03_model_s_data_augm_l2_adam

June 10, 2024

---

1 Model S - Data Augmentation and L2 Regularization with Adam Optimizer

- **32 x 32 x 3** Image size.
- **64** Batch size.
- 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 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.
- **Data Augmentation Pipeline**
  - Random **Horizontal** Flip
  - Random Rotation **5%**
  - Random Zoom **5%**
  - Random Contrast **5%**
  - Random Brightness **5%**
- 3 **Convolutional** blocks with 2 layers each of **32**, **64** and **128** filters, with **ReLU** activation.
- **Batch Normalization** after each Convolutional layer.
- 3 **MaxPooling** layers with **2 x 2** pool size.
- **3 x 3** Convolutional kernel size.
- **Padding** is **valid**, in this case **1**.
- **4 x 4 x 128** Tensor before the **Flatten** layer.
- **256** 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 and **0.00001** rate on the Convolutional layers.
- **815 018** Trainable Parameters.
- **40** Epochs.
- Evaluate the model on the **Validation** dataset.
- Test the model on the **Test** dataset.

Imports and Setup

[1]:
```python
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
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, models
from keras import regularizers
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
train_dirs = ['../data/train1', '../data/train3', '../data/train4', '../data/
↪train5']
validation_dir = '../data/train2'
test_dir = '../data/test'
```

Create Datasets

[3]:
```python
IMG_SIZE = 32
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)
```

```python
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, 32, 32, 3)
labels batch shape: (64,)
```

- We define the image size of 32 x 32 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]:
```python
data_augmentation = keras.Sequential(
    [
        # keras.layers.RandomCrop(height=24, width=24), # 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),
    ]
)
```

- **We define a data augmentation pipeline to apply to the images**.
- **The pipeline:**
  - Applies horizontal flipping to a random 50% of the images that go through it.
  - Randomly rotates the input images by 5%.

- Randomly zooms the input images by 5%.
  - Randomly adjusts the contrast of the input images by 5%.
  - Randomly adjusts the brightness of the input images by 5%.
- **The following techniques were tested but the model didn't perform well:**
  - RandomCrop: Randomly crops the images along the width and height down to 16 x 16.
  - RandomTranslation: Randomly translates the input images along the width and height by 10%.
- **We will implement custom techniques on the transfer learning model**.

Model Architecture

[5]:
```
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu",␣
↪kernel_regularizer=regularizers.L2(1e-5), padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu",␣
↪kernel_regularizer=regularizers.L2(1e-5), padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu",␣
↪kernel_regularizer=regularizers.L2(1e-5), padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu",␣
↪kernel_regularizer=regularizers.L2(1e-5), padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu",␣
↪kernel_regularizer=regularizers.L2(1e-5), padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu",␣
↪kernel_regularizer=regularizers.L2(1e-5), padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(256, 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 = models.Model(inputs=inputs, outputs=outputs)
model.summary()
```

```
Model: "model"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0

 sequential (Sequential)     (None, 32, 32, 3)         0

 rescaling (Rescaling)       (None, 32, 32, 3)         0

 conv2d (Conv2D)             (None, 32, 32, 32)        896

 batch_normalization (Batch  (None, 32, 32, 32)        128
 Normalization)

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 batch_normalization_1 (Bat  (None, 32, 32, 32)        128
 chNormalization)

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 batch_normalization_2 (Bat  (None, 16, 16, 64)        256
 chNormalization)

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 batch_normalization_3 (Bat  (None, 16, 16, 64)        256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_4 (Conv2D)           (None, 8, 8, 128)         73856

 batch_normalization_4 (Bat  (None, 8, 8, 128)         512
 chNormalization)

 conv2d_5 (Conv2D)           (None, 8, 8, 128)         147584

 batch_normalization_5 (Bat  (None, 8, 8, 128)         512
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)
```

```
 flatten (Flatten)           (None, 2048)              0

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 256)               524544

 dropout_1 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 815914 (3.11 MB)
Trainable params: 815018 (3.11 MB)
Non-trainable params: 896 (3.50 KB)
_________________________________________________________________
```

Model Compilation

[6]:
```
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/
 ↪03_model_s_data_augm_l2_adam.keras', save_best_only=True,␣
 ↪monitor='val_loss', verbose=1)

callbacks = [lr_scheduler, early_stopping, save_best_model]

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

Model Training

[7]:
```
history = model.fit(train_dataset,
                    validation_data=validation_dataset,
                    epochs=40,
                    callbacks=callbacks)
```

```
Epoch 1/30
628/628 [==============================] - ETA: 0s - loss: 1.8567 - accuracy:
```

```
0.3631
Epoch 1: val_loss improved from inf to 1.49206, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 59s 91ms/step - loss: 1.8567 -
accuracy: 0.3631 - val_loss: 1.4921 - val_accuracy: 0.4760 - lr: 0.0010
Epoch 2/30
628/628 [==============================] - ETA: 0s - loss: 1.4674 - accuracy:
0.4948
Epoch 2: val_loss improved from 1.49206 to 1.42269, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 91ms/step - loss: 1.4674 -
accuracy: 0.4948 - val_loss: 1.4227 - val_accuracy: 0.5017 - lr: 0.0010
Epoch 3/30
628/628 [==============================] - ETA: 0s - loss: 1.2811 - accuracy:
0.5701
Epoch 3: val_loss improved from 1.42269 to 1.14256, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 92ms/step - loss: 1.2811 -
accuracy: 0.5701 - val_loss: 1.1426 - val_accuracy: 0.6225 - lr: 0.0010
Epoch 4/30
628/628 [==============================] - ETA: 0s - loss: 1.1536 - accuracy:
0.6226
Epoch 4: val_loss improved from 1.14256 to 1.00053, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 91ms/step - loss: 1.1536 -
accuracy: 0.6226 - val_loss: 1.0005 - val_accuracy: 0.6748 - lr: 0.0010
Epoch 5/30
628/628 [==============================] - ETA: 0s - loss: 1.0679 - accuracy:
0.6580
Epoch 5: val_loss did not improve from 1.00053
628/628 [==============================] - 58s 92ms/step - loss: 1.0679 -
accuracy: 0.6580 - val_loss: 1.0437 - val_accuracy: 0.6675 - lr: 0.0010
Epoch 6/30
628/628 [==============================] - ETA: 0s - loss: 1.0021 - accuracy:
0.6860
Epoch 6: val_loss did not improve from 1.00053
628/628 [==============================] - 59s 93ms/step - loss: 1.0021 -
accuracy: 0.6860 - val_loss: 1.0098 - val_accuracy: 0.6846 - lr: 0.0010
Epoch 7/30
628/628 [==============================] - ETA: 0s - loss: 0.9456 - accuracy:
0.7100
Epoch 7: val_loss improved from 1.00053 to 0.89442, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 59s 93ms/step - loss: 0.9456 -
accuracy: 0.7100 - val_loss: 0.8944 - val_accuracy: 0.7275 - lr: 0.0010
Epoch 8/30
628/628 [==============================] - ETA: 0s - loss: 0.9029 - accuracy:
0.7291
```

```
Epoch 8: val_loss improved from 0.89442 to 0.87872, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 59s 93ms/step - loss: 0.9029 -
accuracy: 0.7291 - val_loss: 0.8787 - val_accuracy: 0.7381 - lr: 0.0010
Epoch 9/30
628/628 [==============================] - ETA: 0s - loss: 0.8736 - accuracy:
0.7398
Epoch 9: val_loss did not improve from 0.87872
628/628 [==============================] - 59s 94ms/step - loss: 0.8736 -
accuracy: 0.7398 - val_loss: 0.9018 - val_accuracy: 0.7300 - lr: 0.0010
Epoch 10/30
628/628 [==============================] - ETA: 0s - loss: 0.8389 - accuracy:
0.7537
Epoch 10: val_loss improved from 0.87872 to 0.86829, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 59s 93ms/step - loss: 0.8389 -
accuracy: 0.7537 - val_loss: 0.8683 - val_accuracy: 0.7400 - lr: 0.0010
Epoch 11/30
628/628 [==============================] - ETA: 0s - loss: 0.8166 - accuracy:
0.7656
Epoch 11: val_loss improved from 0.86829 to 0.84683, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 59s 93ms/step - loss: 0.8166 -
accuracy: 0.7656 - val_loss: 0.8468 - val_accuracy: 0.7621 - lr: 0.0010
Epoch 12/30
628/628 [==============================] - ETA: 0s - loss: 0.7903 - accuracy:
0.7770
Epoch 12: val_loss did not improve from 0.84683
628/628 [==============================] - 58s 92ms/step - loss: 0.7903 -
accuracy: 0.7770 - val_loss: 0.9075 - val_accuracy: 0.7526 - lr: 0.0010
Epoch 13/30
628/628 [==============================] - ETA: 0s - loss: 0.7753 - accuracy:
0.7836
Epoch 13: val_loss improved from 0.84683 to 0.80905, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 92ms/step - loss: 0.7753 -
accuracy: 0.7836 - val_loss: 0.8091 - val_accuracy: 0.7702 - lr: 0.0010
Epoch 14/30
628/628 [==============================] - ETA: 0s - loss: 0.7541 - accuracy:
0.7919
Epoch 14: val_loss improved from 0.80905 to 0.71656, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 92ms/step - loss: 0.7541 -
accuracy: 0.7919 - val_loss: 0.7166 - val_accuracy: 0.8051 - lr: 0.0010
Epoch 15/30
628/628 [==============================] - ETA: 0s - loss: 0.7352 - accuracy:
0.7978
Epoch 15: val_loss did not improve from 0.71656
```

```
628/628 [==============================] - 59s 93ms/step - loss: 0.7352 -
accuracy: 0.7978 - val_loss: 0.7175 - val_accuracy: 0.8064 - lr: 0.0010
Epoch 16/30
628/628 [==============================] - ETA: 0s - loss: 0.7118 - accuracy:
0.8076
Epoch 16: val_loss did not improve from 0.71656
628/628 [==============================] - 58s 92ms/step - loss: 0.7118 -
accuracy: 0.8076 - val_loss: 0.7789 - val_accuracy: 0.7885 - lr: 0.0010
Epoch 17/30
628/628 [==============================] - ETA: 0s - loss: 0.7016 - accuracy:
0.8112
Epoch 17: val_loss improved from 0.71656 to 0.70526, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 92ms/step - loss: 0.7016 -
accuracy: 0.8112 - val_loss: 0.7053 - val_accuracy: 0.8143 - lr: 0.0010
Epoch 18/30
628/628 [==============================] - ETA: 0s - loss: 0.6842 - accuracy:
0.8174
Epoch 18: val_loss did not improve from 0.70526
628/628 [==============================] - 58s 92ms/step - loss: 0.6842 -
accuracy: 0.8174 - val_loss: 0.7131 - val_accuracy: 0.8063 - lr: 0.0010
Epoch 19/30
628/628 [==============================] - ETA: 0s - loss: 0.6733 - accuracy:
0.8213
Epoch 19: val_loss did not improve from 0.70526
628/628 [==============================] - 58s 92ms/step - loss: 0.6733 -
accuracy: 0.8213 - val_loss: 0.8112 - val_accuracy: 0.7920 - lr: 0.0010
Epoch 20/30
628/628 [==============================] - ETA: 0s - loss: 0.6589 - accuracy:
0.8245
Epoch 20: val_loss improved from 0.70526 to 0.67245, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 92ms/step - loss: 0.6589 -
accuracy: 0.8245 - val_loss: 0.6725 - val_accuracy: 0.8223 - lr: 0.0010
Epoch 21/30
628/628 [==============================] - ETA: 0s - loss: 0.6525 - accuracy:
0.8276
Epoch 21: val_loss did not improve from 0.67245
628/628 [==============================] - 59s 93ms/step - loss: 0.6525 -
accuracy: 0.8276 - val_loss: 0.7078 - val_accuracy: 0.8170 - lr: 0.0010
Epoch 22/30
628/628 [==============================] - ETA: 0s - loss: 0.6356 - accuracy:
0.8343
Epoch 22: val_loss did not improve from 0.67245
628/628 [==============================] - 58s 91ms/step - loss: 0.6356 -
accuracy: 0.8343 - val_loss: 0.7108 - val_accuracy: 0.8161 - lr: 0.0010
Epoch 23/30
628/628 [==============================] - ETA: 0s - loss: 0.6220 - accuracy:
```

```
0.8382
Epoch 23: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.

Epoch 23: val_loss did not improve from 0.67245
628/628 [==============================] - 58s 91ms/step - loss: 0.6220 -
accuracy: 0.8382 - val_loss: 0.7637 - val_accuracy: 0.8010 - lr: 0.0010
Epoch 24/30
628/628 [==============================] - ETA: 0s - loss: 0.5608 - accuracy:
0.8584
Epoch 24: val_loss improved from 0.67245 to 0.60258, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 92ms/step - loss: 0.5608 -
accuracy: 0.8584 - val_loss: 0.6026 - val_accuracy: 0.8498 - lr: 1.0000e-04
Epoch 25/30
628/628 [==============================] - ETA: 0s - loss: 0.5310 - accuracy:
0.8681
Epoch 25: val_loss did not improve from 0.60258
628/628 [==============================] - 58s 91ms/step - loss: 0.5310 -
accuracy: 0.8681 - val_loss: 0.6038 - val_accuracy: 0.8476 - lr: 1.0000e-04
Epoch 26/30
628/628 [==============================] - ETA: 0s - loss: 0.5172 - accuracy:
0.8704
Epoch 26: val_loss improved from 0.60258 to 0.58894, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 92ms/step - loss: 0.5172 -
accuracy: 0.8704 - val_loss: 0.5889 - val_accuracy: 0.8533 - lr: 1.0000e-04
Epoch 27/30
628/628 [==============================] - ETA: 0s - loss: 0.5029 - accuracy:
0.8759
Epoch 27: val_loss improved from 0.58894 to 0.57572, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 59s 93ms/step - loss: 0.5029 -
accuracy: 0.8759 - val_loss: 0.5757 - val_accuracy: 0.8602 - lr: 1.0000e-04
Epoch 28/30
628/628 [==============================] - ETA: 0s - loss: 0.4940 - accuracy:
0.8787
Epoch 28: val_loss did not improve from 0.57572
628/628 [==============================] - 58s 92ms/step - loss: 0.4940 -
accuracy: 0.8787 - val_loss: 0.5806 - val_accuracy: 0.8564 - lr: 1.0000e-04
Epoch 29/30
628/628 [==============================] - ETA: 0s - loss: 0.4839 - accuracy:
0.8809
Epoch 29: val_loss improved from 0.57572 to 0.56494, saving model to
../models/03_model_s_data_augm_l2_adam.keras
628/628 [==============================] - 58s 92ms/step - loss: 0.4839 -
accuracy: 0.8809 - val_loss: 0.5649 - val_accuracy: 0.8586 - lr: 1.0000e-04
Epoch 30/30
628/628 [==============================] - ETA: 0s - loss: 0.4728 - accuracy:
```

```
0.8827
Epoch 30: val_loss did not improve from 0.56494
628/628 [==============================] - 58s 92ms/step - loss: 0.4728 -
accuracy: 0.8827 - val_loss: 0.5663 - val_accuracy: 0.8575 - lr: 1.0000e-04
```

Save Model History

[8]:
```
with open("../history/03_model_s_data_augm_l2_adam.pkl", "wb") as file:
    pickle.dump(history.history, file)
```

Model Evaluation

[9]:
```
val_loss, val_acc = model.evaluate(validation_dataset)
print(f'Model Validation Loss: {val_loss:.2f}')
print(f'Model Validation Accuracy: {val_acc:.2%}')
```

```
157/157 [==============================] - 3s 16ms/step - loss: 0.5663 -
accuracy: 0.8575
Model Validation Loss: 0.57
Model Validation Accuracy: 85.75%
```

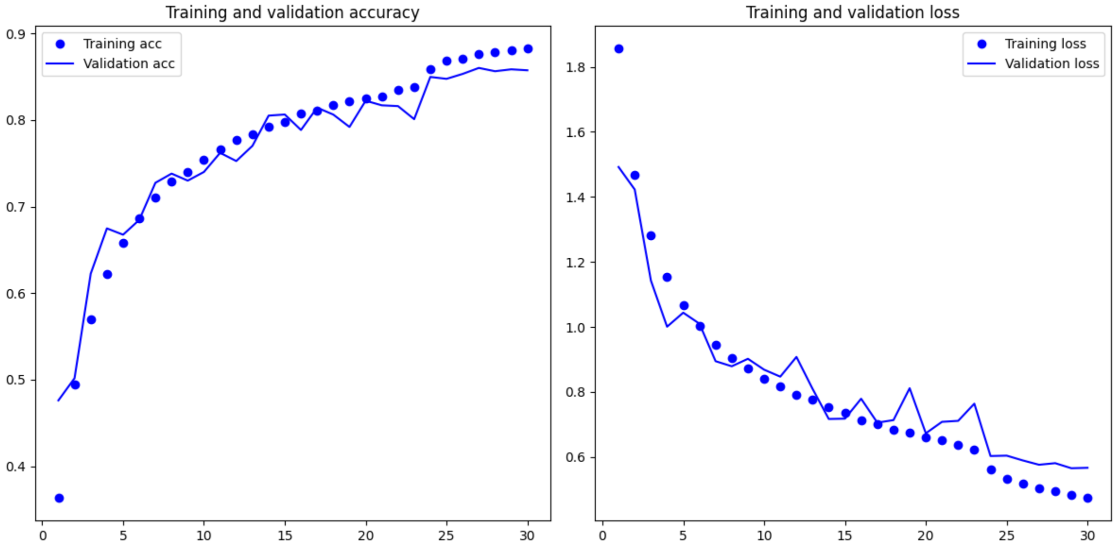
Model Training Visualization

[10]:
```
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 **24th** epoch.
  - The validation accuracy stops improving significantly after the **26th** epoch while the training accuracy keeps improving.
  - The validation loss stops improving significantly after the **24th** epoch while the training loss keeps improving.
  - The best model, based on validation loss, is saved on the **29th** epoch.

Model Testing

[11]:
```
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 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
```

```
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 11ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
```

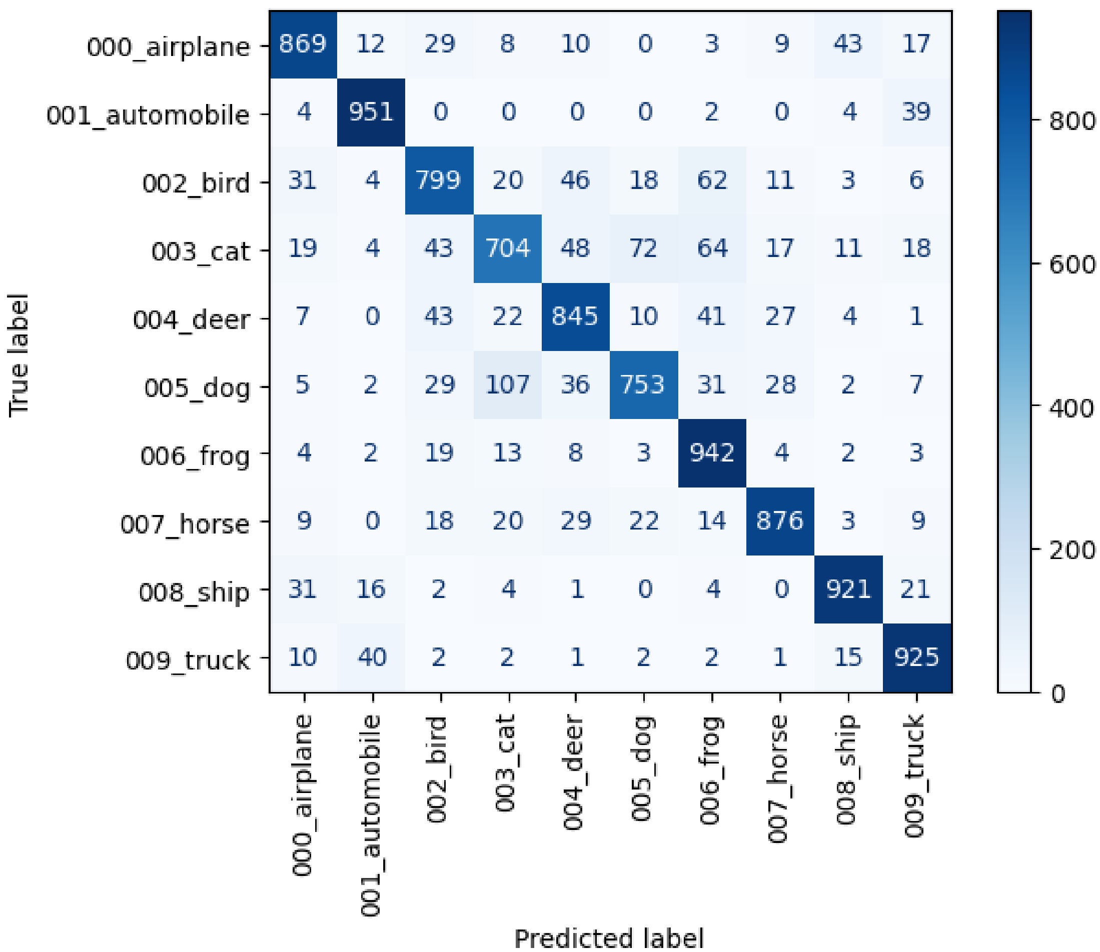
```
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 11ms/step
2/2 [==============================] - 0s 9ms/step
```

```
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
```

```
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 9ms/step
1/1 [==============================] - 0s 92ms/step
```

Confusion Matrix

[12]:
```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, 002_bird and 003_deer, 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

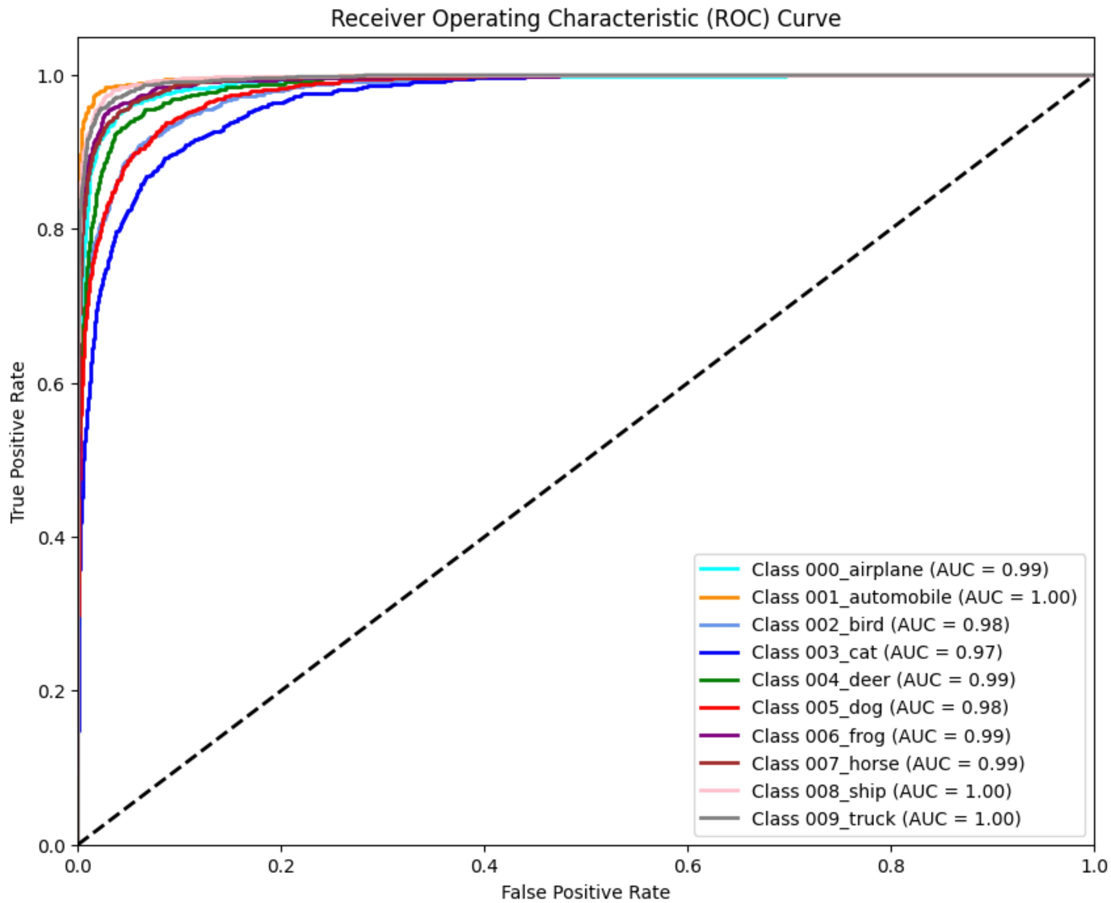
[13]:
```
test_labels_bin = label_binarize(test_labels, classes=range(NUM_CLASSES))

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, 002_bird and 005_dog 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.

[14]:
```
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 : 85.85%
Precision - Macro: 85.83%
Recall - Macro: 85.85%
F1-score - Macro: 85.73%
Precision - Weighted: 85.83%
Recall - Weighted: 85.85%
F1-score - Weighted: 85.73%
```

- **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 enhanced the architecture by:
    - We used L2 regularization.
  - We applied data augmentation techniques:
    - Random Horizontal Flip
    - Random Translation
    - Random Rotation
    - Random Zoom
    - Random Brightness
    - Random Contrast
  - We used the Adaptive Moment Estimation (Adam) optimizer with an initial learning rate of 0.001.
  - We kept the same 30 epochs with a batch size of 64.
  - We evaluated the model on the validation dataset:
    - Overfitting was observed after **24 epochs**, but the best model was saved at the **29th epoch**.
    - Training was intended for 30 epochs but stopped early due to the **Early Stopping** callback.
  - We evaluated 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 **85.85%** on the test set which was a good improvement.

2.0.2 Future Work

- In the next notebook:
  - We will upscale the dataset images to 128 x 128.
  - We will use feature extraction with the VGG16 Convolutional Base to:
    - Extract the train dataset feature maps.
    - Extract the validation dataset feature maps.
  - We will then train a classifier model with those extracted feature maps.
  - We will keep the same 30 epochs with a batch size of 64.
  - We will then join the VGG16 Convolutional Base with the classifier model.
  - We will test the results on the test set.