

ImageClassification-AI: Modelo T

Feature Extraction e Data Augmentation

Modelo que faz uso da técnica de transfer learning, Feature Extraction e, da técnica de Data Augmentation. Este modelo irá fazer uso do modelo VGG16.

1. Setup

1.1 Importar dependências

Importação das bibliotecas necessárias para o desenvolvimento do modelo.

São de notar as bibliotecas:

- Tensorflow e Keras, que vão ser utilizadas na construção do modelo e no seu processo de treino
- Matplotlib (em específico o pyplot), Seaborn e sklearn, que vão ser utilizadas para facilitar a análise e a compreensão das métricas atribuídas ao modelo, da sua evolução, e dos resultados obtidos
- Image_dataset_from_directory (através do keras.utils), numpy e OS para o carregamento e tratamento dos dados

```
import os
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from keras.utils import image_dataset_from_directory
from tensorflow import keras
from keras import layers, regularizers, optimizers
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.applications.vgg16 import VGG16
```

1.2 Desativar warnings do Tensorflow

Para desenvolvimento deste modelo foi utilizada a versão 2.10.0 do Tensorflow. Devido a este facto, ficou compreendido que seria benéfico desativar as mensagens de warning dadas pelo Tensorflow, deixando apenas as mensagens de erro, com o intuito de melhorar substancialmente a legibilidade do notebook. É importante realçar que, nenhuma das mensagens de aviso que serão desativadas, em algum momento afetam qualquer aspeto do modelo ou sequer ajudam a compreender potenciais problemas com este.

```
tf.get_logger().setLevel('ERROR')
```

1.3 Tratamento de dados

Definição das classes do problema:

- Tamanho das imagens RGB (224x224x3 pixels)
- Tamanho de cada batch (32)
- Diretorias dos datasets de treino, validação e teste

Para a criação dos datasets é utilizado o `image_dataset_from_directory` com os parâmetros relativos à diretoria onde estão as imagens, o tamanho destas, o tamanho de cada batch, a definição das labels como `categorical` (requerido devido ao facto do problema em questão envolver 10 classes; as labels serão uma tensor `float32` de tamanho `(batch_size, num_classes)`, que iram representar, cada, um one-hot encoding de cada index de cada classe).

Aqui é, ainda, importante notar:

- O dataset de treino está a ser baralhado de modo a que, durante o processo de treino, o modelo não decore padrões nas imagens de treino. Para além disso, é relevante perceber que o dataset de treino é construído através da concatenação de quatro datasets de treino mais pequenos (cada um relativo a uma das diretorias de treino)
- Os datasets de validação e de testes não são baralhados. Ao baralhar o dataset de treino a análise dos resultados obtidos pelo modelo seria extremamente dificultada (e.g. ao construir um `classification report` para este dataset os resultados seriam incorretos porque as labels não iriam corresponder). No que toca ao dataset de validação, a questão entre baralhar ou não acaba por ser irrelevante já que não existe nenhum tipo de benefício para o fazer. Isto foi confirmado por uma pesquisa sobre o assunto e por tentativas de treino do modelo com o dataset de validação baralhado e sem estar baralhado (os resultados eram os mesmos)

```
class_names = []

IMG_SIZE = 224
BATCH_SIZE = 32

train_dirs = ['train1', 'train2', 'train3', 'train5']
val_dir = 'train4'
test_dir = 'test'

print("BUILDING TRAIN DATASET...")
train_dataset_list = []
for td in train_dirs:
    train_dataset_list.append(image_dataset_from_directory(td,
        image_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE,
        label_mode='categorical', shuffle=True, color_mode='rgb'))

train_dataset = train_dataset_list[0]
for name in train_dataset_list[0].class_names:
    idx = name.index('_') + 1
```

```

class_names.append(name[idx:])

for d in train_dataset_list[1:]:
    train_dataset = train_dataset.concatenate(d)

print("\nBUILDING VALIDATION DATASET...")
val_dataset = image_dataset_from_directory(val_dir,
image_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE,
label_mode='categorical', shuffle=False, color_mode='rgb')

print("\nBUILDING TEST DATASET...")
test_dataset = image_dataset_from_directory(test_dir,
image_size=(IMG_SIZE, IMG_SIZE), batch_size=BATCH_SIZE,
label_mode='categorical', shuffle=False, color_mode='rgb')

BUILDING TRAIN DATASET...
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.

BUILDING VALIDATION DATASET...
Found 10000 files belonging to 10 classes.

BUILDING TEST DATASET...
Found 10000 files belonging to 10 classes.

```

1.4 Carregar o modelo VGG16

Carregar o modelo VGG16 para ser, posteriormente, utilizado na construção e processo de treino do modelo. É de realçar que, aqui, estamos a congelar a parte de feature extraction do modelo VGG16.

```

conv_base = VGG16(weights='imagenet', include_top=False,
input_shape=(IMG_SIZE, IMG_SIZE, 3))
conv_base.trainable = False

```

1.5 Definição operações de Data Augmentation

A Data Augmentation é uma das técnicas utilizadas para combater o overfitting.

Define-se aqui, então, as operações de data augmentation a utilizar posteriormente:

- RandomFlip("horizontal"): vai rodar algumas imagens horizontalmente
- RandomRotation(0.1): vai rodar algumas imagens em 10%
- RandomZoom(0.2): vai aproximar algumas imagens em 20%

```

data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),

```

```
        layers.RandomRotation(0.1),  
        layers.RandomZoom(0.2),  
    ]  
)
```

2 Visualização

2.1 - Classes e número de imagens

Visualização das classes que envolvem o problema e da quantidade de imagens contidas em cada dataset

```
print("\nClasses: " + str(class_names))  
  
total_train = 0  
for td in train_dirs:  
    class_folders = next(os.walk(td))[1]  
    for cf in class_folders:  
        total_train += len(os.listdir(os.path.join(td, cf)))  
  
total_val = 0  
class_folders = next(os.walk(val_dir))[1]  
for folder in class_folders:  
    folder_path = os.path.join(val_dir, folder)  
    total_val += len(os.listdir(folder_path))  
  
total_test = 0  
class_folders = next(os.walk(test_dir))[1]  
for folder in class_folders:  
    folder_path = os.path.join(test_dir, folder)  
    total_test += len(os.listdir(folder_path))  
  
print("Dataset de treino: " + str(total_train) + " imagens")  
print("Dataset de validação: " + str(total_val) + " imagens")  
print("Dataset de teste: " + str(total_test) + " imagens")  
  
Classes: ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog',  
'frog', 'horse', 'ship', 'truck']  
Dataset de treino: 40000 imagens  
Dataset de validação: 10000 imagens  
Dataset de teste: 10000 imagens
```

2.2 Tamanhos

Visualização dos tamanhos:

- Cada batch tem 32 imagens
- Cada imagem RGB tem 224x224 pixels (224x224x3)
- Cada batch de labels tem 10 classes


```
[0.8117647 , 0.80784315, 0.827451 ],
...,
[0.6901961 , 0.6392157 , 0.6627451 ],
[0.6901961 , 0.6392157 , 0.6627451 ],
[0.6901961 , 0.6392157 , 0.6627451 ]],

[[[0.8117647 , 0.80784315, 0.827451 ],
[0.8117647 , 0.80784315, 0.827451 ],
[0.8117647 , 0.80784315, 0.827451 ],
...,
[0.6901961 , 0.6392157 , 0.6627451 ],
[0.6901961 , 0.6392157 , 0.6627451 ],
[0.6901961 , 0.6392157 , 0.6627451 ]],

[[[0.8117647 , 0.80784315, 0.827451 ],
[0.8117647 , 0.80784315, 0.827451 ],
[0.8117647 , 0.80784315, 0.827451 ],
...,
[0.6901961 , 0.6392157 , 0.6627451 ],
[0.6901961 , 0.6392157 , 0.6627451 ],
[0.6901961 , 0.6392157 , 0.6627451 ]]]],

[[[0.52156866, 0.5921569 , 0.6745098 ],
[0.52156866, 0.5921569 , 0.6745098 ],
[0.52156866, 0.5921569 , 0.6745098 ],
...,
[0.85490197, 0.8352941 , 0.8117647 ],
[0.85490197, 0.8352941 , 0.8117647 ],
[0.85490197, 0.8352941 , 0.8117647 ]],

[[[0.52156866, 0.5921569 , 0.6745098 ],
[0.52156866, 0.5921569 , 0.6745098 ],
[0.52156866, 0.5921569 , 0.6745098 ],
...,
[0.85490197, 0.8352941 , 0.8117647 ],
[0.85490197, 0.8352941 , 0.8117647 ],
[0.85490197, 0.8352941 , 0.8117647 ]],

[[[0.52156866, 0.5921569 , 0.6745098 ],
[0.52156866, 0.5921569 , 0.6745098 ],
[0.52156866, 0.5921569 , 0.6745098 ],
...,
[0.85490197, 0.8352941 , 0.8117647 ],
[0.85490197, 0.8352941 , 0.8117647 ],
[0.85490197, 0.8352941 , 0.8117647 ]],

...,

[[[0.34509805, 0.3529412 , 0.38039216],
```

```
[0.34509805, 0.3529412 , 0.38039216],  
[0.34509805, 0.3529412 , 0.38039216],  
...,  
[0.3529412 , 0.4 , 0.44705883],  
[0.3529412 , 0.4 , 0.44705883],  
[0.3529412 , 0.4 , 0.44705883]]],
```

```
[[0.34509805, 0.3529412 , 0.38039216],  
[0.34509805, 0.3529412 , 0.38039216],  
[0.34509805, 0.3529412 , 0.38039216],  
...,  
[0.3529412 , 0.4 , 0.44705883],  
[0.3529412 , 0.4 , 0.44705883],  
[0.3529412 , 0.4 , 0.44705883]]],
```

```
[[0.34509805, 0.3529412 , 0.38039216],  
[0.34509805, 0.3529412 , 0.38039216],  
[0.34509805, 0.3529412 , 0.38039216],  
...,  
[0.3529412 , 0.4 , 0.44705883],  
[0.3529412 , 0.4 , 0.44705883],  
[0.3529412 , 0.4 , 0.44705883]]],
```

```
[[[0.12941177, 0.1254902 , 0.07450981],  
[0.12941177, 0.1254902 , 0.07450981],  
[0.12941177, 0.1254902 , 0.07450981],  
...,  
[0.44705883, 0.2901961 , 0.14901961],  
[0.44705883, 0.2901961 , 0.14901961],  
[0.44705883, 0.2901961 , 0.14901961]]],
```

```
[[0.12941177, 0.1254902 , 0.07450981],  
[0.12941177, 0.1254902 , 0.07450981],  
[0.12941177, 0.1254902 , 0.07450981],  
...,  
[0.44705883, 0.2901961 , 0.14901961],  
[0.44705883, 0.2901961 , 0.14901961],  
[0.44705883, 0.2901961 , 0.14901961]]],
```

```
[[0.12941177, 0.1254902 , 0.07450981],  
[0.12941177, 0.1254902 , 0.07450981],  
[0.12941177, 0.1254902 , 0.07450981],  
...,  
[0.44705883, 0.2901961 , 0.14901961],  
[0.44705883, 0.2901961 , 0.14901961],  
[0.44705883, 0.2901961 , 0.14901961]]],
```

```
...,
```

```
[[0.6156863 , 0.47058824, 0.28627452],
 [0.6156863 , 0.47058824, 0.28627452],
 [0.6156863 , 0.47058824, 0.28627452],
 ...,
 [0.4 , 0.3372549 , 0.1882353 ],
 [0.4 , 0.3372549 , 0.1882353 ],
 [0.4 , 0.3372549 , 0.1882353 ]],

[[0.6156863 , 0.47058824, 0.28627452],
 [0.6156863 , 0.47058824, 0.28627452],
 [0.6156863 , 0.47058824, 0.28627452],
 ...,
 [0.4 , 0.3372549 , 0.1882353 ],
 [0.4 , 0.3372549 , 0.1882353 ],
 [0.4 , 0.3372549 , 0.1882353 ]],

[[0.6156863 , 0.47058824, 0.28627452],
 [0.6156863 , 0.47058824, 0.28627452],
 [0.6156863 , 0.47058824, 0.28627452],
 ...,
 [0.4 , 0.3372549 , 0.1882353 ],
 [0.4 , 0.3372549 , 0.1882353 ],
 [0.4 , 0.3372549 , 0.1882353 ]]],

...,

[[[0.627451 , 0.54509807, 0.49019608],
 [0.627451 , 0.54509807, 0.49019608],
 [0.627451 , 0.54509807, 0.49019608],
 ...,
 [0.57254905, 0.4117647 , 0.30980393],
 [0.57254905, 0.4117647 , 0.30980393],
 [0.57254905, 0.4117647 , 0.30980393]]],

[[0.627451 , 0.54509807, 0.49019608],
 [0.627451 , 0.54509807, 0.49019608],
 [0.627451 , 0.54509807, 0.49019608],
 ...,
 [0.57254905, 0.4117647 , 0.30980393],
 [0.57254905, 0.4117647 , 0.30980393],
 [0.57254905, 0.4117647 , 0.30980393]]],

[[0.627451 , 0.54509807, 0.49019608],
 [0.627451 , 0.54509807, 0.49019608],
 [0.627451 , 0.54509807, 0.49019608],
 ...,
 [0.57254905, 0.4117647 , 0.30980393],
 [0.57254905, 0.4117647 , 0.30980393],
 [0.57254905, 0.4117647 , 0.30980393]]],
```


[0.57254905, 0.4117647 , 0.30980393]],

...,

[[0.3372549 , 0.29803923, 0.28627452],
[0.3372549 , 0.29803923, 0.28627452],
[0.3372549 , 0.29803923, 0.28627452],

...,

[0.6117647 , 0.5411765 , 0.40784314],
[0.6117647 , 0.5411765 , 0.40784314],
[0.6117647 , 0.5411765 , 0.40784314]]],

[[0.3372549 , 0.29803923, 0.28627452],
[0.3372549 , 0.29803923, 0.28627452],
[0.3372549 , 0.29803923, 0.28627452],

...,

[0.6117647 , 0.5411765 , 0.40784314],
[0.6117647 , 0.5411765 , 0.40784314],
[0.6117647 , 0.5411765 , 0.40784314]]],

[[0.3372549 , 0.29803923, 0.28627452],
[0.3372549 , 0.29803923, 0.28627452],
[0.3372549 , 0.29803923, 0.28627452],

...,

[0.6117647 , 0.5411765 , 0.40784314],
[0.6117647 , 0.5411765 , 0.40784314],
[0.6117647 , 0.5411765 , 0.40784314]]],

[[[0.8039216 , 0.73333335, 0.7137255],
[0.8039216 , 0.73333335, 0.7137255],
[0.8039216 , 0.73333335, 0.7137255],

...,

[0.28235295, 0.25882354, 0.26666668],
[0.28235295, 0.25882354, 0.26666668],
[0.28235295, 0.25882354, 0.26666668]]],

[[0.8039216 , 0.73333335, 0.7137255],
[0.8039216 , 0.73333335, 0.7137255],
[0.8039216 , 0.73333335, 0.7137255],

...,

[0.28235295, 0.25882354, 0.26666668],
[0.28235295, 0.25882354, 0.26666668],
[0.28235295, 0.25882354, 0.26666668]]],

[[0.8039216 , 0.73333335, 0.7137255],
[0.8039216 , 0.73333335, 0.7137255],
[0.8039216 , 0.73333335, 0.7137255],

...,

[0.28235295, 0.25882354, 0.26666668],

```
[0.28235295, 0.25882354, 0.26666668],  
[0.28235295, 0.25882354, 0.26666668]],
```

```
...,
```

```
[[0.16470589, 0.14117648, 0.15686275],  
 [0.16470589, 0.14117648, 0.15686275],  
 [0.16470589, 0.14117648, 0.15686275],  
 ...,  
 [0.15686275, 0.13333334, 0.15686275],  
 [0.15686275, 0.13333334, 0.15686275],  
 [0.15686275, 0.13333334, 0.15686275]]],
```

```
[[0.16470589, 0.14117648, 0.15686275],  
 [0.16470589, 0.14117648, 0.15686275],  
 [0.16470589, 0.14117648, 0.15686275],  
 ...,  
 [0.15686275, 0.13333334, 0.15686275],  
 [0.15686275, 0.13333334, 0.15686275],  
 [0.15686275, 0.13333334, 0.15686275]]],
```

```
[[0.16470589, 0.14117648, 0.15686275],  
 [0.16470589, 0.14117648, 0.15686275],  
 [0.16470589, 0.14117648, 0.15686275],  
 ...,  
 [0.15686275, 0.13333334, 0.15686275],  
 [0.15686275, 0.13333334, 0.15686275],  
 [0.15686275, 0.13333334, 0.15686275]]],
```

```
[[[1.      , 1.      , 1.      ],  
  [1.      , 1.      , 1.      ],  
  [1.      , 1.      , 1.      ],  
  ...,  
  [1.      , 1.      , 1.      ],  
  [1.      , 1.      , 1.      ],  
  [1.      , 1.      , 1.      ]],
```

```
[[1.      , 1.      , 1.      ],  
 [1.      , 1.      , 1.      ],  
 [1.      , 1.      , 1.      ],  
 ...,  
 [1.      , 1.      , 1.      ],  
 [1.      , 1.      , 1.      ],  
 [1.      , 1.      , 1.      ]],
```

```
[[1.      , 1.      , 1.      ],  
 [1.      , 1.      , 1.      ],  
 [1.      , 1.      , 1.      ],  
 ...,
```

```

        [1.          , 1.          , 1.          ],
        [1.          , 1.          , 1.          ],
        [1.          , 1.          , 1.          ]],
    ...,
    [[1.          , 1.          , 1.          ],
     [1.          , 1.          , 1.          ],
     [1.          , 1.          , 1.          ],
     ...,
     [1.          , 0.99215686, 1.          ],
     [1.          , 0.99215686, 1.          ],
     [1.          , 0.99215686, 1.          ]],
    [[1.          , 1.          , 1.          ],
     [1.          , 1.          , 1.          ],
     [1.          , 1.          , 1.          ],
     ...,
     [1.          , 0.99215686, 1.          ],
     [1.          , 0.99215686, 1.          ],
     [1.          , 0.99215686, 1.          ]],
    [[1.          , 1.          , 1.          ],
     [1.          , 1.          , 1.          ],
     [1.          , 1.          , 1.          ],
     ...,
     [1.          , 0.99215686, 1.          ],
     [1.          , 0.99215686, 1.          ],
     [1.          , 0.99215686, 1.          ]]]], dtype=float32)

```

2.4 - Imagens do dataset de treino

Visualização de dez imagens aleatórias do dataset de treino.

```

plt.figure(figsize=(12, 6)) # Aumentar o tamanho das imagens no plot
for data_batch, label_batch in train_dataset.take(1):
    for i in range(10):
        plt.subplot(2, 5, i + 1) # mostrar as imagens todas no "mesmo
        plot" de modo a economizar espaço
        plt.title(class_names[np.argmax(label_batch[i])]) # mostrar a
        classe da imagem
        plt.imshow(data_batch[i].numpy().astype('uint8'))
        plt.xticks([]) # não mostrar os eixos (irrelevante para a
        visualização)
        plt.yticks([]) # não mostrar os eixos (irrelevante para a
        visualização)
    plt.show()

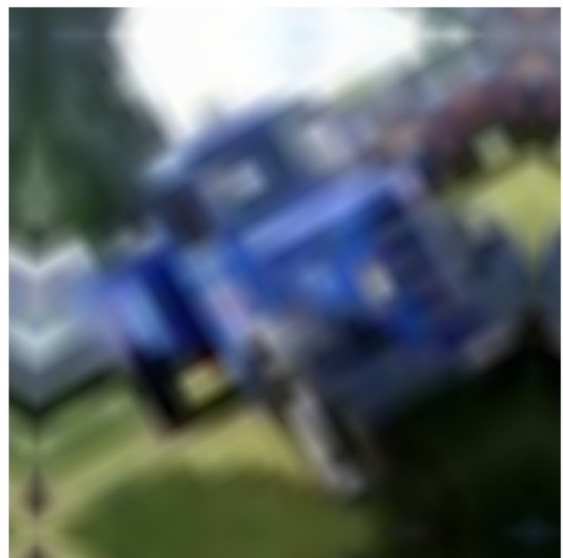
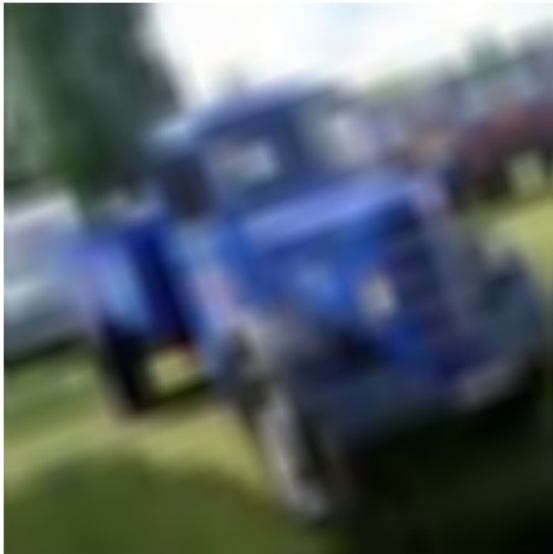
```



2.5 Imagem com Data Augmentation

Visualizar os efeitos das operações de Data Augmentation definidas anteriormente.

```
plt.figure(figsize=(10, 10))
for images, _ in train_dataset.take(1):
    for i in range(4):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(2, 2, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"))
        plt.axis("off")
```



3. Modelo

3.1 Definição do bloco de classificação

Aqui está a ser efetuado a definição da parte relativa a classificação do modelo, estando a parte restante encarregue do modelo VGG16.

É de notar:

- `Input_shape` de `7x7x512`: vai mudar consoante o tamanho das imagens que for escolhido. Como neste caso estamos a utilizar imagens `224x224x3` este é o shape que irá chegar ao bloco de classificação.

- A aplicação de técnicas de Data Augmentation
- A utilização do Flatten para transformar os valores obtidos até aquele ponto pelo modelo VGG16
- A camada densa com 256 filtros à qual aplicamos a técnica de normalização BatchNormalization de modo a manter consistente a distribuição dos valores. Depois de aplicada esta técnica, é então utilizada a função de ativação ReLu e a técnica de Dropout() com o valor de 0.5, ou seja, vão ser "ignorados" 50% dos neurónios contidos na camada densa.
- É utilizada uma outra camada densa, com 10 filtros, para efetuar a classificação da imagem. Aqui é utilizada a função de ativação "softmax" devido a esta ser mais apropriada a um problema de classificação com várias classes diferentes. Para além disso, é também, utilizado a regularização L2 para, tal como o Dropout, mitigar o overfitting

É feito um sumário do modelo para melhor compreensão deste, especialmente no que toca ao tamanho dos feature maps em cada ponto e à quantidade de parâmetros que este envolve.

```
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.Dense(256)(x)
x = layers.BatchNormalization()(x)
x = layers.Activation('relu')(x)
x = layers.Dropout(0.5)(x)

outputs = layers.Dense(10, activation='softmax',
kernel_regularizer=regularizers.l2(0.01))(x)
model = keras.Model(inputs, outputs)
```

```
model.summary()
```

```
Model: "model"
```

Layer (type)	Output Shape	Param #
=====		
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
sequential (Sequential)	(None, 224, 224, 3)	0
tf.__operators__.getitem (SlicingOpLambda)	(None, 224, 224, 3)	0

tf.nn.bias_add (TFOpLambda)	(None, 224, 224, 3)	0
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 256)	6422784
batch_normalization (Batch Normalization)	(None, 256)	1024
activation (Activation)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2570
=====		
Total params: 21,141,066		
Trainable params: 6,425,866		
Non-trainable params: 14,715,200		

3.2 Compilação

É utilizada a função de loss "categorical_crossentropy" devido à natureza do problema (várias classes). Para analisar o desempenho do modelo são utilizadas métricas de acerto (neste caso o "CategoricalAccuracy" em vez do Accuracy normal devido ao contexto do problema), precisão e recall. É, ainda, importante referir que inicialmente era para ser incluída uma métrica de cálculo relativo ao F1-Score, mas, devido ao facto de ter sido utilizado o Tensorflow 2.10.0 para treinar os modelos, como supramencionado, não foi possível utilizar esta métrica. Isto acontece porque esta versão do Tensorflow não suporta a referida métrica. Realizaram-se experiências utilizando a métrica F1-Score do Tensorflow Addons mas, os resultados não foram satisfatórios.

Nesta modelo S foi utilizado como otimizador o Adam(), com o objetivo de explorar mais optimizadores. Não é definido um learning rate a ser utilizado por este optimizador, sendo utilizado o por omissão, já que esta já possui, de base, técnicas de optimização do learning rate.

```
model.compile(
    loss="categorical_crossentropy",
    optimizer=keras.optimizers.Adam(),
    metrics=[
        tf.keras.metrics.CategoricalAccuracy(name="accuracy"),
        keras.metrics.Precision(name="precision"),
        keras.metrics.Recall(name="recall"),
    ]
)
```

3.3 Processo de treino

São definidas callbacks de:

- EarlyStopping, que vai servir para interromper o processo de treino. É monitorizada a loss no dataset de validação em cada epoch e, se após 10 epochs não houver melhoria desta métrica, então o treino vai ser interrompido
- ModelCheckpoint, que vai permitir guardar o melhor modelo obtido durante o processo de treino (em troca de se guardar o modelo na ultima epoch de treino que, pode não ser necessariamente o melhor como é o caso de, por exemplo, situações onde o modelo começa a entrar em overfitting). Aqui é definida a diretoria onde guardar o melhor modelo e a metrica de monitorização que, neste caso, volta a ser a loss no dataset de validação. É, também utilizado o verbose para melhorar a compreensão do processo de treino.

Com isto, é, então, realizado o processo de treino (model.fit()) utilizando:

- O dataset de treino
- 50 epochs
- O dataset de validação para representar a capacidade de generalização do modelo
- As callbacks de EarlyStopping e ModelCheckpoint definidas

```
# Definir as callbacks
callbacks = [
    keras.callbacks.EarlyStopping(
        monitor="val_loss",
        patience=10,
    ),
    keras.callbacks.ModelCheckpoint(
        filepath='models/IC_T_FE_DA.keras',
        save_best_only = True,
        monitor='val_loss',
        verbose=1
    )
]

# Treinar o modelo
history = model.fit(train_dataset, epochs=50,
                    validation_data=val_dataset, callbacks=callbacks)

Epoch 1/50
1252/1252 [=====] - ETA: 0s - loss: 0.8329 -
accuracy: 0.7561 - precision: 0.8411 - recall: 0.6710
Epoch 1: val_loss improved from inf to 0.48697, saving model to
models\IC_T_FE_DA.keras
1252/1252 [=====] - 347s 266ms/step - loss:
0.8329 - accuracy: 0.7561 - precision: 0.8411 - recall: 0.6710 -
val_loss: 0.4870 - val_accuracy: 0.8670 - val_precision: 0.9112 -
val_recall: 0.8220
Epoch 2/50
```



```
1252/1252 [=====] - ETA: 0s - loss: 0.6144 -  
accuracy: 0.8184 - precision: 0.8783 - recall: 0.7536  
Epoch 2: val_loss improved from 0.48697 to 0.44528, saving model to  
models\IC_T_FE_DA.keras  
1252/1252 [=====] - 349s 279ms/step - loss:  
0.6144 - accuracy: 0.8184 - precision: 0.8783 - recall: 0.7536 -  
val_loss: 0.4453 - val_accuracy: 0.8754 - val_precision: 0.9111 -  
val_recall: 0.8381  
Epoch 3/50  
1252/1252 [=====] - ETA: 0s - loss: 0.5642 -  
accuracy: 0.8314 - precision: 0.8846 - recall: 0.7772  
Epoch 3: val_loss improved from 0.44528 to 0.43136, saving model to  
models\IC_T_FE_DA.keras  
1252/1252 [=====] - 350s 279ms/step - loss:  
0.5642 - accuracy: 0.8314 - precision: 0.8846 - recall: 0.7772 -  
val_loss: 0.4314 - val_accuracy: 0.8759 - val_precision: 0.9118 -  
val_recall: 0.8425  
Epoch 4/50  
1252/1252 [=====] - ETA: 0s - loss: 0.5311 -  
accuracy: 0.8422 - precision: 0.8899 - recall: 0.7921  
Epoch 4: val_loss improved from 0.43136 to 0.41406, saving model to  
models\IC_T_FE_DA.keras  
1252/1252 [=====] - 348s 278ms/step - loss:  
0.5311 - accuracy: 0.8422 - precision: 0.8899 - recall: 0.7921 -  
val_loss: 0.4141 - val_accuracy: 0.8819 - val_precision: 0.9119 -  
val_recall: 0.8533  
Epoch 5/50  
1252/1252 [=====] - ETA: 0s - loss: 0.5060 -  
accuracy: 0.8481 - precision: 0.8951 - recall: 0.8026  
Epoch 5: val_loss improved from 0.41406 to 0.40968, saving model to  
models\IC_T_FE_DA.keras  
1252/1252 [=====] - 339s 271ms/step - loss:  
0.5060 - accuracy: 0.8481 - precision: 0.8951 - recall: 0.8026 -  
val_loss: 0.4097 - val_accuracy: 0.8848 - val_precision: 0.9133 -  
val_recall: 0.8567  
Epoch 6/50  
1252/1252 [=====] - ETA: 0s - loss: 0.4845 -  
accuracy: 0.8534 - precision: 0.8967 - recall: 0.8121  
Epoch 6: val_loss improved from 0.40968 to 0.40193, saving model to  
models\IC_T_FE_DA.keras  
1252/1252 [=====] - 329s 263ms/step - loss:  
0.4845 - accuracy: 0.8534 - precision: 0.8967 - recall: 0.8121 -  
val_loss: 0.4019 - val_accuracy: 0.8840 - val_precision: 0.9125 -  
val_recall: 0.8587  
Epoch 7/50  
1252/1252 [=====] - ETA: 0s - loss: 0.4672 -  
accuracy: 0.8576 - precision: 0.8994 - recall: 0.8199  
Epoch 7: val_loss improved from 0.40193 to 0.39901, saving model to  
models\IC_T_FE_DA.keras
```

```
1252/1252 [=====] - 329s 263ms/step - loss: 0.4672 - accuracy: 0.8576 - precision: 0.8994 - recall: 0.8199 - val_loss: 0.3990 - val_accuracy: 0.8847 - val_precision: 0.9112 - val_recall: 0.8631
Epoch 8/50
1252/1252 [=====] - ETA: 0s - loss: 0.4505 - accuracy: 0.8647 - precision: 0.9034 - recall: 0.8277
Epoch 8: val_loss did not improve from 0.39901
1252/1252 [=====] - 329s 263ms/step - loss: 0.4505 - accuracy: 0.8647 - precision: 0.9034 - recall: 0.8277 - val_loss: 0.4069 - val_accuracy: 0.8826 - val_precision: 0.9076 - val_recall: 0.8619
Epoch 9/50
1252/1252 [=====] - ETA: 0s - loss: 0.4292 - accuracy: 0.8735 - precision: 0.9099 - recall: 0.8383
Epoch 9: val_loss did not improve from 0.39901
1252/1252 [=====] - 330s 263ms/step - loss: 0.4292 - accuracy: 0.8735 - precision: 0.9099 - recall: 0.8383 - val_loss: 0.4028 - val_accuracy: 0.8815 - val_precision: 0.9085 - val_recall: 0.8647
Epoch 10/50
1252/1252 [=====] - ETA: 0s - loss: 0.4184 - accuracy: 0.8758 - precision: 0.9103 - recall: 0.8417
Epoch 10: val_loss did not improve from 0.39901
1252/1252 [=====] - 330s 263ms/step - loss: 0.4184 - accuracy: 0.8758 - precision: 0.9103 - recall: 0.8417 - val_loss: 0.3999 - val_accuracy: 0.8832 - val_precision: 0.9074 - val_recall: 0.8663
Epoch 11/50
1252/1252 [=====] - ETA: 0s - loss: 0.4080 - accuracy: 0.8795 - precision: 0.9139 - recall: 0.8485
Epoch 11: val_loss improved from 0.39901 to 0.39161, saving model to models\IC_T_FE_DA.keras
1252/1252 [=====] - 330s 263ms/step - loss: 0.4080 - accuracy: 0.8795 - precision: 0.9139 - recall: 0.8485 - val_loss: 0.3916 - val_accuracy: 0.8863 - val_precision: 0.9091 - val_recall: 0.8686
Epoch 12/50
1252/1252 [=====] - ETA: 0s - loss: 0.3957 - accuracy: 0.8832 - precision: 0.9135 - recall: 0.8525
Epoch 12: val_loss did not improve from 0.39161
1252/1252 [=====] - 330s 264ms/step - loss: 0.3957 - accuracy: 0.8832 - precision: 0.9135 - recall: 0.8525 - val_loss: 0.3992 - val_accuracy: 0.8847 - val_precision: 0.9095 - val_recall: 0.8682
Epoch 13/50
1252/1252 [=====] - ETA: 0s - loss: 0.3917 - accuracy: 0.8849 - precision: 0.9148 - recall: 0.8550
Epoch 13: val_loss did not improve from 0.39161
```

```
1252/1252 [=====] - 330s 264ms/step - loss:
0.3917 - accuracy: 0.8849 - precision: 0.9148 - recall: 0.8550 -
val_loss: 0.3959 - val_accuracy: 0.8857 - val_precision: 0.9064 -
val_recall: 0.8681
Epoch 14/50
1252/1252 [=====] - ETA: 0s - loss: 0.3801 -
accuracy: 0.8874 - precision: 0.9173 - recall: 0.8599
Epoch 14: val_loss did not improve from 0.39161
1252/1252 [=====] - 330s 264ms/step - loss:
0.3801 - accuracy: 0.8874 - precision: 0.9173 - recall: 0.8599 -
val_loss: 0.4003 - val_accuracy: 0.8852 - val_precision: 0.9078 -
val_recall: 0.8694
Epoch 15/50
1252/1252 [=====] - ETA: 0s - loss: 0.3664 -
accuracy: 0.8925 - precision: 0.9200 - recall: 0.8656
Epoch 15: val_loss did not improve from 0.39161
1252/1252 [=====] - 331s 264ms/step - loss:
0.3664 - accuracy: 0.8925 - precision: 0.9200 - recall: 0.8656 -
val_loss: 0.4100 - val_accuracy: 0.8818 - val_precision: 0.9041 -
val_recall: 0.8663
Epoch 16/50
1252/1252 [=====] - ETA: 0s - loss: 0.3607 -
accuracy: 0.8948 - precision: 0.9225 - recall: 0.8682
Epoch 16: val_loss did not improve from 0.39161
1252/1252 [=====] - 332s 265ms/step - loss:
0.3607 - accuracy: 0.8948 - precision: 0.9225 - recall: 0.8682 -
val_loss: 0.4003 - val_accuracy: 0.8841 - val_precision: 0.9055 -
val_recall: 0.8694
Epoch 17/50
1252/1252 [=====] - ETA: 0s - loss: 0.3563 -
accuracy: 0.8964 - precision: 0.9223 - recall: 0.8700
Epoch 17: val_loss did not improve from 0.39161
1252/1252 [=====] - 331s 265ms/step - loss:
0.3563 - accuracy: 0.8964 - precision: 0.9223 - recall: 0.8700 -
val_loss: 0.3989 - val_accuracy: 0.8865 - val_precision: 0.9063 -
val_recall: 0.8704
Epoch 18/50
1252/1252 [=====] - ETA: 0s - loss: 0.3514 -
accuracy: 0.8961 - precision: 0.9221 - recall: 0.8704
Epoch 18: val_loss did not improve from 0.39161
1252/1252 [=====] - 332s 265ms/step - loss:
0.3514 - accuracy: 0.8961 - precision: 0.9221 - recall: 0.8704 -
val_loss: 0.4102 - val_accuracy: 0.8825 - val_precision: 0.9034 -
val_recall: 0.8681
Epoch 19/50
1252/1252 [=====] - ETA: 0s - loss: 0.3429 -
accuracy: 0.9002 - precision: 0.9248 - recall: 0.8770
Epoch 19: val_loss did not improve from 0.39161
1252/1252 [=====] - 331s 265ms/step - loss:
```

```

0.3429 - accuracy: 0.9002 - precision: 0.9248 - recall: 0.8770 -
val_loss: 0.4019 - val_accuracy: 0.8836 - val_precision: 0.9038 -
val_recall: 0.8719
Epoch 20/50
1252/1252 [=====] - ETA: 0s - loss: 0.3349 -
accuracy: 0.9015 - precision: 0.9249 - recall: 0.8783
Epoch 20: val_loss did not improve from 0.39161
1252/1252 [=====] - 332s 265ms/step - loss:
0.3349 - accuracy: 0.9015 - precision: 0.9249 - recall: 0.8783 -
val_loss: 0.4019 - val_accuracy: 0.8838 - val_precision: 0.9038 -
val_recall: 0.8710
Epoch 21/50
1252/1252 [=====] - ETA: 0s - loss: 0.3274 -
accuracy: 0.9046 - precision: 0.9287 - recall: 0.8826
Epoch 21: val_loss did not improve from 0.39161
1252/1252 [=====] - 331s 264ms/step - loss:
0.3274 - accuracy: 0.9046 - precision: 0.9287 - recall: 0.8826 -
val_loss: 0.4057 - val_accuracy: 0.8854 - val_precision: 0.9049 -
val_recall: 0.8708

```

3.4 Avaliação

O melhor modelo obtido durante o processo de treino é carregado e avaliado utilizando o dataset de teste. Aqui é mostrado os valores das métricas de accuracy, loss, precision e recall obtidas pelo modelo nas imagens de teste.

```

# Carregar o melhor modelo obtido durante o processo de treino
model = keras.models.load_model('models/IC_T_FE_DA.keras')

# Avaliar o modelo
test_loss, test_acc, test_precision, test_recall =
model.evaluate(test_dataset)

print("Test Accuracy: " + str(test_acc))
print("Test Loss: " + str(test_loss))
print("Test Precision: " + str(test_precision))
print("Test Recall: " + str(test_recall))

313/313 [=====] - 23s 71ms/step - loss:
0.4027 - accuracy: 0.8824 - precision: 0.9044 - recall: 0.8627
Test Accuracy: 0.8823999762535095
Test Loss: 0.4027153551578522
Test Precision: 0.9043924808502197
Test Recall: 0.8626999855041504

```

4. Análise de resultados

4.1 Evolução das métricas durante o processo de treino

É utilizado gráficos para melhor compreender de que maneira as métricas, nomeadamente a accuracy, loss, precision e recall, foram evoluindo ao longo do processo de treino.

É possível observar que:

- O problema de overfitting no modelo sem Data Augmentation foi mitigado
- O modelo não melhora substancialmente o desempenho para as imagens do dataset de validação ao longo do processo de treino

```
# Buscar as métricas
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
precision = history.history['precision']
val_precision = history.history['val_precision']
recall = history.history['recall']
val_recall = history.history['val_recall']

# Calcular o número de épocas que foram realizadas
epochs = range(1, len(acc) + 1)

# Gráfico da accuracy
plt.plot(epochs, acc, 'blue', label='Training Accuracy')
plt.plot(epochs, val_acc, 'orange', label='Validation Accuracy')
plt.title('Training and validation Accuracy evolution')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.figure()

# Gráfico da loss
plt.plot(epochs, loss, 'blue', label='Training Loss')
plt.plot(epochs, val_loss, 'orange', label='Validation Loss')
plt.title('Training and validation Loss evolution')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.figure()

# Gráfico da precision
plt.plot(epochs, precision, 'blue', label='Training Precision')
plt.plot(epochs, val_precision, 'orange', label='Validation Precision')
plt.title('Training and validation Precision evolution')
```

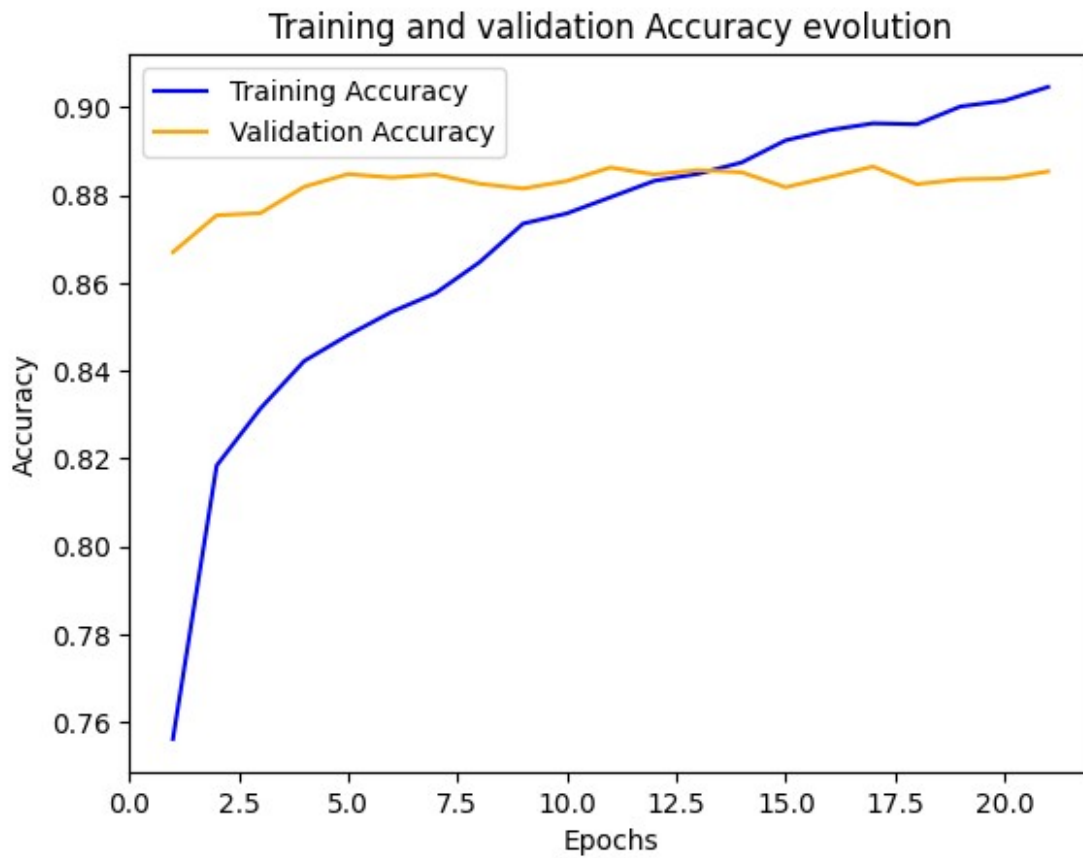
```

plt.xlabel('Epochs')
plt.ylabel('Precision')
plt.legend()
plt.figure()

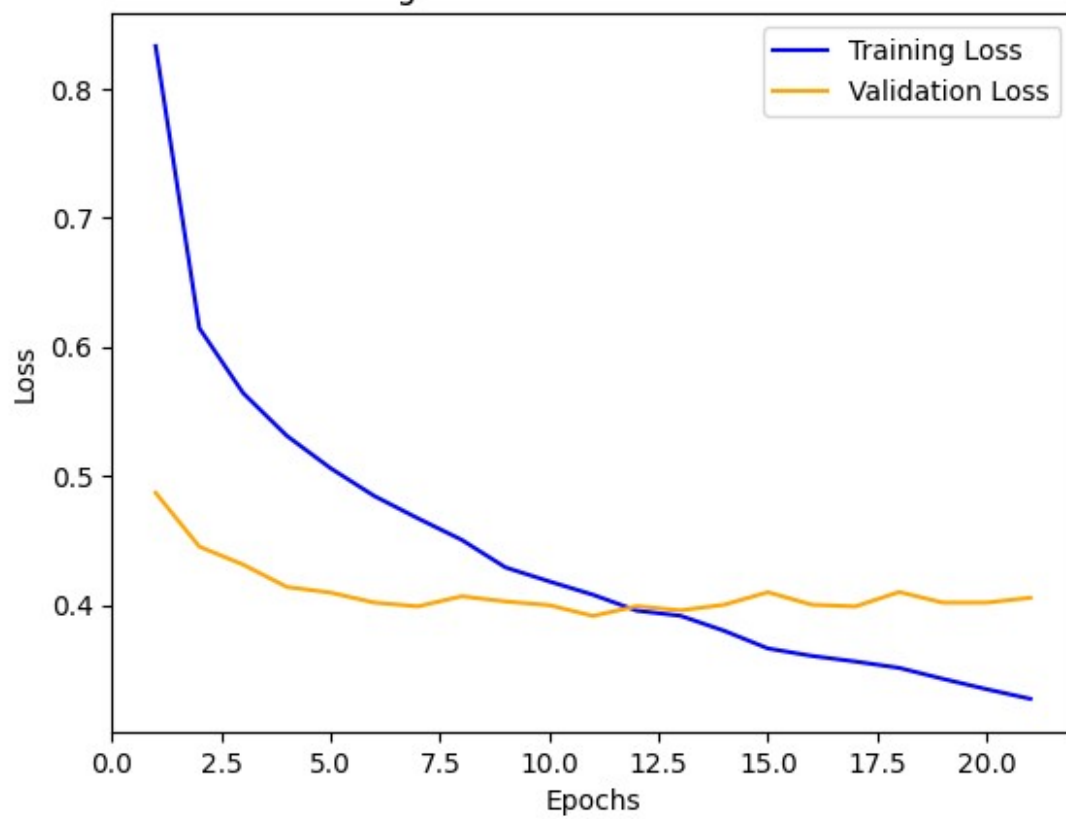
# Gráfico do recall
plt.plot(epochs, recall, 'blue', label='Training Recall')
plt.plot(epochs, val_recall, 'orange', label='Validation Recall')
plt.title('Training and validation Recall evolution')
plt.xlabel('Epochs')
plt.ylabel('Recall')
plt.legend()

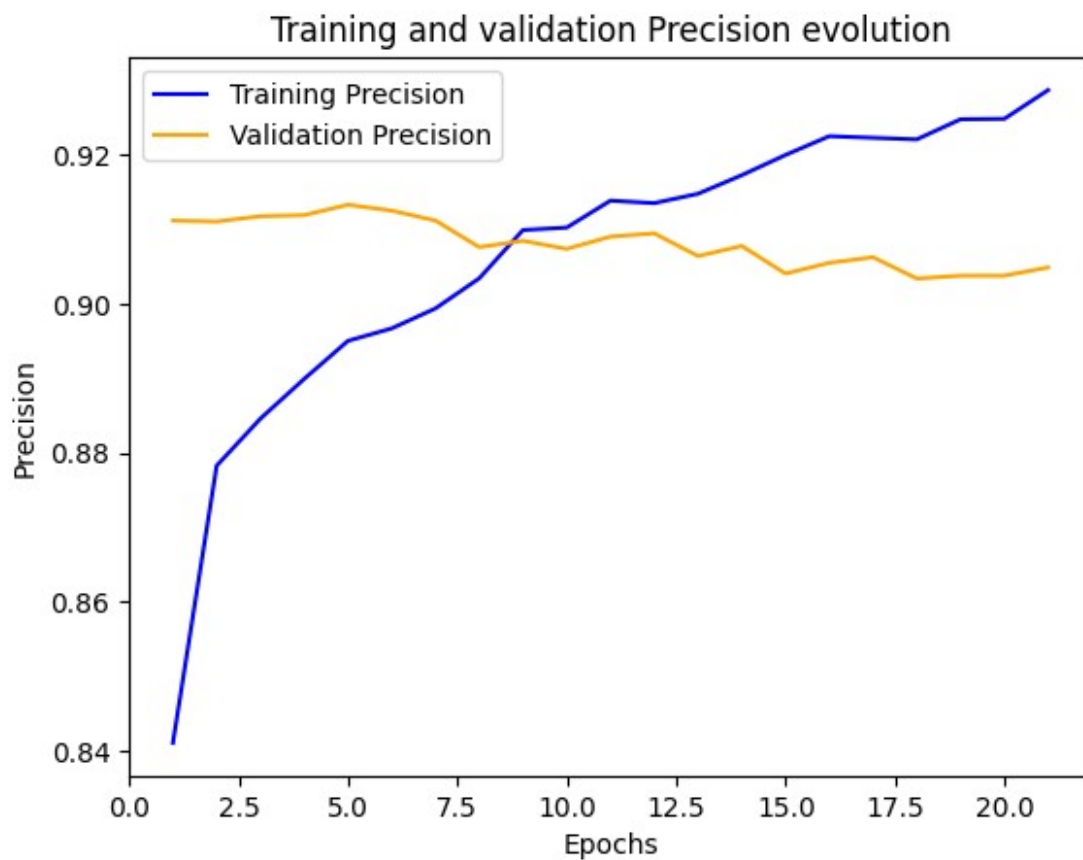
plt.show()

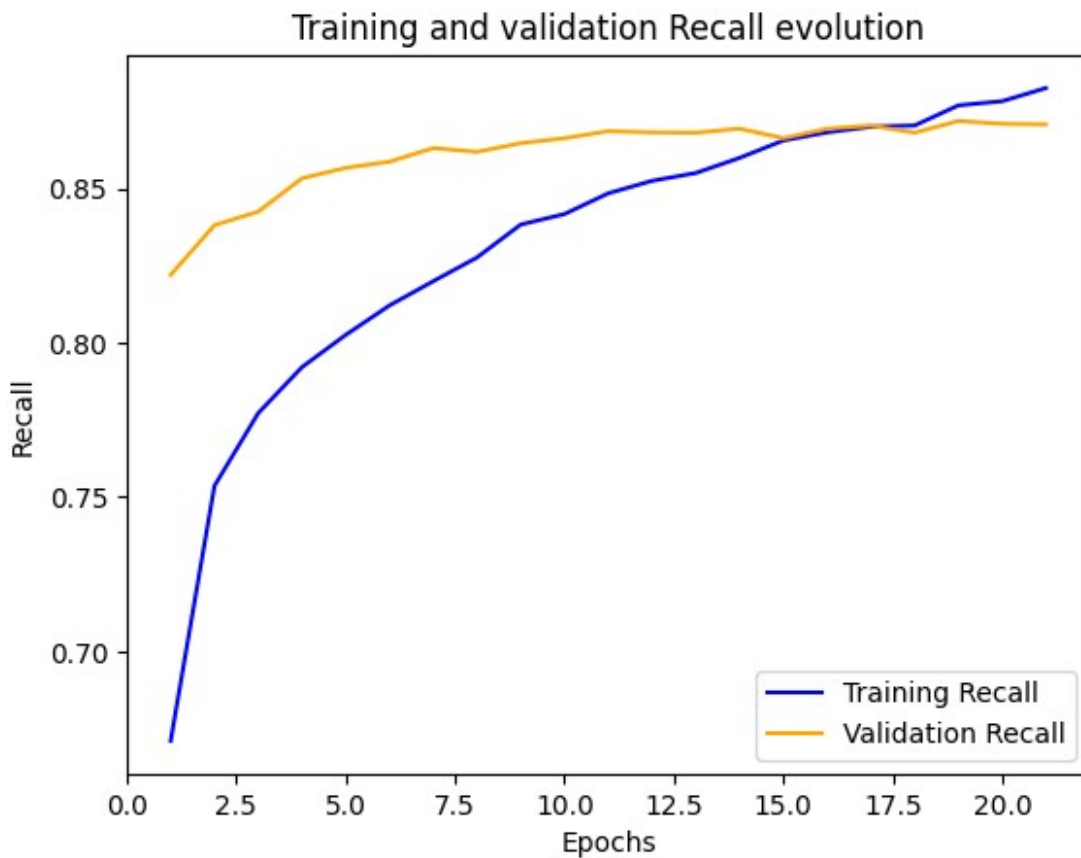
```



Training and validation Loss evolution







4.2 Desempenho no dataset de teste

De modo a compreender o real desempenho do modelo precisamos avaliar este utilizando o dataset de teste (que contém imagens que o este nunca viu anteriormente).

São feitas, e guardadas, previsões do modelo sobre o dataset de teste para, posteriormente, ser criado um classification report, que nos vai permitir analisar a taxa de acerto global e a precision, recall e f1-score para cada classe. Para além disso, é, também, construída uma matriz de confusão que, vai permitir ilustrar de uma outra maneira as previsões (vai ser possível ver, por exemplo, que quando a imagem pertencia à classe "dog", o modelo achou n vezes que a imagem pertencia à classe "cat").

Com isto, podemos concluir que:

- O modelo obtem resultados satisfatórios
- É de realçar que a maior dificuldade do modelo, apontada na versão deste treinada sem Data Augmentation, foi ligeiramente reduzida.

```
# Fazer previsões para o dataset de teste
predictions = model.predict(test_dataset)
predicted_classes = np.argmax(predictions, axis=1)

# Obter as classes verdadeiras de cada imagem no dataset de teste
```

```

true_classes = []
for images, labels in test_dataset:
    true_classes.extend(np.argmax(labels.numpy(), axis=1))
true_classes = np.array(true_classes)

# Criar o classification report
report = classification_report(true_classes, predicted_classes,
target_names=class_names)
print(report)

# Mostrar a matriz de confusão
cm = confusion_matrix(true_classes, predicted_classes)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion Matrix')
plt.ylabel('True Class')
plt.xlabel('Predicted Class')
plt.show()

```

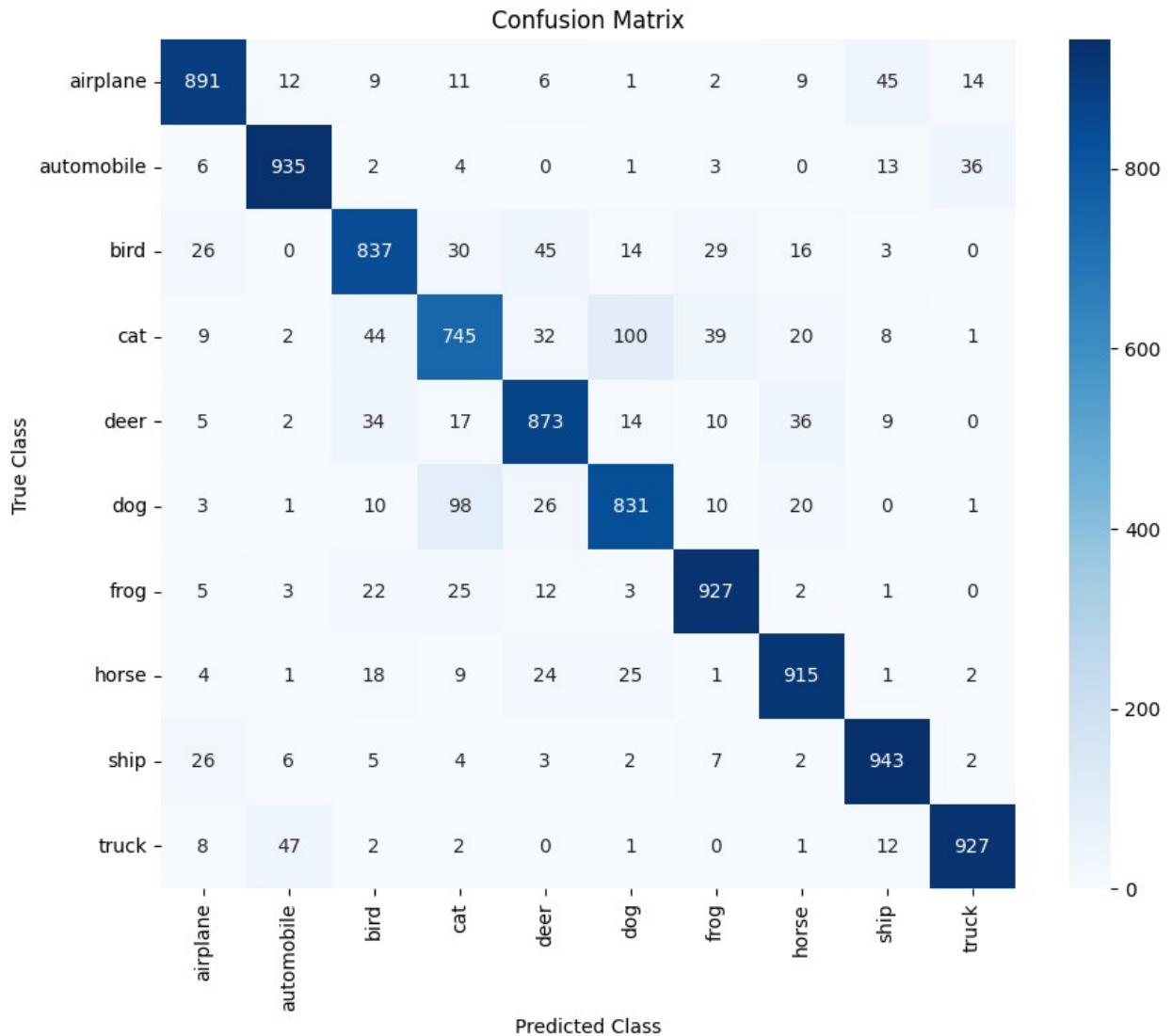
```

313/313 [=====] - 22s 69ms/step

```

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

airplane	0.91	0.89	0.90	1000
automobile	0.93	0.94	0.93	1000
bird	0.85	0.84	0.84	1000
cat	0.79	0.74	0.77	1000
deer	0.86	0.87	0.86	1000
dog	0.84	0.83	0.83	1000
frog	0.90	0.93	0.91	1000
horse	0.90	0.92	0.91	1000
ship	0.91	0.94	0.93	1000
truck	0.94	0.93	0.93	1000
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000



4.3 Visualização de previsões

Aqui fazemos a visualização de imagens tal como anteriormente, mas introduzimos a previsão do modelo para cada uma das imagens, sendo possível visualizar, também, a classe real de cada imagem.

```
displayed_classes = set()

plt.figure(figsize=(12, 6)) # Ajustar o tamanho das imagens

for data_batch, label_batch in test_dataset:
    for i in range(len(label_batch)):
        true_class_idx = np.argmax(label_batch[i])
        true_label = class_names[true_class_idx]

        if true_class_idx not in displayed_classes:
```

```

        displayed_classes.add(true_class_idx)

        plt.subplot(2, 5, len(displayed_classes))

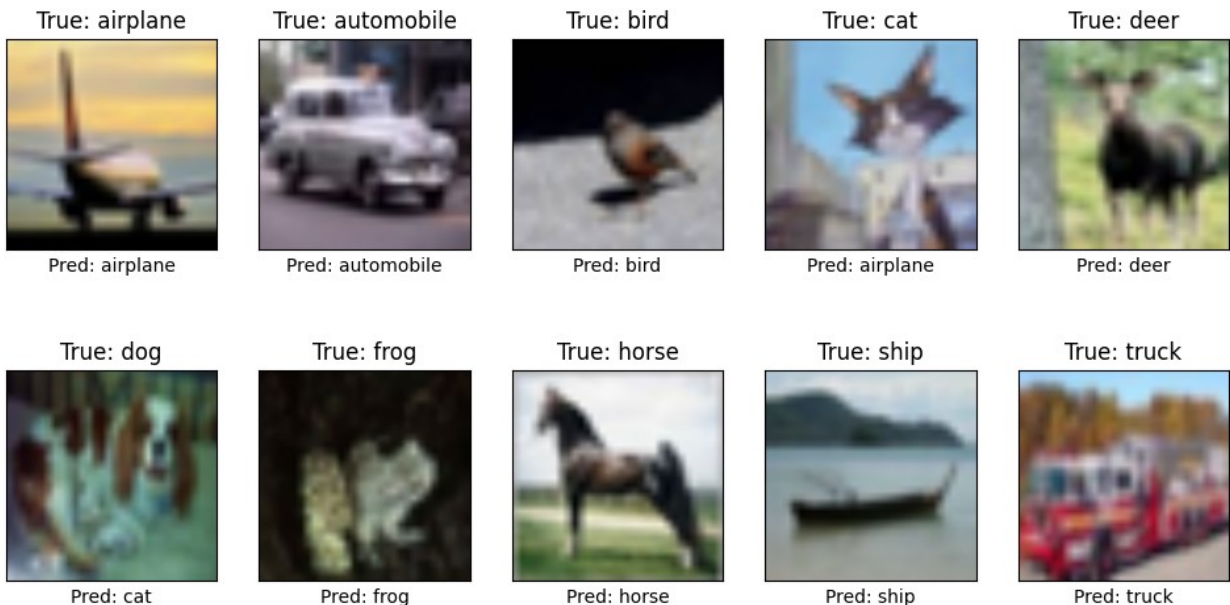
        pred_label = model.predict(np.expand_dims(data_batch[i],
axis=0), verbose=0)
        pred_label = class_names[np.argmax(pred_label)]

        plt.title("True: " + true_label)
        plt.xlabel("Pred: " + pred_label)
        plt.imshow(data_batch[i].numpy().astype('uint8'))
        plt.xticks([])
        plt.yticks([])

# Stop condition para no caso de já terem sido mostrada 10
imagens
        if len(displayed_classes) == 10:
            break
        if len(displayed_classes) == 10:
            break

plt.show()

```



Conclusão

O problema de overfitting, presente na versão do modelo treinada sem Data Augmentation foi, como prespectivado, resolvido.

Apesar disto, é importante notar que, para as imagens do dataset de validação o modelo não apresenta melhorias, no que toca ao seu desempenho, ao longo do processo de treino.

Bibliografia

<https://www.markdownguide.org/basic-syntax/>

<https://www.tensorflow.org/>

<https://keras.io/api/applications/>

<https://keras.io/api/optimizers/>

https://keras.io/api/data_loading/

<https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/>

https://nchlis.github.io/2017_08_10/page.html

<https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7>