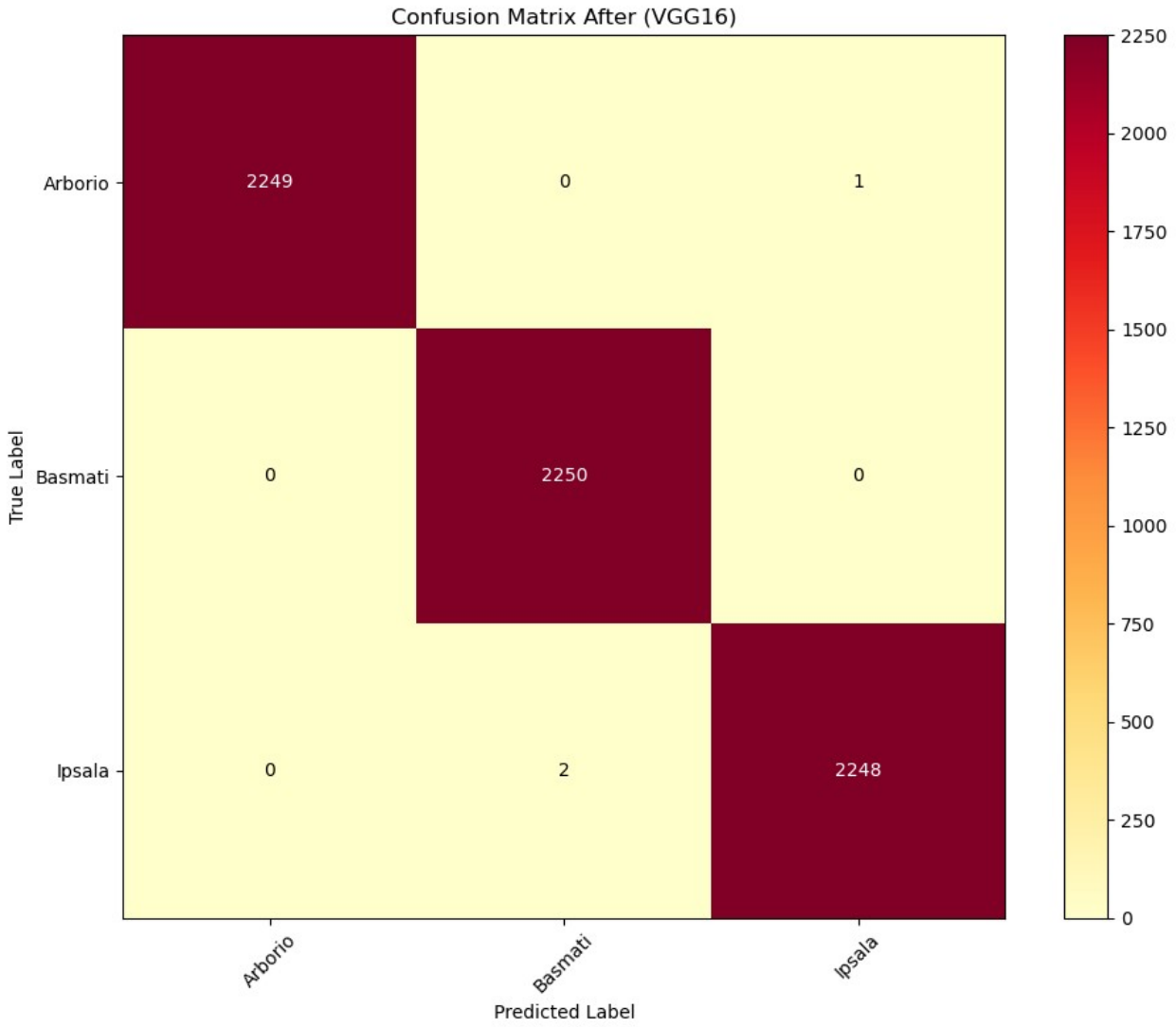

```
print("Evaluating after training...")
vgg16_testing_scores = model_vgg16.evaluate(test_generator)
vgg16_testing_accuracy = vgg16_testing_scores[1]
```

```
Evaluating after training...
211/211 [==============================] - 522s 2s/step - loss:
5.9087e-04 - accuracy: 0.9996
```

```
vgg16_y_true = []
vgg16_y_pred = []
for images, labels in test_generator:
    predictions = model_vgg16.predict(images, verbose=0)
    predicted_labels = tf.argmax(predictions, axis=1)
    vgg16_y_true.extend(labels.numpy())
    vgg16_y_pred.extend(predicted_labels.numpy())

vgg16_confusion_matrix = confusion_matrix(vgg16_y_true, vgg16_y_pred,
labels=range(num_classes))

vgg16_f1_score = f1_score(vgg16_y_true, vgg16_y_pred,
average='weighted')

vgg16_classification_report = classification_report(vgg16_y_true,
vgg16_y_pred, target_names=class_names)
```

```
print("Testing Accuracy (VGG16):", vgg16_testing_accuracy)
print("F1 Score (VGG16):", vgg16_f1_score)
print("Classification Report After (VGG16):\n",
vgg16_classification_report)

plt.figure(figsize=(10, 8))
plt.imshow(vgg16_confusion_matrix, interpolation='nearest',
cmap=plt.cm.YlOrRd)
plt.title('Confusion Matrix After (VGG16)')
plt.colorbar()

tick_marks = np.arange(num_classes)
plt.xticks(tick_marks, class_names, rotation=45)
plt.yticks(tick_marks, class_names)

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.grid(False)

thresh = vgg16_confusion_matrix.max() / 2.
for i in range(num_classes):
    for j in range(num_classes):
        plt.text(j, i, format(vgg16_confusion_matrix[i, j], 'd'),
                 ha="center", va="center",
                 color="white" if vgg16_confusion_matrix[i, j] >
thresh else "black")

plt.tight_layout()
plt.show()
```

```
Testing Accuracy (VGG16): 0.9995555281639099
F1 Score (VGG16): 0.9995555555116695
Classification Report After (VGG16):
               precision    recall  f1-score   support

     Arborio       1.00      1.00      1.00      2250
     Basmati       1.00      1.00      1.00      2250
      Ipsala       1.00      1.00      1.00      2250

    accuracy                           1.00      6750
   macro avg       1.00      1.00      1.00      6750
weighted avg       1.00      1.00      1.00      6750
```



RESNET50 MODEL

```
model_name = "ResNet50"
epochs = 5
desired_steps_per_epoch = 30

num_classes = len(train_generator.class_names)

base_model_resnet50 = ResNet50(include_top=False, weights='imagenet',
input_shape=(iz, iz, 3))

base_model_resnet50.trainable = False

model_resnet50 = tf_models.Sequential(
[
    base_model_resnet50,
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu'),
```

```
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation='softmax')
])

model_resnet50.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])



# *** Evaluate accuracy and confusion matrix BEFORE training ***
print("Evaluating before training...")

resnet50_y_true = []
resnet50_y_pred = []

for images, labels in test_generator:
    predictions = model_resnet50.predict(images, verbose=0)
    predicted_labels = tf.argmax(predictions, axis=1)
    resnet50_y_true.extend(labels.numpy())
    resnet50_y_pred.extend(predicted_labels.numpy())

# Confusion matrix and accuracy before training
resnet50_confusion_matrix_before = confusion_matrix(resnet50_y_true,
resnet50_y_pred, labels=range(num_classes))
resnet50_f1_score_before = f1_score(resnet50_y_true, resnet50_y_pred,
average='weighted')
resnet50_classification_report_before =
classification_report(resnet50_y_true, resnet50_y_pred,
target_names=class_names)

print("Confusion Matrix Before Training:\n",
resnet50_confusion_matrix_before)
print("F1 Score Before Training:", resnet50_f1_score_before)
print("Classification Report Before Training:\n",
resnet50_classification_report_before)

plt.figure(figsize=(10, 8))
plt.imshow(resnet50_confusion_matrix_before, interpolation='nearest',
cmap=plt.cm.YlOrRd)
plt.title('Confusion Matrix (ResNet50) Before Training')
plt.colorbar()

tick_marks = np.arange(num_classes)
plt.xticks(tick_marks, class_names, rotation=45)
plt.yticks(tick_marks, class_names)

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.grid(False)

thresh = resnet50_confusion_matrix_before.max() / 2.
```

```
for i in range(num_classes):
    for j in range(num_classes):
        plt.text(j, i, format(resnet50_confusion_matrix_before[i, j],
'd'),
                 ha="center", va="center",
                 color="white" if resnet50_confusion_matrix_before[i,
j] > thresh else "black")

plt.tight_layout()
plt.show()
```

```
Evaluating before training...
```

```
---------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Cell In[13], line 8
      5 resnet50_y_pred = []
      7 for images, labels in test_generator:
----> 8     predictions = model_resnet50.predict(images, verbose=0)
      9     predicted_labels = tf.argmax(predictions, axis=1)
     10     resnet50_y_true.extend(labels.numpy())

File c:\Users\iamim\anaconda3\Lib\site-packages\keras\src\utils\
traceback_utils.py:65, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
     63 filtered_tb = None
     64 try:
---> 65     return fn(*args, **kwargs)
     66 except Exception as e:
     67     filtered_tb = _process_traceback_frames(e.__traceback__)

File c:\Users\iamim\anaconda3\Lib\site-packages\keras\src\engine\
training.py:2655, in Model.predict(self, x, batch_size, verbose,
steps, callbacks, max_queue_size, workers, use_multiprocessing)
   2653 for step in data_handler.steps():
   2654     callbacks.on_predict_batch_begin(step)
-> 2655     tmp_batch_outputs = self.predict_function(iterator)
   2656     if data_handler.should_sync:
   2657         context.async_wait()

File c:\Users\iamim\anaconda3\Lib\site-packages\tensorflow\python\
util\traceback_utils.py:150, in
filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)
```

```
File c:\Users\iamim\anaconda3\Lib\site-packages\tensorflow\python\
eager\polymorphic_function\polymorphic_function.py:832, in
Function.__call__(self, *args, **kwds)
    829 compiler = "xla" if self._jit_compile else "nonXla"
    831 with OptionalXlaContext(self._jit_compile):
--> 832   result = self._call(*args, **kwds)
    834 new_tracing_count = self.experimental_get_tracing_count()
    835 without_tracing = (tracing_count == new_tracing_count)

File c:\Users\iamim\anaconda3\Lib\site-packages\tensorflow\python\
eager\polymorphic_function\polymorphic_function.py:877, in
Function._call(self, *args, **kwds)
    874 self._lock.release()
    875 # In this case we have not created variables on the first
call. So we can
    876 # run the first trace but we should fail if variables are
created.
--> 877 results = tracing_compilation.call_function(
    878     args, kwds, self._variable_creation_config
    879 )
    880 if self._created_variables:
    881   raise ValueError("Creating variables on a non-first call to
a function"
    882                    " decorated with tf.function.")

File c:\Users\iamim\anaconda3\Lib\site-packages\tensorflow\python\
eager\polymorphic_function\tracing_compilation.py:139, in
call_function(args, kwargs, tracing_options)
    137 bound_args = function.function_type.bind(*args, **kwargs)
    138 flat_inputs = function.function_type.unpack_inputs(bound_args)
--> 139 return function._call_flat(  # pylint: disable=protected-
access
    140     flat_inputs, captured_inputs=function.captured_inputs
    141 )

File c:\Users\iamim\anaconda3\Lib\site-packages\tensorflow\python\
eager\polymorphic_function\concrete_function.py:1323, in
ConcreteFunction._call_flat(self, tensor_inputs, captured_inputs)
   1319 possible_gradient_type =
gradients_util.PossibleTapeGradientTypes(args)
   1320 if (possible_gradient_type ==
gradients_util.POSSIBLE_GRADIENT_TYPES_NONE
   1321     and executing_eagerly):
   1322   # No tape is watching; skip to running the function.
-> 1323   return self._inference_function.call_preflattened(args)
   1324 forward_backward =
self._select_forward_and_backward_functions(
   1325     args,
   1326     possible_gradient_type,
```

```
   1327     executing_eagerly)
   1328 forward_function, args_with_tangents =
forward_backward.forward()

File c:\Users\iamim\anaconda3\Lib\site-packages\tensorflow\python\
eager\polymorphic_function\atomic_function.py:216, in
AtomicFunction.call_preflattened(self, args)
    214 def call_preflattened(self, args: Sequence[core.Tensor]) ->
Any:
    215   """Calls with flattened tensor inputs and returns the
structured output."""
--> 216   flat_outputs = self.call_flat(*args)
    217   return self.function_type.pack_output(flat_outputs)

File c:\Users\iamim\anaconda3\Lib\site-packages\tensorflow\python\
eager\polymorphic_function\atomic_function.py:251, in
AtomicFunction.call_flat(self, *args)
    249 with record.stop_recording():
    250   if self._bound_context.executing_eagerly():
--> 251     outputs = self._bound_context.call_function(
    252         self.name,
    253         list(args),
    254         len(self.function_type.flat_outputs),
    255     )
    256   else:
    257     outputs = make_call_op_in_graph(
    258         self,
    259         list(args),
    260         self._bound_context.function_call_options.as_attrs(),
    261     )

File c:\Users\iamim\anaconda3\Lib\site-packages\tensorflow\python\
eager\context.py:1486, in Context.call_function(self, name,
tensor_inputs, num_outputs)
   1484 cancellation_context = cancellation.context()
   1485 if cancellation_context is None:
-> 1486   outputs = execute.execute(
   1487       name.decode("utf-8"),
   1488       num_outputs=num_outputs,
   1489       inputs=tensor_inputs,
   1490       attrs=attrs,
   1491       ctx=self,
   1492   )
   1493 else:
   1494   outputs = execute.execute_with_cancellation(
   1495       name.decode("utf-8"),
   1496       num_outputs=num_outputs,
   (...)
   1500       cancellation_manager=cancellation_context,
   1501   )
```

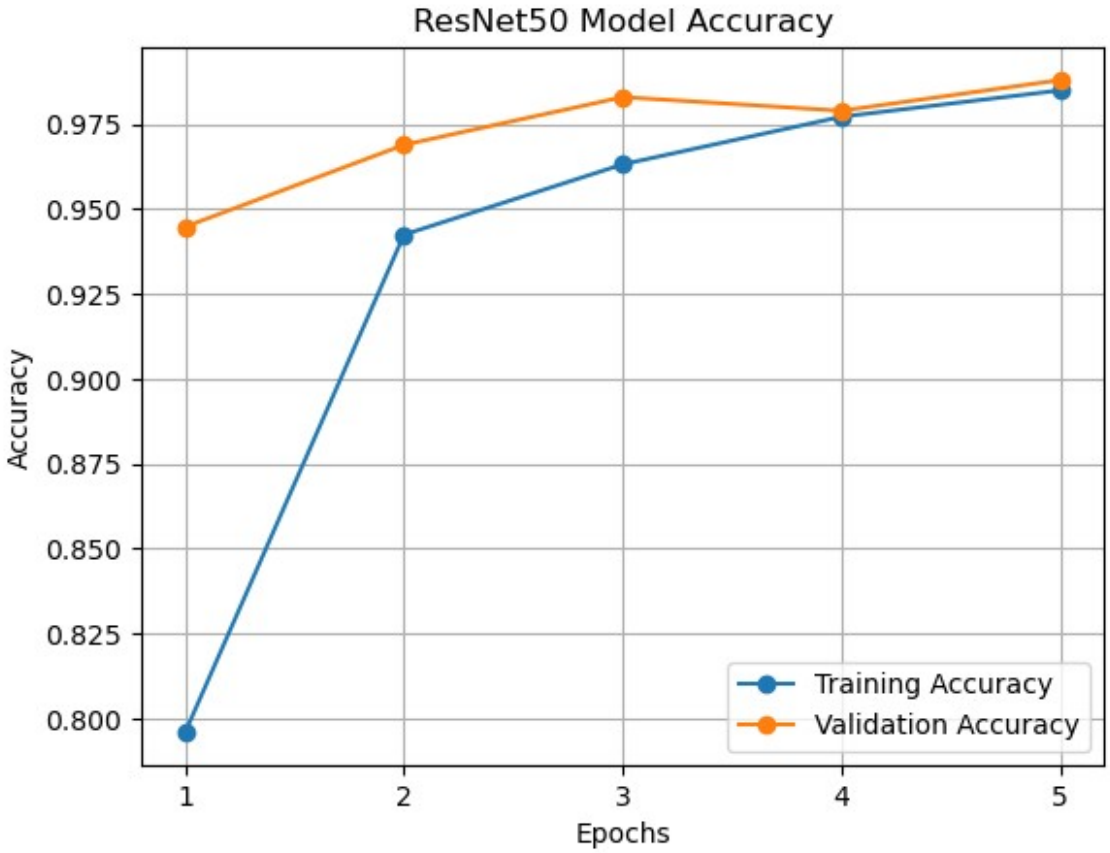
```
File c:\Users\iamim\anaconda3\Lib\site-packages\tensorflow\python\
eager\execute.py:53, in quick_execute(op_name, num_outputs, inputs,
attrs, ctx, name)
     51 try:
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle,
device_name, op_name,
     54                                       inputs, attrs,
num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:

KeyboardInterrupt:
```

```
history_resnet50 = model_resnet50.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=epochs,
)
```

```
Epoch 1/5
145/145 [==============================] - 411s 3s/step - loss: 0.5357
- accuracy: 0.7961 - val_loss: 0.1699 - val_accuracy: 0.9448
Epoch 2/5
145/145 [==============================] - 403s 3s/step - loss: 0.1700
- accuracy: 0.9424 - val_loss: 0.1034 - val_accuracy: 0.9689
Epoch 3/5
145/145 [==============================] - 386s 3s/step - loss: 0.1054
- accuracy: 0.9631 - val_loss: 0.0666 - val_accuracy: 0.9829
Epoch 4/5
145/145 [==============================] - 363s 3s/step - loss: 0.0696
- accuracy: 0.9772 - val_loss: 0.0569 - val_accuracy: 0.9789
Epoch 5/5
145/145 [==============================] - 325s 2s/step - loss: 0.0482
- accuracy: 0.9849 - val_loss: 0.0388 - val_accuracy: 0.9880
```

```
train_accuracy_resnet50 = history_resnet50.history['accuracy']
val_accuracy_resnet50 = history_resnet50.history['val_accuracy']

epochs_range = range(1, len(train_accuracy_resnet50) + 1)

plt.plot(epochs_range, train_accuracy_resnet50, label='Training
Accuracy', marker='o')
plt.plot(epochs_range, val_accuracy_resnet50, label='Validation
Accuracy', marker='o')

plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('ResNet50 Model Accuracy')
```

```
plt.legend()

plt.xticks(range(1, len(train_accuracy_resnet50) + 1))

plt.grid(True)
plt.show()
```




```
print("Evaluating after training...")
resnet50_testing_scores = model_resnet50.evaluate(test_generator)
resnet50_testing_accuracy = resnet50_testing_scores[1]
```

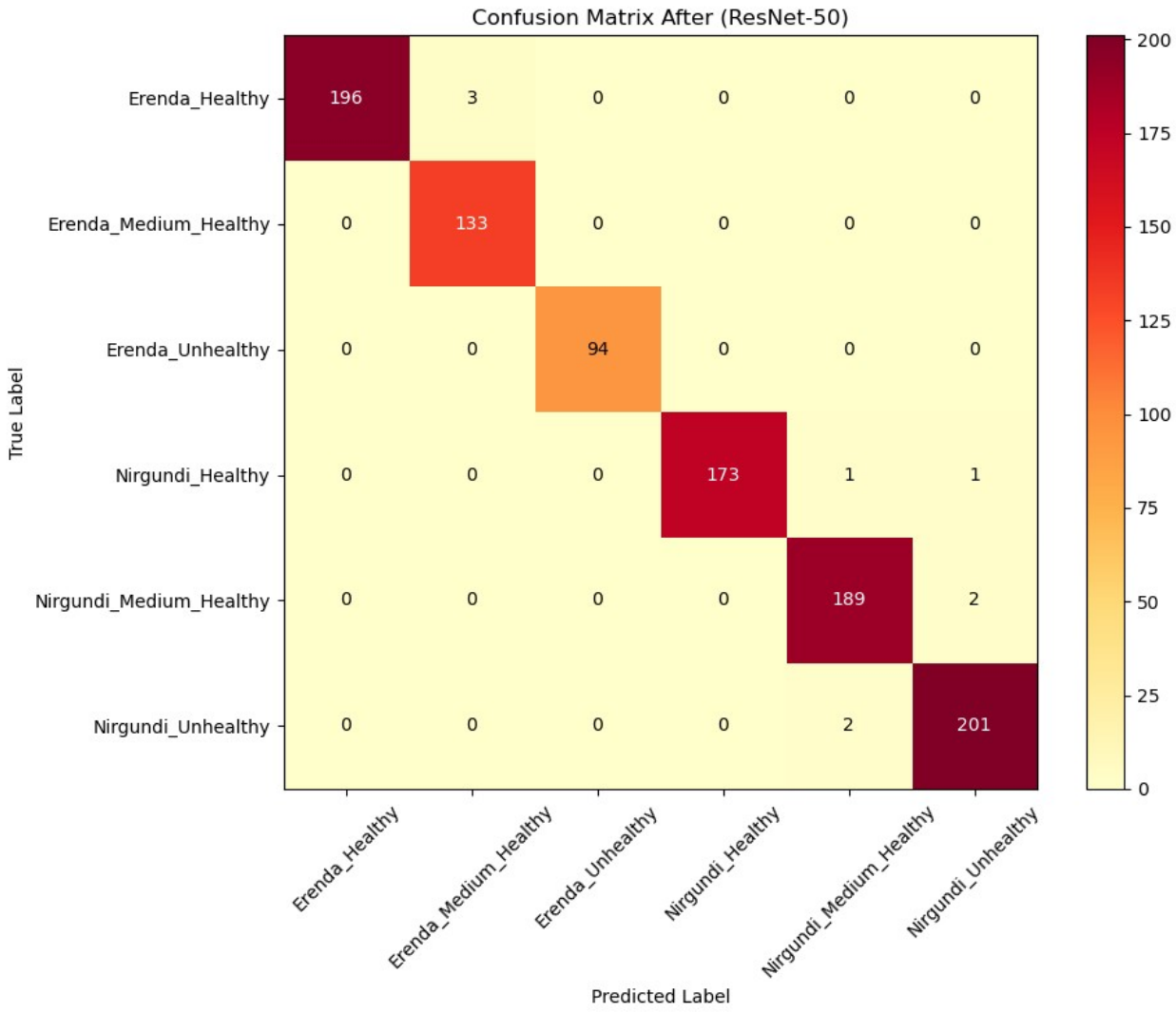
```
Evaluating after training...
32/32 [==============================] - 52s 2s/step - loss: 0.0335 -
accuracy: 0.9910
```

```
resnet50_y_true = []
resnet50_y_pred = []

for images, labels in test_generator:
    predictions = model_resnet50.predict(images, verbose=0)
    predicted_labels = tf.argmax(predictions, axis=1)
    resnet50_y_true.extend(labels.numpy())
    resnet50_y_pred.extend(predicted_labels.numpy())
```

```
resnet50_confusion_matrix = confusion_matrix(resnet50_y_true,
resnet50_y_pred, labels=range(num_classes))

resnet50_f1_score = f1_score(resnet50_y_true, resnet50_y_pred,
average='weighted')

resnet50_classification_report =
classification_report(resnet50_y_true, resnet50_y_pred,
target_names=class_names)

print("Testing Accuracy After Training (ResNet-50):",
resnet50_testing_accuracy)
print("F1 Score After (ResNet-50):", resnet50_f1_score)
print("Classification Report After (ResNet-50):\n",
resnet50_classification_report)

plt.figure(figsize=(10, 8))
plt.imshow(resnet50_confusion_matrix, interpolation='nearest',
cmap=plt.cm.YlOrRd)
plt.title('Confusion Matrix After (ResNet-50)')
plt.colorbar()

tick_marks = np.arange(num_classes)
plt.xticks(tick_marks, class_names, rotation=45)
plt.yticks(tick_marks, class_names)

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.grid(False)

thresh = resnet50_confusion_matrix.max() / 2.
for i in range(num_classes):
    for j in range(num_classes):
        plt.text(j, i, format(resnet50_confusion_matrix[i, j], 'd'),
                 ha="center", va="center",
                 color="white" if resnet50_confusion_matrix[i, j] >
thresh else "black")

plt.tight_layout()
plt.show()
```

```
Testing Accuracy After Training (ResNet-50): 0.9909547567367554
F1 Score After (ResNet-50): 0.9909670944867551
Classification Report After (ResNet-50):
                          precision    recall  f1-score   support

         Erenda_Healthy       1.00      0.98      0.99       199
  Erenda_Medium_Healthy       0.98      1.00      0.99       133
      Erenda_Unhealthy       1.00      1.00      1.00        94
```

Nirgundi_Healthy	1.00	0.99	0.99	175
Nirgundi_Medium_Healthy	0.98	0.99	0.99	191
Nirgundi_Unhealthy	0.99	0.99	0.99	203
accuracy			0.99	995
macro avg	0.99	0.99	0.99	995
weighted avg	0.99	0.99	0.99	995



```python
import pandas as pd

# Data
data = {
    "Model": ["VGG16", "ResNet50"],
    "Before Training Accuracy": ["15%", "18%"],
```

```
    # "Before Training F1 Score": [0.06, 0.05, 0.03, 0.21, 0.10, 0.21,
0.10],

    "After Training Accuracy": ["98%", "98%"],
    # "After Training F1 Score": [0.98, 0.99, 0.60, 0.94, 0.61, 1.00,
0.99]
}

# Create DataFrame
df = pd.DataFrame(data)

# Display the table
print(df.to_string(index=False))
```

```
         Model Before Training Accuracy After Training Accuracy
         VGG16                      15%                      98%
      ResNet50                      18%                      98%
   InceptionV3                      21%                      38%
   MobileNetV2                      18%                      88%
      Xception                      12%                      72%
EfficientNetB0                      19%                      99%
         VGG19                      14%                      99%
```
