06_model_t_tl_data_augm_adam

June 10, 2024

---

1 Model T - Transfer Learning, Data Augmentation, Adaptive Moment Estimation (Adam)

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

- **64** Batch size.
- Build a **full model** with **VGG16** convolutional base and a **Classifier** and **Train it**.
  - **Data Augmentation Pipeline**
    * Random **Horizontal** Flip
    * Random Rotation **5%**
    * Random Zoom **5%**
    * Random Contrast **5%**
    * Random Brightness **5%**
  - **VGG16** Convolutional Base **(Frozen)** will produce a **4 x 4 x 512** Tensor.
  - **Classifier**:
    * Adaptive Moment Estimation **(Adam)** optimizer.
    * **0.001** Initial Learning rate.
    * **Sparse Categorical Cross-Entropy** loss function.
    * **Reduce Learning Rate on Plateau** callback with a **0.1** factor and **3** patience.
    * **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 **0.3** rate after the Flatten layer and **0.5** after the Dense layer.
    * **L2** regularization with **0.00001** rate on the Dense layers.
  - **4 199 946** Trainable Parameters.
  - **50 Epochs**.
- **Evaluate** the model on the **Validation Set**.
- **Test** the model on the **Test Set**.

Imports and Setup

[1]:
```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

```python
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
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.

Data Augmentation Pipeline

[4]:
```
data_augmentation = keras.Sequential(
    [
        # keras.layers.RandomCrop(height=96, width=96), # This layer is␣
↪commented out because it didn't improve the model performance.
        keras.layers.RandomFlip("horizontal"),
        # keras.layers.RandomTranslation(0.1, 0.1), # This layer is commented␣
↪out because it didn't improve the model performance.
        keras.layers.RandomRotation(0.05),
        keras.layers.RandomZoom(0.05),
        keras.layers.RandomContrast(0.05),
        keras.layers.RandomBrightness(0.05),
        # keras.layers.GaussianNoise(0.1), # This layer is commented out␣
↪because it didn't improve the model performance.
    ]
)
```

Loading the VGG16 Model

[5]:
```python
from tensorflow.keras.applications.vgg16 import VGG16
conv_base = VGG16(weights='imagenet', include_top=False)
conv_base.trainable = False
conv_base.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 3s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None, None, 3)]   0

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Model Arquitecture

[6]:
```
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)
x = keras.applications.vgg16.preprocess_input(x)
x = conv_base(x)
x = layers.Flatten()(x)
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)
model = keras.Model(inputs, outputs)
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 128, 128, 3)]     0

 sequential (Sequential)     (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, None, None, 512)   14714688

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

Model Compilation

[7]:
```
initial_learning_rate = 0.001
optimizer = optimizers.Adam(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/
 ↪06_model_t_tl_data_augm_adam.h5', save_best_only=True, monitor='val_loss',␣
 ↪verbose=1)

callbacks = [lr_scheduler, early_stopping, save_best_model]

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

Model Training

[8]:
```
history = model.fit(
    train_dataset,
    epochs=50,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/50
628/628 [==============================] - ETA: 0s - loss: 1.8919 - accuracy:
0.6648
Epoch 1: val_loss improved from inf to 0.70400, saving model to
../models/06_model_t_tl_data_augm_adam.h5
```

```
628/628 [==============================] - 1100s 2s/step - loss: 1.8919 -
accuracy: 0.6648 - val_loss: 0.7040 - val_accuracy: 0.8131 - lr: 0.0010
Epoch 2/50

/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(

628/628 [==============================] - ETA: 0s - loss: 0.9934 - accuracy:
0.7368
Epoch 2: val_loss improved from 0.70400 to 0.62587, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1103s 2s/step - loss: 0.9934 -
accuracy: 0.7368 - val_loss: 0.6259 - val_accuracy: 0.8512 - lr: 0.0010
Epoch 3/50
628/628 [==============================] - ETA: 0s - loss: 0.9315 - accuracy:
0.7602
Epoch 3: val_loss did not improve from 0.62587
628/628 [==============================] - 1100s 2s/step - loss: 0.9315 -
accuracy: 0.7602 - val_loss: 0.6303 - val_accuracy: 0.8632 - lr: 0.0010
Epoch 4/50
628/628 [==============================] - ETA: 0s - loss: 0.9117 - accuracy:
0.7771
Epoch 4: val_loss did not improve from 0.62587
628/628 [==============================] - 1100s 2s/step - loss: 0.9117 -
accuracy: 0.7771 - val_loss: 0.6289 - val_accuracy: 0.8611 - lr: 0.0010
Epoch 5/50
628/628 [==============================] - ETA: 0s - loss: 0.9022 - accuracy:
0.7857
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.

Epoch 5: val_loss did not improve from 0.62587
628/628 [==============================] - 1101s 2s/step - loss: 0.9022 -
accuracy: 0.7857 - val_loss: 0.6531 - val_accuracy: 0.8622 - lr: 0.0010
Epoch 6/50
628/628 [==============================] - ETA: 0s - loss: 0.8041 - accuracy:
0.8132
Epoch 6: val_loss improved from 0.62587 to 0.60767, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.8041 -
accuracy: 0.8132 - val_loss: 0.6077 - val_accuracy: 0.8756 - lr: 1.0000e-04
Epoch 7/50
628/628 [==============================] - ETA: 0s - loss: 0.7573 - accuracy:
0.8272
Epoch 7: val_loss improved from 0.60767 to 0.59135, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1098s 2s/step - loss: 0.7573 -
```

```
accuracy: 0.8272 - val_loss: 0.5913 - val_accuracy: 0.8797 - lr: 1.0000e-04
Epoch 8/50
628/628 [==============================] - ETA: 0s - loss: 0.7293 - accuracy:
0.8316
Epoch 8: val_loss improved from 0.59135 to 0.58296, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.7293 -
accuracy: 0.8316 - val_loss: 0.5830 - val_accuracy: 0.8830 - lr: 1.0000e-04
Epoch 9/50
628/628 [==============================] - ETA: 0s - loss: 0.7097 - accuracy:
0.8367
Epoch 9: val_loss improved from 0.58296 to 0.56811, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1098s 2s/step - loss: 0.7097 -
accuracy: 0.8367 - val_loss: 0.5681 - val_accuracy: 0.8854 - lr: 1.0000e-04
Epoch 10/50
628/628 [==============================] - ETA: 0s - loss: 0.7005 - accuracy:
0.8353
Epoch 10: val_loss improved from 0.56811 to 0.56243, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1096s 2s/step - loss: 0.7005 -
accuracy: 0.8353 - val_loss: 0.5624 - val_accuracy: 0.8836 - lr: 1.0000e-04
Epoch 11/50
628/628 [==============================] - ETA: 0s - loss: 0.6802 - accuracy:
0.8432
Epoch 11: val_loss improved from 0.56243 to 0.55154, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.6802 -
accuracy: 0.8432 - val_loss: 0.5515 - val_accuracy: 0.8867 - lr: 1.0000e-04
Epoch 12/50
628/628 [==============================] - ETA: 0s - loss: 0.6681 - accuracy:
0.8424
Epoch 12: val_loss improved from 0.55154 to 0.54052, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.6681 -
accuracy: 0.8424 - val_loss: 0.5405 - val_accuracy: 0.8914 - lr: 1.0000e-04
Epoch 13/50
628/628 [==============================] - ETA: 0s - loss: 0.6547 - accuracy:
0.8455
Epoch 13: val_loss improved from 0.54052 to 0.53324, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.6547 -
accuracy: 0.8455 - val_loss: 0.5332 - val_accuracy: 0.8890 - lr: 1.0000e-04
Epoch 14/50
628/628 [==============================] - ETA: 0s - loss: 0.6449 - accuracy:
0.8460
Epoch 14: val_loss improved from 0.53324 to 0.52337, saving model to
../models/06_model_t_tl_data_augm_adam.h5
```

```
628/628 [==============================] - 1097s 2s/step - loss: 0.6449 -
accuracy: 0.8460 - val_loss: 0.5234 - val_accuracy: 0.8899 - lr: 1.0000e-04
Epoch 15/50
628/628 [==============================] - ETA: 0s - loss: 0.6244 - accuracy:
0.8512
Epoch 15: val_loss improved from 0.52337 to 0.51077, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1098s 2s/step - loss: 0.6244 -
accuracy: 0.8512 - val_loss: 0.5108 - val_accuracy: 0.8932 - lr: 1.0000e-04
Epoch 16/50
628/628 [==============================] - ETA: 0s - loss: 0.6127 - accuracy:
0.8522
Epoch 16: val_loss improved from 0.51077 to 0.50336, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.6127 -
accuracy: 0.8522 - val_loss: 0.5034 - val_accuracy: 0.8918 - lr: 1.0000e-04
Epoch 17/50
628/628 [==============================] - ETA: 0s - loss: 0.6033 - accuracy:
0.8552
Epoch 17: val_loss improved from 0.50336 to 0.49610, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.6033 -
accuracy: 0.8552 - val_loss: 0.4961 - val_accuracy: 0.8931 - lr: 1.0000e-04
Epoch 18/50
628/628 [==============================] - ETA: 0s - loss: 0.5880 - accuracy:
0.8597
Epoch 18: val_loss improved from 0.49610 to 0.48448, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.5880 -
accuracy: 0.8597 - val_loss: 0.4845 - val_accuracy: 0.8953 - lr: 1.0000e-04
Epoch 19/50
628/628 [==============================] - ETA: 0s - loss: 0.5765 - accuracy:
0.8594
Epoch 19: val_loss improved from 0.48448 to 0.48101, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.5765 -
accuracy: 0.8594 - val_loss: 0.4810 - val_accuracy: 0.8943 - lr: 1.0000e-04
Epoch 20/50
628/628 [==============================] - ETA: 0s - loss: 0.5750 - accuracy:
0.8594
Epoch 20: val_loss did not improve from 0.48101
628/628 [==============================] - 1097s 2s/step - loss: 0.5750 -
accuracy: 0.8594 - val_loss: 0.4810 - val_accuracy: 0.8949 - lr: 1.0000e-04
Epoch 21/50
628/628 [==============================] - ETA: 0s - loss: 0.5538 - accuracy:
0.8659
Epoch 21: val_loss improved from 0.48101 to 0.47392, saving model to
../models/06_model_t_tl_data_augm_adam.h5
```

```
628/628 [==============================] - 1097s 2s/step - loss: 0.5538 -
accuracy: 0.8659 - val_loss: 0.4739 - val_accuracy: 0.8958 - lr: 1.0000e-04
Epoch 22/50
628/628 [==============================] - ETA: 0s - loss: 0.5550 - accuracy:
0.8659
Epoch 22: val_loss improved from 0.47392 to 0.47360, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.5550 -
accuracy: 0.8659 - val_loss: 0.4736 - val_accuracy: 0.8935 - lr: 1.0000e-04
Epoch 23/50
628/628 [==============================] - ETA: 0s - loss: 0.5436 - accuracy:
0.8674
Epoch 23: val_loss improved from 0.47360 to 0.47184, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.5436 -
accuracy: 0.8674 - val_loss: 0.4718 - val_accuracy: 0.8942 - lr: 1.0000e-04
Epoch 24/50
628/628 [==============================] - ETA: 0s - loss: 0.5363 - accuracy:
0.8691
Epoch 24: val_loss improved from 0.47184 to 0.46441, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1096s 2s/step - loss: 0.5363 -
accuracy: 0.8691 - val_loss: 0.4644 - val_accuracy: 0.8965 - lr: 1.0000e-04
Epoch 25/50
628/628 [==============================] - ETA: 0s - loss: 0.5378 - accuracy:
0.8681
Epoch 25: val_loss did not improve from 0.46441
628/628 [==============================] - 1097s 2s/step - loss: 0.5378 -
accuracy: 0.8681 - val_loss: 0.4658 - val_accuracy: 0.8983 - lr: 1.0000e-04
Epoch 26/50
628/628 [==============================] - ETA: 0s - loss: 0.5275 - accuracy:
0.8709
Epoch 26: val_loss improved from 0.46441 to 0.46372, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.5275 -
accuracy: 0.8709 - val_loss: 0.4637 - val_accuracy: 0.8981 - lr: 1.0000e-04
Epoch 27/50
628/628 [==============================] - ETA: 0s - loss: 0.5173 - accuracy:
0.8755
Epoch 27: val_loss improved from 0.46372 to 0.46027, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1098s 2s/step - loss: 0.5173 -
accuracy: 0.8755 - val_loss: 0.4603 - val_accuracy: 0.8994 - lr: 1.0000e-04
Epoch 28/50
628/628 [==============================] - ETA: 0s - loss: 0.5132 - accuracy:
0.8758
Epoch 28: val_loss improved from 0.46027 to 0.45799, saving model to
../models/06_model_t_tl_data_augm_adam.h5
```

```
628/628 [==============================] - 1098s 2s/step - loss: 0.5132 -
accuracy: 0.8758 - val_loss: 0.4580 - val_accuracy: 0.8987 - lr: 1.0000e-04
Epoch 29/50
628/628 [==============================] - ETA: 0s - loss: 0.5071 - accuracy:
0.8765
Epoch 29: val_loss improved from 0.45799 to 0.45107, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.5071 -
accuracy: 0.8765 - val_loss: 0.4511 - val_accuracy: 0.8996 - lr: 1.0000e-04
Epoch 30/50
628/628 [==============================] - ETA: 0s - loss: 0.4965 - accuracy:
0.8804
Epoch 30: val_loss did not improve from 0.45107
628/628 [==============================] - 1096s 2s/step - loss: 0.4965 -
accuracy: 0.8804 - val_loss: 0.4523 - val_accuracy: 0.9002 - lr: 1.0000e-04
Epoch 31/50
628/628 [==============================] - ETA: 0s - loss: 0.4911 - accuracy:
0.8817
Epoch 31: val_loss did not improve from 0.45107
628/628 [==============================] - 1096s 2s/step - loss: 0.4911 -
accuracy: 0.8817 - val_loss: 0.4537 - val_accuracy: 0.8997 - lr: 1.0000e-04
Epoch 32/50
628/628 [==============================] - ETA: 0s - loss: 0.4875 - accuracy:
0.8825
Epoch 32: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.

Epoch 32: val_loss improved from 0.45107 to 0.45100, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1096s 2s/step - loss: 0.4875 -
accuracy: 0.8825 - val_loss: 0.4510 - val_accuracy: 0.9007 - lr: 1.0000e-04
Epoch 33/50
628/628 [==============================] - ETA: 0s - loss: 0.4739 - accuracy:
0.8858
Epoch 33: val_loss improved from 0.45100 to 0.44847, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4739 -
accuracy: 0.8858 - val_loss: 0.4485 - val_accuracy: 0.9026 - lr: 1.0000e-05
Epoch 34/50
628/628 [==============================] - ETA: 0s - loss: 0.4690 - accuracy:
0.8881
Epoch 34: val_loss improved from 0.44847 to 0.44708, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1098s 2s/step - loss: 0.4690 -
accuracy: 0.8881 - val_loss: 0.4471 - val_accuracy: 0.9014 - lr: 1.0000e-05
Epoch 35/50
628/628 [==============================] - ETA: 0s - loss: 0.4695 - accuracy:
0.8881
Epoch 35: val_loss did not improve from 0.44708
```

```
628/628 [==============================] - 1097s 2s/step - loss: 0.4695 -
accuracy: 0.8881 - val_loss: 0.4477 - val_accuracy: 0.9019 - lr: 1.0000e-05
Epoch 36/50
628/628 [==============================] - ETA: 0s - loss: 0.4677 - accuracy:
0.8882
Epoch 36: val_loss improved from 0.44708 to 0.44701, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4677 -
accuracy: 0.8882 - val_loss: 0.4470 - val_accuracy: 0.9020 - lr: 1.0000e-05
Epoch 37/50
628/628 [==============================] - ETA: 0s - loss: 0.4609 - accuracy:
0.8891
Epoch 37: val_loss improved from 0.44701 to 0.44588, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4609 -
accuracy: 0.8891 - val_loss: 0.4459 - val_accuracy: 0.9020 - lr: 1.0000e-05
Epoch 38/50
628/628 [==============================] - ETA: 0s - loss: 0.4578 - accuracy:
0.8912
Epoch 38: val_loss improved from 0.44588 to 0.44477, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4578 -
accuracy: 0.8912 - val_loss: 0.4448 - val_accuracy: 0.9018 - lr: 1.0000e-05
Epoch 39/50
628/628 [==============================] - ETA: 0s - loss: 0.4619 - accuracy:
0.8895
Epoch 39: val_loss improved from 0.44477 to 0.44417, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4619 -
accuracy: 0.8895 - val_loss: 0.4442 - val_accuracy: 0.9031 - lr: 1.0000e-05
Epoch 40/50
628/628 [==============================] - ETA: 0s - loss: 0.4586 - accuracy:
0.8910
Epoch 40: val_loss improved from 0.44417 to 0.44393, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1098s 2s/step - loss: 0.4586 -
accuracy: 0.8910 - val_loss: 0.4439 - val_accuracy: 0.9027 - lr: 1.0000e-05
Epoch 41/50
628/628 [==============================] - ETA: 0s - loss: 0.4583 - accuracy:
0.8923
Epoch 41: val_loss improved from 0.44393 to 0.44286, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4583 -
accuracy: 0.8923 - val_loss: 0.4429 - val_accuracy: 0.9031 - lr: 1.0000e-05
Epoch 42/50
628/628 [==============================] - ETA: 0s - loss: 0.4559 - accuracy:
0.8910
Epoch 42: val_loss did not improve from 0.44286
```

```
628/628 [==============================] - 1097s 2s/step - loss: 0.4559 -
accuracy: 0.8910 - val_loss: 0.4438 - val_accuracy: 0.9015 - lr: 1.0000e-05
Epoch 43/50
628/628 [==============================] - ETA: 0s - loss: 0.4492 - accuracy:
0.8939
Epoch 43: val_loss improved from 0.44286 to 0.44246, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4492 -
accuracy: 0.8939 - val_loss: 0.4425 - val_accuracy: 0.9011 - lr: 1.0000e-05
Epoch 44/50
628/628 [==============================] - ETA: 0s - loss: 0.4488 - accuracy:
0.8926
Epoch 44: val_loss improved from 0.44246 to 0.44167, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1099s 2s/step - loss: 0.4488 -
accuracy: 0.8926 - val_loss: 0.4417 - val_accuracy: 0.9028 - lr: 1.0000e-05
Epoch 45/50
628/628 [==============================] - ETA: 0s - loss: 0.4502 - accuracy:
0.8940
Epoch 45: val_loss improved from 0.44167 to 0.44163, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4502 -
accuracy: 0.8940 - val_loss: 0.4416 - val_accuracy: 0.9030 - lr: 1.0000e-05
Epoch 46/50
628/628 [==============================] - ETA: 0s - loss: 0.4523 - accuracy:
0.8942
Epoch 46: val_loss improved from 0.44163 to 0.44074, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1102s 2s/step - loss: 0.4523 -
accuracy: 0.8942 - val_loss: 0.4407 - val_accuracy: 0.9028 - lr: 1.0000e-05
Epoch 47/50
628/628 [==============================] - ETA: 0s - loss: 0.4432 - accuracy:
0.8955
Epoch 47: val_loss improved from 0.44074 to 0.44064, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4432 -
accuracy: 0.8955 - val_loss: 0.4406 - val_accuracy: 0.9027 - lr: 1.0000e-05
Epoch 48/50
628/628 [==============================] - ETA: 0s - loss: 0.4442 - accuracy:
0.8952
Epoch 48: val_loss improved from 0.44064 to 0.44031, saving model to
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1098s 2s/step - loss: 0.4442 -
accuracy: 0.8952 - val_loss: 0.4403 - val_accuracy: 0.9041 - lr: 1.0000e-05
Epoch 49/50
628/628 [==============================] - ETA: 0s - loss: 0.4448 - accuracy:
0.8942
Epoch 49: val_loss improved from 0.44031 to 0.43939, saving model to
```

```
../models/06_model_t_tl_data_augm_adam.h5
628/628 [==============================] - 1097s 2s/step - loss: 0.4448 -
accuracy: 0.8942 - val_loss: 0.4394 - val_accuracy: 0.9036 - lr: 1.0000e-05
Epoch 50/50
628/628 [==============================] - ETA: 0s - loss: 0.4418 - accuracy:
0.8955
Epoch 50: val_loss did not improve from 0.43939
628/628 [==============================] - 1097s 2s/step - loss: 0.4418 -
accuracy: 0.8955 - val_loss: 0.4399 - val_accuracy: 0.9031 - lr: 1.0000e-05
```

Save Model History

[9]:
```
with open("../history/06_model_t_tl_data_augm_adam.pkl", "wb") as file:
    pickle.dump(history.history, file)
```

Model Evaluation

[10]:
```
val_loss, val_acc = model.evaluate(validation_dataset)
print(f'Classifier Validation Loss: {val_loss:.2f}')
print(f'Classifier Validation Accuracy: {val_acc:.2%}')
```

```
157/157 [==============================] - 214s 1s/step - loss: 0.4399 -
accuracy: 0.9031
Classifier Validation Loss: 0.44
Classifier Validation Accuracy: 90.31%
```

Model Visualization

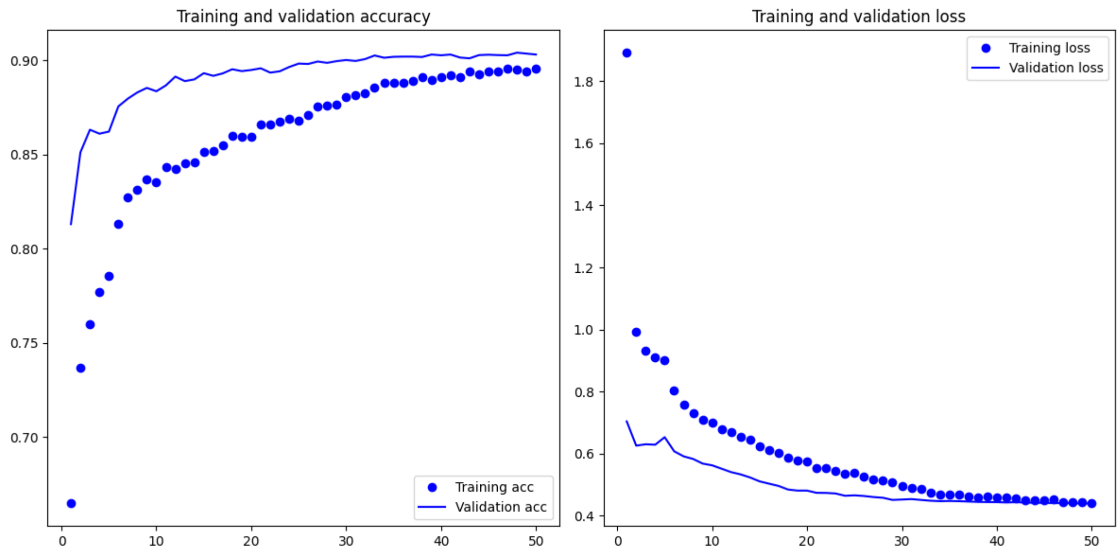
[11]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

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 never really overfits which indicates that we could **train it for more epochs**.
  - The validation accuracy keeps improving throughout the epochs, **never being surpassed by the training accuracy**.
  - The validation loss keeps decreasing throughout the epochs, **never surpassing the training loss**.
  - The best model, based on validation loss, is saved on the **49th** epoch.

Model Testing

[12]:
```
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)
```

```python
test_probabilities = np.array(test_probabilities)
```

```
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 688ms/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
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 683ms/step
```

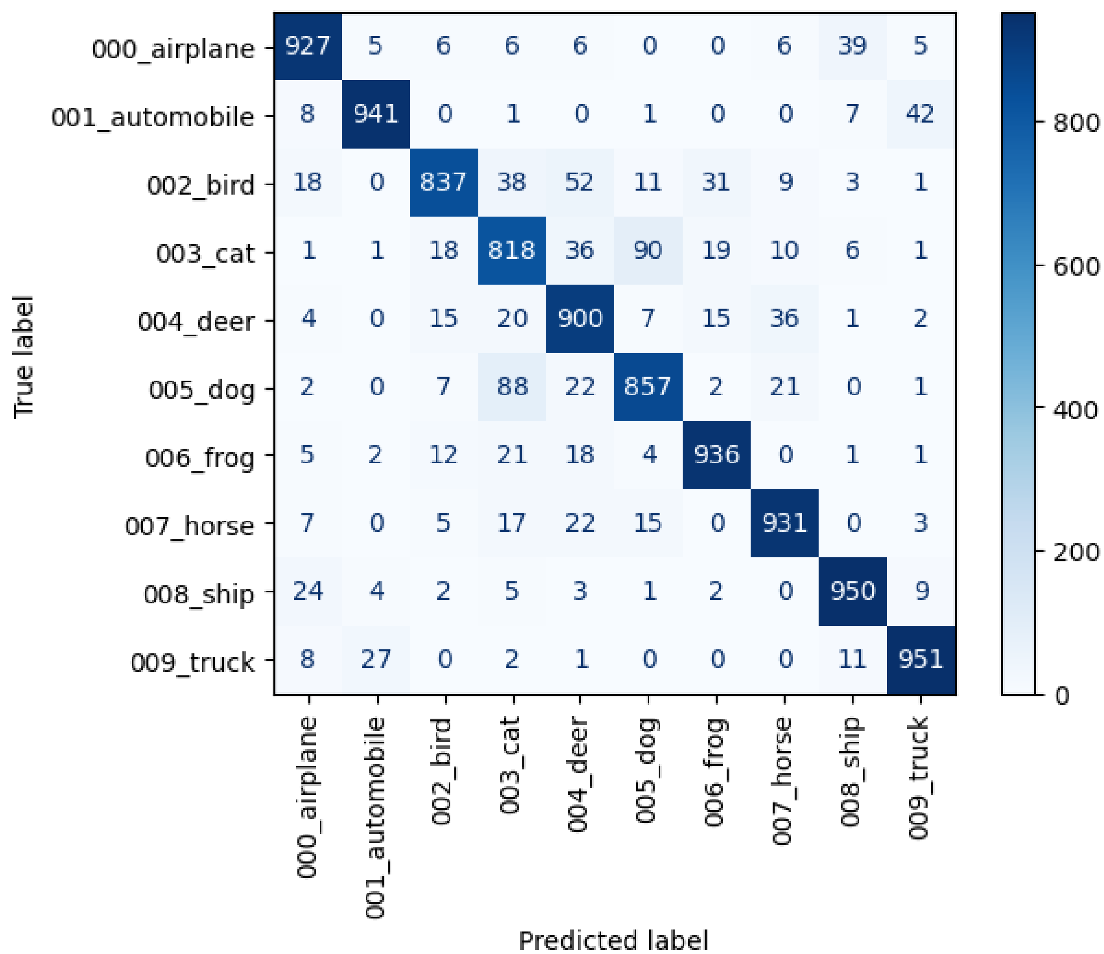
```
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 698ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 688ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
```

```
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 681ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 683ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 687ms/step
```

```
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 685ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 682ms/step
2/2 [==============================] - 1s 684ms/step
2/2 [==============================] - 1s 686ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 687ms/step
2/2 [==============================] - 1s 686ms/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
1/1 [==============================] - 0s 413ms/step
```

Confusion Matrix

[13]:
```
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 still has a hard time distinguishing between the categories 003_cat and 005_dog but with less error.
  - 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 002_bird, 003_cat and 005_dog, 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, 007_horse 008_ship and 009_truck.
  - **Basically, the model has the same error distribution but with higher accuracy**.

ROC Curve Analysis

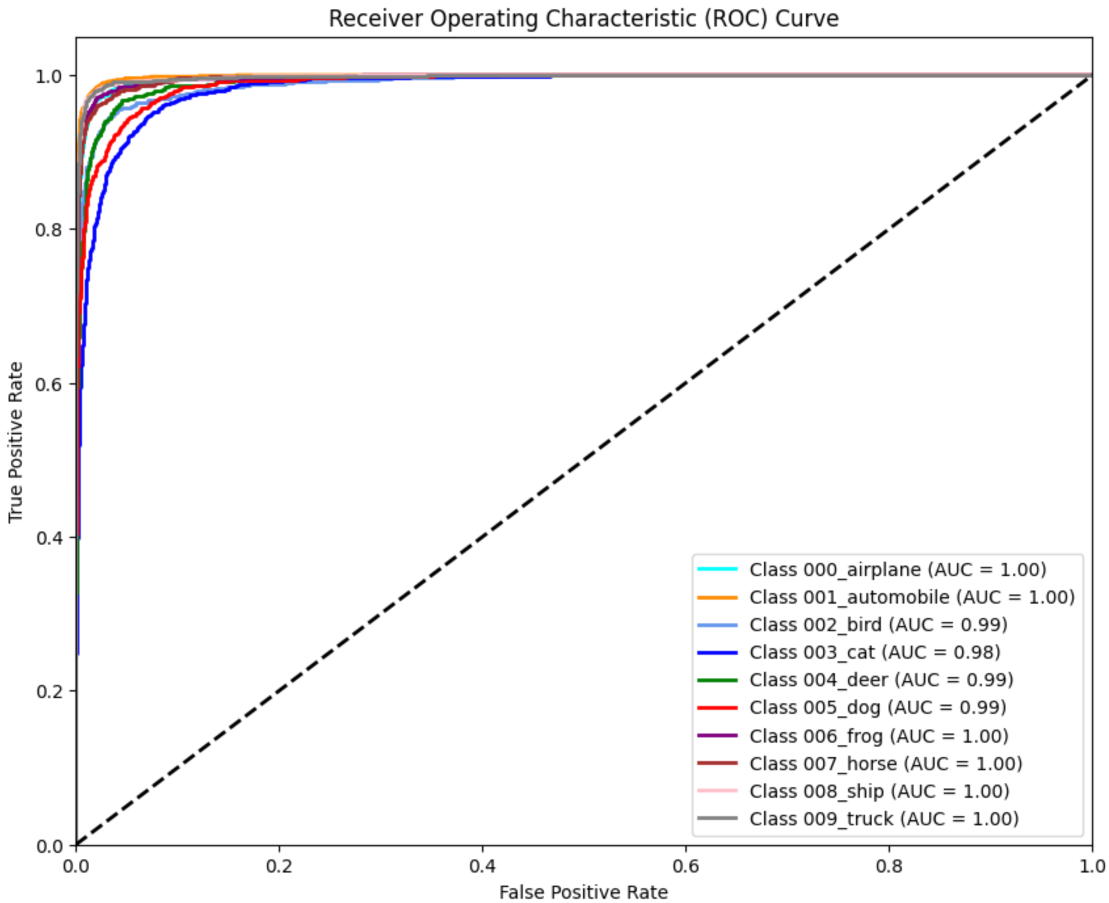
[14]:
```python
test_labels_bin = label_binarize(test_labels, classes=range(NUM_CLASSES))

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 002_bird, 003_cat, 004_deer and 005_dog have the worst AUC (Area Under Curve) performance with 003_cat being the worst.
  - The other categories have a better performance with higher AUC.
  - The categories 000_airplane, 001_automobile, 006_frog, 007_horse, 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.

[15]:
```
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 : 90.48%
Precision - Macro: 90.54%
Recall - Macro: 90.48%
F1-score - Macro: 90.48%
Precision - Weighted: 90.54%
Recall - Weighted: 90.48%
F1-score - Weighted: 90.48%
```

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

2 Conclusion

2.0.1 Summary

- In this notebook:
  - We built a model with the VGG16 convolutional base and a classifier.
    - We used a data augmentation pipeline to increase the dataset size.
      - Random **Horizontal** Flip
      - Random Rotation **5%**
      - Random Zoom **5%**
      - Random Contrast **5%**
      - Random Brightness **5%**
    - We used the VGG16 convolutional base with its weights frozen.
    - We built a classifier:
      - We used the Adaptive Moment Estimation (Adam) optimizer.
      - We used an initial learning rate of 0.001.
    - We built the full model using the VGG16 convolutional base and the classifier.
  - We trained the model for 50 epochs.
  - We evaluated the model on the validation set.
    - The model never overfitted.
    - The best model was saved at the 49th epoch.
  - We tested the 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 90.48% on the test set.

2.0.2 Future Work

- In the next notebook we will:
  - Load this model and fine-tune it:
    - We will unfreeze the 4 last layers of convolutional base of the VGG16.
    - Train the model with a smaller learning rate for 30 more epochs.
  - Test the model performance.