05_model_t_tl_feat_ext_rmsprop

June 10, 2024

---

1 Model T - Transfer Learning, Feature Extraction, RMSProp

- **128 x 128 x 3** Image size.

- **64** Batch size.
- **Feature Extraction**.
  - **VGG16** Convolutional Base.
  - **4 x 4 x 512** Feature Maps.
- **Classifier**:
  - **Root Mean Square Propagation (RMSProp)** optimizer.
  - **0.01** Initial Learning rate.
  - **Sparse Categorical Cross-Entropy** loss function.
  - **Reduce Learning Rate on Plateau** callback with a **0.1** factor and patience of **3**.
  - **Early Stopping** callback with patience of **6** and restore best weights.
  - **Model Checkpoint** callback to save the best model based on validation loss.
  - **4 x 4 x 512** Tensor before the **Flatten** layer.
  - **512** Dense layer with **ReLU** activation.
  - **10** Dense output layer with **Softmax** activation.
  - **Dropout** layers with **0.5** rate after the Flatten and Dense layers.
  - **L2** regularization with **0.0001** rate on the Dense layers.
  - **4 199 946** Trainable Parameters.
  - **30 Epochs** to train the classifier.

- Build the **full model** with the **VGG16 Convolutional Base** and the **Classifier**.
- **Test** the model on the **Test Dataset**.

Imports and Setup

[1]:
```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
print(f'TensorFlow version: {tf.__version__}')
tf.get_logger().setLevel('ERROR')
tf.autograph.set_verbosity(3)
import matplotlib.pyplot as plt
```

```python
import pickle
import numpy as np
from tensorflow.keras.utils import image_dataset_from_directory
from tensorflow import keras
from tensorflow.keras import callbacks, layers, optimizers
from keras import regularizers, Model
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay␣
↪,accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
from sklearn.preprocessing import label_binarize
from itertools import cycle
```

```
TensorFlow version: 2.15.0
```

Group Datasets

[2]:
```python
IMG_SIZE = 128

train_dirs = [f'../data/train1_resized_{IMG_SIZE}',
              f'../data/train3_resized_{IMG_SIZE}',
              f'../data/train4_resized_{IMG_SIZE}',
              f'../data/train5_resized_{IMG_SIZE}']
validation_dir = f'../data/train2_resized_{IMG_SIZE}'
test_dir = f'../data/test_resized_{IMG_SIZE}'
```

Create Datasets

[3]:
```python
BATCH_SIZE = 64
NUM_CLASSES = 10

train_datasets = [image_dataset_from_directory(directory, image_size=(IMG_SIZE,␣
↪IMG_SIZE), batch_size=BATCH_SIZE) for directory in train_dirs]

train_dataset = train_datasets[0]
for dataset in train_datasets[1:]:
    train_dataset = train_dataset.concatenate(dataset)

train_dataset = train_dataset.shuffle(buffer_size=1000).prefetch(buffer_size=tf.
↪data.AUTOTUNE)
validation_dataset = image_dataset_from_directory(validation_dir,␣
↪image_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE).
↪prefetch(buffer_size=tf.data.AUTOTUNE)
test_dataset = image_dataset_from_directory(test_dir, image_size=(IMG_SIZE,␣
↪IMG_SIZE), batch_size=BATCH_SIZE).prefetch(buffer_size=tf.data.AUTOTUNE)

class_names = train_datasets[0].class_names
```

```
for data_batch, labels_batch in train_dataset.take(1):
    print('data batch shape:', data_batch.shape)
    print('labels batch shape:', labels_batch.shape)
```

```
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
data batch shape: (64, 128, 128, 3)
labels batch shape: (64,)
```

- We define the image size of 128 x 128 x 3, batch size of 64 and create an array with the label's names.
- We create the train dataset by concatenating them, we **shuffle** the samples before each epoch and **prefetch** them to memory.
- We do the same for the validation and test dataset except **shuffling** which is **unwanted** for these datasets.

Loading the VGG16 Model

[4]:
```
from tensorflow.keras.applications.vgg16 import VGG16
conv_base = VGG16(weights='imagenet', include_top=False, input_shape=(IMG_SIZE,␣
↪IMG_SIZE, 3))
conv_base.trainable = False
conv_base.summary()
```

```
Model: "vgg16"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 128, 128, 3)]     0

 block1_conv1 (Conv2D)       (None, 128, 128, 64)      1792

 block1_conv2 (Conv2D)       (None, 128, 128, 64)      36928

 block1_pool (MaxPooling2D)  (None, 64, 64, 64)        0

 block2_conv1 (Conv2D)       (None, 64, 64, 128)       73856

 block2_conv2 (Conv2D)       (None, 64, 64, 128)       147584
```

```
 block2_pool (MaxPooling2D)  (None, 32, 32, 128)       0

 block3_conv1 (Conv2D)       (None, 32, 32, 256)       295168

 block3_conv2 (Conv2D)       (None, 32, 32, 256)       590080

 block3_conv3 (Conv2D)       (None, 32, 32, 256)       590080

 block3_pool (MaxPooling2D)  (None, 16, 16, 256)       0

 block4_conv1 (Conv2D)       (None, 16, 16, 512)       1180160

 block4_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 8, 8, 512)         0

 block5_conv1 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 0 (0.00 Byte)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

- We **load the VGG16** model with the **imagenet weights**, **without the top layer** and with the input shape of **128 x 128 pixels and 3 channels**.

Feature Extraction

[5]:
```python
def get_features_and_labels(dataset):
    all_features = []
    all_labels = []
    for images, labels in dataset:
        preprocessed_images = keras.applications.vgg16.preprocess_input(images)
        features = conv_base.predict(preprocessed_images)
        all_features.append(features)
        all_labels.append(labels)
    return np.concatenate(all_features), np.concatenate(all_labels)
```

```
train_features, train_labels = get_features_and_labels(train_dataset)
val_features, val_labels = get_features_and_labels(validation_dataset)
```

```
2/2 [==============================] - 1s 619ms/step
2/2 [==============================] - 1s 621ms/step
2/2 [==============================] - 1s 631ms/step
2/2 [==============================] - 1s 632ms/step
2/2 [==============================] - 1s 623ms/step
2/2 [==============================] - 1s 620ms/step
2/2 [==============================] - 1s 619ms/step
2/2 [==============================] - 1s 634ms/step
2/2 [==============================] - 1s 620ms/step
2/2 [==============================] - 1s 622ms/step
2/2 [==============================] - 1s 619ms/step
2/2 [==============================] - 1s 619ms/step
2/2 [==============================] - 1s 621ms/step
2/2 [==============================] - 1s 625ms/step
2/2 [==============================] - 1s 628ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 700ms/step
2/2 [==============================] - 1s 714ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 680ms/step
```

```
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 673ms/step
1/1 [==============================] - 0s 392ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 710ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
```

```
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 2s 1s/step
2/2 [==============================] - 2s 825ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 707ms/step
2/2 [==============================] - 1s 719ms/step
2/2 [==============================] - 1s 714ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 807ms/step
2/2 [==============================] - 2s 763ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 716ms/step
2/2 [==============================] - 1s 678ms/step
```

```
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 2s 795ms/step
2/2 [==============================] - 2s 999ms/step
1/1 [==============================] - 0s 426ms/step
2/2 [==============================] - 2s 893ms/step
2/2 [==============================] - 2s 686ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 2s 844ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 699ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 707ms/step
2/2 [==============================] - 2s 867ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 2s 717ms/step
2/2 [==============================] - 2s 893ms/step
2/2 [==============================] - 2s 988ms/step
2/2 [==============================] - 2s 685ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 804ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 2s 675ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 2s 871ms/step
2/2 [==============================] - 2s 725ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 699ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 708ms/step
2/2 [==============================] - 1s 699ms/step
2/2 [==============================] - 1s 721ms/step
2/2 [==============================] - 1s 708ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 741ms/step
2/2 [==============================] - 2s 681ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 679ms/step
```

```
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 2s 833ms/step
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 695ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 2s 689ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 697ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 697ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 725ms/step
2/2 [==============================] - 1s 705ms/step
2/2 [==============================] - 1s 729ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 2s 690ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 2s 802ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 680ms/step
```

```
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 2s 876ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 2s 700ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 2s 946ms/step
2/2 [==============================] - 2s 815ms/step
2/2 [==============================] - 2s 729ms/step
2/2 [==============================] - 1s 790ms/step
2/2 [==============================] - 1s 708ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 2s 946ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 725ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
```

```
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 713ms/step
2/2 [==============================] - 2s 759ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 2s 833ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 2s 841ms/step
2/2 [==============================] - 2s 677ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 2s 717ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 2s 822ms/step
2/2 [==============================] - 2s 677ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 2s 1s/step
2/2 [==============================] - 2s 947ms/step
2/2 [==============================] - 2s 877ms/step
2/2 [==============================] - 2s 680ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 2s 877ms/step
2/2 [==============================] - 2s 919ms/step
2/2 [==============================] - 2s 850ms/step
2/2 [==============================] - 2s 929ms/step
2/2 [==============================] - 2s 859ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 705ms/step
2/2 [==============================] - 1s 708ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 684ms/step
```

```
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 2s 780ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 2s 684ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 706ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 699ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 2s 923ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 715ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 717ms/step
1/1 [==============================] - 0s 362ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 676ms/step
```

```
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 706ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 711ms/step
2/2 [==============================] - 1s 723ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 2s 707ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 678ms/step
```

```
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 2s 811ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 705ms/step
2/2 [==============================] - 1s 711ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 699ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 697ms/step
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 697ms/step
2/2 [==============================] - 1s 708ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 695ms/step
2/2 [==============================] - 1s 705ms/step
2/2 [==============================] - 1s 713ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 700ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 695ms/step
```

```
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 714ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 705ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 2s 785ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 724ms/step
2/2 [==============================] - 1s 738ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 2s 816ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 697ms/step
2/2 [==============================] - 2s 895ms/step
2/2 [==============================] - 2s 842ms/step
2/2 [==============================] - 2s 865ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 2s 683ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 2s 678ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 755ms/step
2/2 [==============================] - 2s 716ms/step
2/2 [==============================] - 1s 751ms/step
2/2 [==============================] - 2s 961ms/step
2/2 [==============================] - 2s 844ms/step
```

```
2/2 [==============================] - 2s 704ms/step
2/2 [==============================] - 1s 706ms/step
2/2 [==============================] - 1s 691ms/step
1/1 [==============================] - 0s 358ms/step
2/2 [==============================] - 1s 709ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 2s 829ms/step
2/2 [==============================] - 1s 697ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 722ms/step
2/2 [==============================] - 1s 729ms/step
2/2 [==============================] - 1s 721ms/step
2/2 [==============================] - 1s 707ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 675ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 676ms/step
```

```
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 677ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 672ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 708ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 672ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 673ms/step
2/2 [==============================] - 1s 671ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 674ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 2s 890ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 696ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 686ms/step
```

```
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 2s 692ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 725ms/step
2/2 [==============================] - 1s 722ms/step
2/2 [==============================] - 1s 726ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
```

```
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 2s 687ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 2s 880ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 695ms/step
2/2 [==============================] - 1s 703ms/step
2/2 [==============================] - 1s 711ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 697ms/step
2/2 [==============================] - 1s 706ms/step
2/2 [==============================] - 1s 738ms/step
2/2 [==============================] - 1s 706ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 704ms/step
```

```
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 705ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 700ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 707ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 693ms/step
2/2 [==============================] - 1s 692ms/step
```

```
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 697ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
1/1 [==============================] - 0s 351ms/step
```

- We **extract the features** from the convolutional base of the VGG16 model for the **train and validation dataset**.
- We **don't need** to extract **the features from the test dataset** because we will use the full model to predict it.

1.0.1 Saving the features and labels

[6]:
```
np.save('../features/05_model_t_tl_feat_ext_rmsprop_train_features.npy',␣
 ↪train_features)
np.save('../features/05_model_t_tl_feat_ext_rmsprop_train_labels.npy',␣
 ↪train_labels)
np.save('../features/05_model_t_tl_feat_ext_rmsprop_val_features.npy',␣
 ↪val_features)
np.save('../features/05_model_t_tl_feat_ext_rmsprop_val_labels.npy', val_labels)
```

1.0.2 Loading the features and labels

[7]:
```
# train_features = np.load('../features/
 ↪05_model_t_tl_feat_ext_rmsprop_train_features.npy')
# train_labels = np.load('../features/
 ↪05_model_t_tl_feat_ext_rmsprop_train_labels.npy')
```

```
# val_features = np.load('../features/
↪05_model_t_tl_feat_ext_rmsprop_val_features.npy')
# val_labels = np.load('../features/05_model_t_tl_feat_ext_rmsprop_val_labels.
↪npy')
```

Classifier Arquitecture

[8]:
```
inputs = keras.Input(shape=(4, 4, 512))
x = layers.Flatten()(inputs)
x = layers.Dropout(0.5)(x)
x = layers.Dense(512, activation="relu", kernel_regularizer=regularizers.
↪L2(1e-4))(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(NUM_CLASSES, activation="softmax",␣
↪kernel_regularizer=regularizers.L2(1e-4))(x)
classifier = keras.Model(inputs, outputs)
classifier.summary()
```

```
Model: "model"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 4, 4, 512)]       0

 flatten (Flatten)           (None, 8192)              0

 dropout (Dropout)           (None, 8192)              0

 dense (Dense)               (None, 512)               4194816

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 4199946 (16.02 MB)
Trainable params: 4199946 (16.02 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Model Compilation

[9]:
```
initial_learning_rate = 0.001
optimizer = optimizers.RMSprop(learning_rate=initial_learning_rate)
loss_function = keras.losses.SparseCategoricalCrossentropy()
```

```
lr_scheduler = callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.1,␣
 ↪patience=3, verbose=1)
early_stopping = callbacks.EarlyStopping(monitor='val_loss', patience=6,␣
 ↪restore_best_weights=True, verbose=1)
save_best_model = callbacks.ModelCheckpoint(filepath='../models/
 ↪05_model_t_tl_feat_ext_rmsprop_classifier.h5', save_best_only=True,␣
 ↪monitor='val_loss', verbose=1)

callbacks = [lr_scheduler, early_stopping, save_best_model]

classifier.compile(optimizer=optimizer,
                   loss=loss_function,
                   metrics=['accuracy'])
```

Model Training

[10]:
```
history = classifier.fit(
    train_features, train_labels,
    epochs=30,
    validation_data=(val_features, val_labels),
    callbacks=callbacks)
```

```
Epoch 1/30
1249/1250 [============================>.] - ETA: 0s - loss: 2.0882 - accuracy:
0.7210
Epoch 1: val_loss improved from inf to 0.69868, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 25s 20ms/step - loss: 2.0876 -
accuracy: 0.7211 - val_loss: 0.6987 - val_accuracy: 0.8348 - lr: 0.0010
Epoch 2/30
   7/1250 […] - ETA: 23s - loss: 1.3775 - accuracy:
0.7455
```

```
/usr/local/lib/python3.9/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
1249/1250 [============================>.] - ETA: 0s - loss: 1.1691 - accuracy:
0.7922
Epoch 2: val_loss improved from 0.69868 to 0.65767, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 24s 19ms/step - loss: 1.1695 -
accuracy: 0.7921 - val_loss: 0.6577 - val_accuracy: 0.8536 - lr: 0.0010
Epoch 3/30
```

```
1249/1250 [============================>.] - ETA: 0s - loss: 1.0680 - accuracy:
0.8083
Epoch 3: val_loss improved from 0.65767 to 0.64178, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 24s 19ms/step - loss: 1.0682 -
accuracy: 0.8083 - val_loss: 0.6418 - val_accuracy: 0.8589 - lr: 0.0010
Epoch 4/30
1249/1250 [============================>.] - ETA: 0s - loss: 1.0201 - accuracy:
0.8153
Epoch 4: val_loss did not improve from 0.64178
1250/1250 [==============================] - 24s 19ms/step - loss: 1.0200 -
accuracy: 0.8154 - val_loss: 0.6591 - val_accuracy: 0.8540 - lr: 0.0010
Epoch 5/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.9945 - accuracy:
0.8241
Epoch 5: val_loss did not improve from 0.64178
1250/1250 [==============================] - 23s 19ms/step - loss: 0.9945 -
accuracy: 0.8241 - val_loss: 0.6654 - val_accuracy: 0.8655 - lr: 0.0010
Epoch 6/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.9914 - accuracy:
0.8252
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.

Epoch 6: val_loss did not improve from 0.64178
1250/1250 [==============================] - 24s 19ms/step - loss: 0.9913 -
accuracy: 0.8253 - val_loss: 0.6751 - val_accuracy: 0.8596 - lr: 0.0010
Epoch 7/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.7153 - accuracy:
0.8567
Epoch 7: val_loss improved from 0.64178 to 0.60073, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 23s 19ms/step - loss: 0.7160 -
accuracy: 0.8567 - val_loss: 0.6007 - val_accuracy: 0.8780 - lr: 1.0000e-04
Epoch 8/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.6890 - accuracy:
0.8693
Epoch 8: val_loss improved from 0.60073 to 0.59749, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 24s 19ms/step - loss: 0.6887 -
accuracy: 0.8694 - val_loss: 0.5975 - val_accuracy: 0.8820 - lr: 1.0000e-04
Epoch 9/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.6423 - accuracy:
0.8773
Epoch 9: val_loss improved from 0.59749 to 0.58528, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 24s 19ms/step - loss: 0.6427 -
accuracy: 0.8773 - val_loss: 0.5853 - val_accuracy: 0.8840 - lr: 1.0000e-04
Epoch 10/30
```

```
1249/1250 [============================>.] - ETA: 0s - loss: 0.6051 - accuracy:
0.8843
Epoch 10: val_loss improved from 0.58528 to 0.58060, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 23s 19ms/step - loss: 0.6049 -
accuracy: 0.8843 - val_loss: 0.5806 - val_accuracy: 0.8882 - lr: 1.0000e-04
Epoch 11/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.5881 - accuracy:
0.8877
Epoch 11: val_loss improved from 0.58060 to 0.57706, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 24s 19ms/step - loss: 0.5879 -
accuracy: 0.8877 - val_loss: 0.5771 - val_accuracy: 0.8866 - lr: 1.0000e-04
Epoch 12/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.5687 - accuracy:
0.8927
Epoch 12: val_loss improved from 0.57706 to 0.57455, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 23s 19ms/step - loss: 0.5687 -
accuracy: 0.8927 - val_loss: 0.5746 - val_accuracy: 0.8886 - lr: 1.0000e-04
Epoch 13/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.5586 - accuracy:
0.8910
Epoch 13: val_loss improved from 0.57455 to 0.57241, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 23s 19ms/step - loss: 0.5583 -
accuracy: 0.8910 - val_loss: 0.5724 - val_accuracy: 0.8926 - lr: 1.0000e-04
Epoch 14/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.5354 - accuracy:
0.8969
Epoch 14: val_loss improved from 0.57241 to 0.56496, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 23s 19ms/step - loss: 0.5353 -
accuracy: 0.8970 - val_loss: 0.5650 - val_accuracy: 0.8893 - lr: 1.0000e-04
Epoch 15/30
1248/1250 [============================>.] - ETA: 0s - loss: 0.5137 - accuracy:
0.8961
Epoch 15: val_loss improved from 0.56496 to 0.55465, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 24s 20ms/step - loss: 0.5140 -
accuracy: 0.8961 - val_loss: 0.5546 - val_accuracy: 0.8932 - lr: 1.0000e-04
Epoch 16/30
1250/1250 [==============================] - ETA: 0s - loss: 0.4910 - accuracy:
0.9011
Epoch 16: val_loss did not improve from 0.55465
1250/1250 [==============================] - 24s 19ms/step - loss: 0.4910 -
accuracy: 0.9011 - val_loss: 0.5557 - val_accuracy: 0.8919 - lr: 1.0000e-04
Epoch 17/30
```

```
1249/1250 [============================>.] - ETA: 0s - loss: 0.4817 - accuracy:
0.9028
Epoch 17: val_loss improved from 0.55465 to 0.54566, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 24s 19ms/step - loss: 0.4818 -
accuracy: 0.9028 - val_loss: 0.5457 - val_accuracy: 0.8945 - lr: 1.0000e-04
Epoch 18/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.4604 - accuracy:
0.9072
Epoch 18: val_loss did not improve from 0.54566
1250/1250 [==============================] - 23s 19ms/step - loss: 0.4605 -
accuracy: 0.9071 - val_loss: 0.5500 - val_accuracy: 0.8946 - lr: 1.0000e-04
Epoch 19/30
1248/1250 [============================>.] - ETA: 0s - loss: 0.4589 - accuracy:
0.9078
Epoch 19: val_loss improved from 0.54566 to 0.54361, saving model to
../models/05_model_t_tl_feat_ext_rmsprop_classifier.h5
1250/1250 [==============================] - 23s 19ms/step - loss: 0.4596 -
accuracy: 0.9078 - val_loss: 0.5436 - val_accuracy: 0.8923 - lr: 1.0000e-04
Epoch 20/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.4455 - accuracy:
0.9092
Epoch 20: val_loss did not improve from 0.54361
1250/1250 [==============================] - 23s 18ms/step - loss: 0.4453 -
accuracy: 0.9092 - val_loss: 0.5539 - val_accuracy: 0.8930 - lr: 1.0000e-04
Epoch 21/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.4454 - accuracy:
0.9116
Epoch 21: val_loss did not improve from 0.54361
1250/1250 [==============================] - 23s 19ms/step - loss: 0.4455 -
accuracy: 0.9116 - val_loss: 0.5487 - val_accuracy: 0.8948 - lr: 1.0000e-04
Epoch 22/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.4343 - accuracy:
0.9141
Epoch 22: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.

Epoch 22: val_loss did not improve from 0.54361
1250/1250 [==============================] - 23s 19ms/step - loss: 0.4342 -
accuracy: 0.9141 - val_loss: 0.5552 - val_accuracy: 0.8945 - lr: 1.0000e-04
Epoch 23/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.4082 - accuracy:
0.9195
Epoch 23: val_loss did not improve from 0.54361
1250/1250 [==============================] - 23s 19ms/step - loss: 0.4087 -
accuracy: 0.9194 - val_loss: 0.5511 - val_accuracy: 0.8964 - lr: 1.0000e-05
Epoch 24/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.4066 - accuracy:
0.9208
```

```
Epoch 24: val_loss did not improve from 0.54361
1250/1250 [==============================] - 23s 19ms/step - loss: 0.4070 -
accuracy: 0.9207 - val_loss: 0.5491 - val_accuracy: 0.8974 - lr: 1.0000e-05
Epoch 25/30
1249/1250 [============================>.] - ETA: 0s - loss: 0.4017 - accuracy:
0.9206
Epoch 25: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Restoring model weights from the end of the best epoch: 19.

Epoch 25: val_loss did not improve from 0.54361
1250/1250 [==============================] - 23s 19ms/step - loss: 0.4018 -
accuracy: 0.9205 - val_loss: 0.5497 - val_accuracy: 0.8970 - lr: 1.0000e-05
Epoch 25: early stopping
```

Save Model History

[11]:
```
with open("../history/05_model_t_tl_feat_ext_rmsprop_classifier.pkl", "wb") as␣
↪file:
    pickle.dump(history.history, file)
```

Classifier Evaluation

[12]:
```
val_loss, val_acc = classifier.evaluate(val_features, val_labels)
print(f'Classifier Validation Loss: {val_loss:.2f}')
print(f'Classifier Validation Accuracy: {val_acc:.2%}')
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.5436 -
accuracy: 0.8923
Classifier Validation Loss: 0.54
Classifier Validation Accuracy: 89.23%
```

Classifier Training Visualization

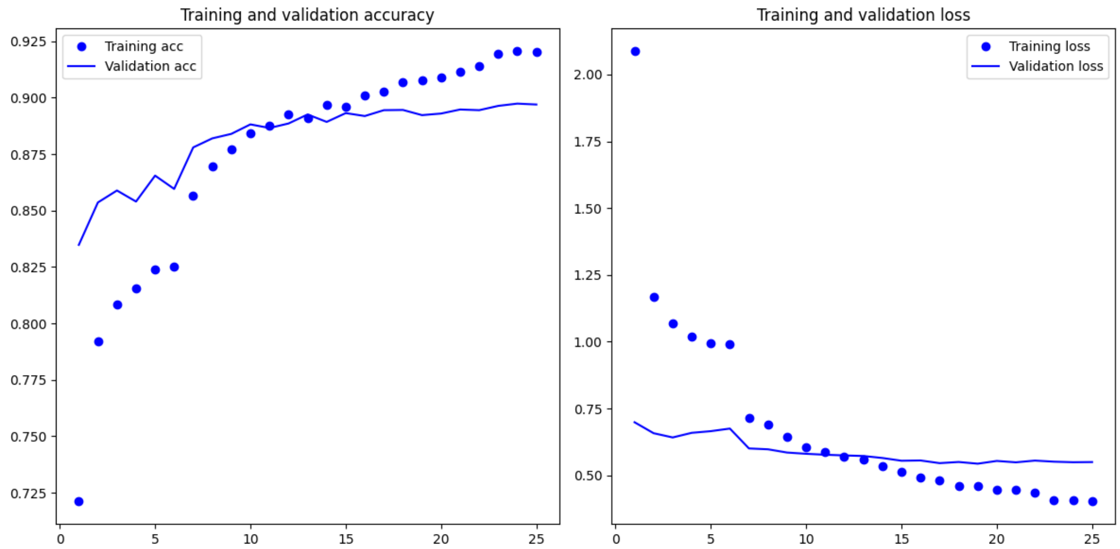
[13]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
```

```
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

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



- Analyzing the training and validation, accuracy and loss over the epochs:
  - We see that the model begins overfitting slightly after the **16th** epoch.
  - The validation accuracy stops improving significantly after the **17th** epoch while the training accuracy keeps improving.
  - The validation loss stops improving significantly after the **17th** epoch while the training loss keeps improving.
  - The best model, based on validation loss, is saved on the **19th** epoch.
  - The training stops after the **25th** epoch because of the **Early Stopping** callback.

Building the Full Model with the VGG16 Convolutional Base and the Classifier

[14]:
```
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = keras.applications.vgg16.preprocess_input(inputs)
x = conv_base(x)
outputs = classifier(x)
model = keras.Model(inputs, outputs)
```

```python
model.compile(
    loss=loss_function,
    optimizer=optimizer,
    metrics=["accuracy"])

model.save('../models/05_model_t_tl_feat_ext_rmsprop_model.h5')
model.summary()
```

```
Model: "model_1"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 128, 128, 3)]     0

 tf.__operators__.getitem (  (None, 128, 128, 3)       0
 SlicingOpLambda)

 tf.nn.bias_add (TFOpLambda  (None, 128, 128, 3)       0
 )

 vgg16 (Functional)          (None, 4, 4, 512)         14714688

 model (Functional)          (None, 10)                4199946

=================================================================
Total params: 18914634 (72.15 MB)
Trainable params: 4199946 (16.02 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

Model Testing

[15]:
```python
test_labels = []
test_predictions = []
test_probabilities = []

for images, labels in test_dataset:
    test_labels.extend(labels.numpy())
    predictions = model.predict(images)
    test_predictions.extend(np.argmax(predictions, axis=-1))
    test_probabilities.extend(predictions)

test_labels = np.array(test_labels)
test_predictions = np.array(test_predictions)
test_probabilities = np.array(test_probabilities)
```

```
2/2 [==============================] - 1s 623ms/step
```

```
2/2 [==============================] - 1s 622ms/step
2/2 [==============================] - 1s 624ms/step
2/2 [==============================] - 1s 624ms/step
2/2 [==============================] - 1s 623ms/step
2/2 [==============================] - 1s 653ms/step
2/2 [==============================] - 1s 623ms/step
2/2 [==============================] - 1s 627ms/step
2/2 [==============================] - 1s 632ms/step
2/2 [==============================] - 1s 676ms/step
2/2 [==============================] - 2s 742ms/step
2/2 [==============================] - 1s 740ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 701ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 690ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 685ms/step
```

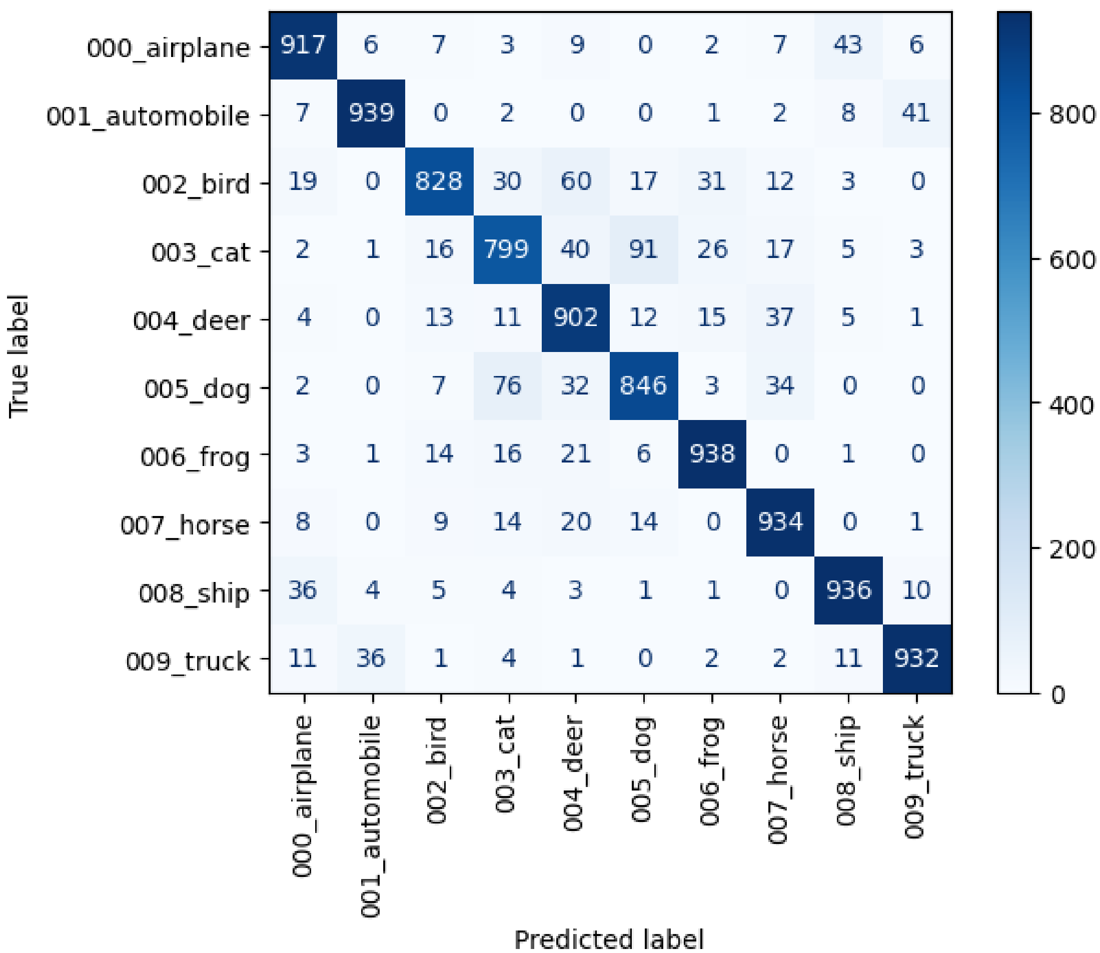
```
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 725ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 705ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 678ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 680ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 679ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 691ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 709ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 719ms/step
2/2 [==============================] - 1s 714ms/step
2/2 [==============================] - 1s 713ms/step
```

```
2/2 [==============================] - 1s 708ms/step
2/2 [==============================] - 1s 707ms/step
2/2 [==============================] - 1s 717ms/step
2/2 [==============================] - 1s 704ms/step
2/2 [==============================] - 1s 710ms/step
2/2 [==============================] - 1s 705ms/step
2/2 [==============================] - 1s 742ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 689ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 706ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 702ms/step
2/2 [==============================] - 1s 714ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 694ms/step
2/2 [==============================] - 1s 713ms/step
2/2 [==============================] - 1s 751ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 715ms/step
2/2 [==============================] - 1s 708ms/step
2/2 [==============================] - 1s 712ms/step
2/2 [==============================] - 1s 714ms/step
2/2 [==============================] - 1s 715ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 692ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 717ms/step
```

```
2/2 [==============================] - 1s 718ms/step
2/2 [==============================] - 1s 727ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
1/1 [==============================] - 0s 422ms/step
```

Confusion Matrix

[16]:
```python
cm = confusion_matrix(test_labels, test_predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap=plt.cm.Blues, xticks_rotation=90)
plt.show()
```



- Looking at the confusion matrix, we see that:
  - The model has a hard time distinguishing the categories 003_cat and 005_dog.
  - The model has a very low performance on the category 003_cat.
  - The model performs better on the vehicle categories than on the animal categories.
  - The model has a below average performance on the categories 005_dog and 002_bird, in which we see a very high false positive rate.
  - The model also has a hard time distinguishing between some other categories but the deviation is not as significant.

  - The model has an above average performance on the categories 000_airplane, 001_automobile, 006_frog, 008_ship and 009_truck.
  - **Basically, the model has the same error distribution but with higher accuracy**.

ROC Curve Analysis

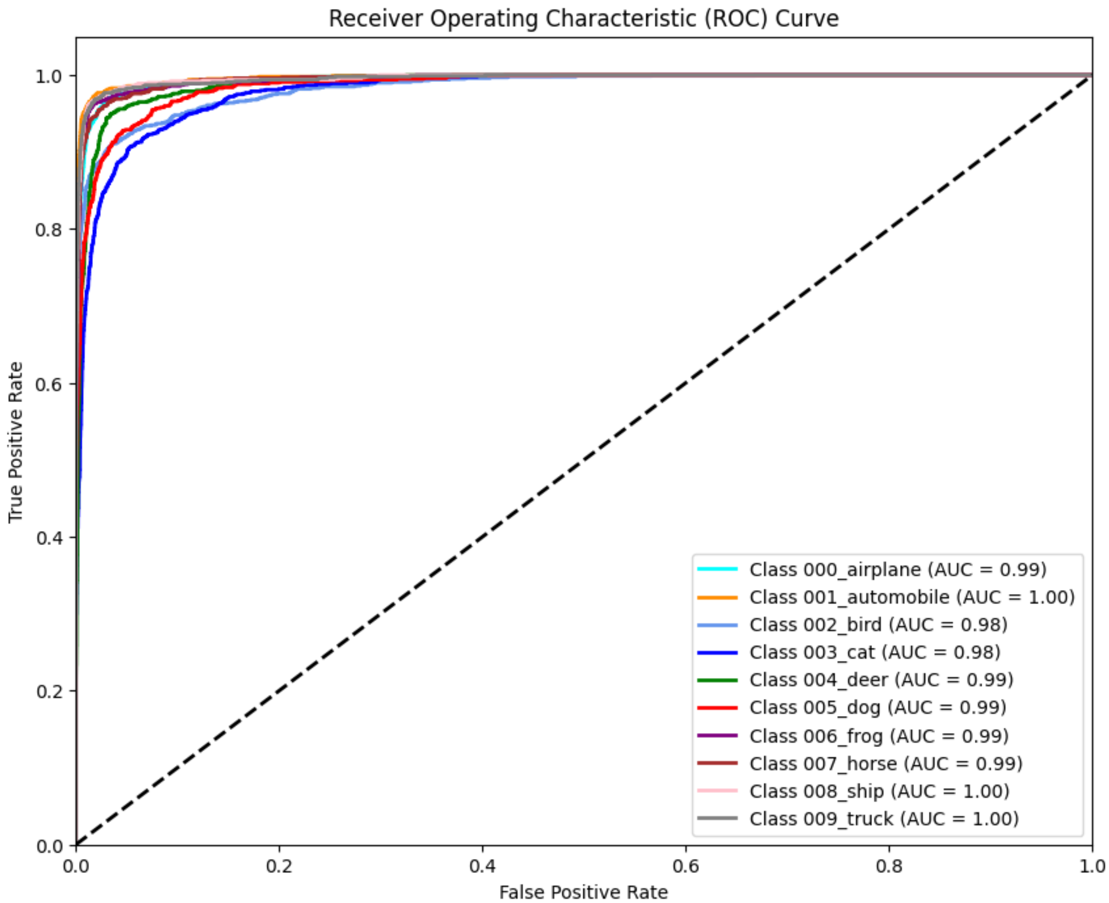
[17]:
```
test_labels_bin = label_binarize(test_labels, classes=range(NUM_CLASSES))

false_positive_rate = dict()
true_positive_rate = dict()
roc_auc = dict()

for i in range(NUM_CLASSES):
    false_positive_rate[i], true_positive_rate[i], _ = roc_curve(test_labels_bin[:, i], test_probabilities[:, i])
    roc_auc[i] = auc(false_positive_rate[i], true_positive_rate[i])

plt.figure(figsize=(10, 8))
colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'blue', 'green', 'red', 'purple', 'brown', 'pink', 'grey'])
for i, color in zip(range(NUM_CLASSES), colors):
    plt.plot(false_positive_rate[i], true_positive_rate[i], color=color, lw=2, label=f'Class {class_names[i]} (AUC = {roc_auc[i]:.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



- Looking at the ROC curve:
  - We see that the model has a good performance on the ROC curve for most categories.
  - The categories 003_cat and 002_bird have the worst AUC (Area Under Curve) performance.
  - The other categories have the same performance but with higher AUC.
  - The category 001_automobile, 008_ship and 009_truck has the best AUC performance.
  - The overall AUC performance increases as the false positive rate decreases and the true positive rate increases.
  - **A perfect AUC of 1.0 would mean that the model classifies all images either true positives or true negatives**.

Performance Metrics

- **Accuracy** is the proportion of correctly predicted instances out of the total instances.
- **Precision** is the ratio of true positive predictions to the total predicted positives. Macro precision calculates this for each class independently and then averages them.

- **Weighted precision** calculates the precision for each class, then averages them, weighted by the number of true instances for each class.

- **Recall** is the ratio of true positive predictions to the total actual positives. Macro recall calculates this for each class independently and then averages them.

- **Weighted recall** calculates the recall for each class, then averages them, weighted by the number of true instances for each class.

- The **F1-score** is the harmonic mean of precision and recall. Macro F1-score calculates this for each class independently and then averages them.

- **Weighted F1-score** calculates the F1-score for each class, then averages them, weighted by the number of true instances for each class.

[18]:
```
acc = accuracy_score(y_true =  test_labels, y_pred = test_predictions)
print(f'Accuracy : {np.round(acc*100,2)}%')
precision = precision_score(y_true =  test_labels, y_pred = test_predictions,␣
 ↪average='macro')
print(f'Precision - Macro: {np.round(precision*100,2)}%')
recall = recall_score(y_true =  test_labels, y_pred = test_predictions,␣
 ↪average='macro')
print(f'Recall - Macro: {np.round(recall*100,2)}%')
f1 = f1_score(y_true =  test_labels, y_pred = test_predictions, average='macro')
print(f'F1-score - Macro: {np.round(f1*100,2)}%')
precision = precision_score(y_true =  test_labels, y_pred = test_predictions,␣
 ↪average='weighted')
print(f'Precision - Weighted: {np.round(precision*100,2)}%')
recall = recall_score(y_true =  test_labels, y_pred = test_predictions,␣
 ↪average='weighted')
print(f'Recall - Weighted: {np.round(recall*100,2)}%')
f1 = f1_score(y_true =  test_labels, y_pred = test_predictions,␣
 ↪average='weighted')
print(f'F1-score - Weighted: {np.round(f1*100,2)}%')
```

```
Accuracy : 89.71%
Precision - Macro: 89.76%
Recall - Macro: 89.71%
F1-score - Macro: 89.69%
Precision - Weighted: 89.76%
Recall - Weighted: 89.71%
F1-score - Weighted: 89.69%
```

- **Since the dataset is balanced, the** MACRO** average is a good metric to evaluate the model.**

2 Conclusion

2.0.1 Summary

- Before this notebook:
  - We resized our images to be the 128 x 128 x 3.
  - Our reasoning was up scaling by a factor of 4.
- In this notebook:
  - We extracted feature maps from our train and validation datasets using the convolutional base of the VGG16.
  - We trained a classifier with those extracted features:
    * We used the Root Mean Squared Propagation (RMSProp) optimizer with an initial learning rate of 0.001.
    * We kept the same 30 epochs with a batch size of 64.
  - We evaluated the classifier on the validation dataset:
    * Overfitting was observed after **16 epochs**, but the best classifier was saved at the **19th epoch**.
    * Training was intended for 30 epochs but stopped early due to the **Early Stopping** callback.
  - We then joined the VGG16 Convolutional Base with our Classifier
  - We tested the resulting model on the test set:
    * We evaluated the model using a confusion matrix to analyze its performance on each category.
    * We evaluated the model using ROC curves for a deeper performance analysis.
    * The model achieved an accuracy of 89.71% on the test set, which was a good improvement.

2.0.2 Future Work

- In the next notebook:
  - Implement and train a transfer learning model with the VGG16 convolutional base frozen and a new classifier.
  - Experiment with **custom** data augmentation to improve classifier generalization
  - We will use 50 Epochs with a batch size of 64.
  - Test the model performance.