ImageClassification-AI: Modelo S

Versão A **Data Augmentation**

Modelo de raiz com o intuito de seguir o que foi lecionado durante o semestre. O objetivo deste modelo, nesta versão, é servir de base e patamar para outros modelos de raiz.

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 atribuidas 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
```

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 beneficial 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.

```
# Remove tensorflow warnings (because of Tensorflow 2.10.0)
tf.get_logger().setLevel('ERROR')
```

1.3 Tratamento de dados

Definição das classes do problema:

- Tamanho das imagens RGB (32x32x3 pixeis)
- 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 paramêtros 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, importar 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 construido através da concatenação de quatro datasets de treino mais pequenos (cada um relativo a uma das diretoria 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 mesmo)

```
class_names = []

IMG_SIZE = 32
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 Definir 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 32x32 pixeis (32x32x3)
- Cada batch de labels tem 10 classes

```
for data_batch, label_batch in train_dataset:
    print('Shape de cada data batch: ', data_batch.shape)
    print('Shape de cada label batch: ', label_batch.shape)
    break
```

```
Shape de cada data batch:  (32, 32, 32, 3)
Shape de cada label batch:  (32, 10)
```

2.3 - Normalização

Visualização da normalização dos pixeis:

- Divisão do valor de cada pixel por 255
- Operação definida, posteriormente, na construção do modelo e, feita durante o processo de treino para cada imagem de modo a que, cada pixel tenha um valor associado que pertença ao intervalo de [0,1].
- Mostrar como o modelo irá interpretar cada imagem (os valores de cada pixel)

```
iterator = train_dataset.as_numpy_iterator()
batch = iterator.next()
batch[0] / 255 # normalizar (feito mais à frente relo rescalling)
```

```
array([[[[0.5686275 , 0.78039217, 0.96862745],
         [0.57254905, 0.78431374, 0.98039216],
         [0.5686275 , 0.7921569 , 0.99215686],
         ...,
         [0.56078434, 0.78039217, 1.        ],
         [0.5568628 , 0.78039217, 0.99215686],
         [0.5647059 , 0.7882353 , 0.98039216]],

        [[0.5568628 , 0.77254903, 0.9647059 ],
         [0.5647059 , 0.78431374, 0.98039216],
         [0.54901963, 0.78039217, 0.98039216],
         ...,
         [0.5529412 , 0.7764706 , 1.        ],
         [0.5529412 , 0.7764706 , 0.99607843],
         [0.5529412 , 0.78039217, 0.9882353 ]],

        [[0.54901963, 0.77254903, 0.972549  ],
         [0.56078434, 0.7882353 , 0.99215686],
         [0.5411765 , 0.78039217, 0.9843137 ],
         ...,
         [0.54509807, 0.78039217, 1.        ],
         [0.54509807, 0.78039217, 1.        ],
         [0.54901963, 0.7882353 , 0.99607843]],

        ...,

        [[0.56078434, 0.7882353 , 0.9607843 ],
         [0.57254905, 0.8039216 , 0.98039216],
         [0.57254905, 0.79607844, 0.98039216],
         ...,
         [0.56078434, 0.8       , 1.        ],
         [0.56078434, 0.8039216 , 1.        ],
         [0.56078434, 0.8039216 , 0.9882353 ]],

        [[0.5647059 , 0.78431374, 0.9647059 ],
         [0.5764706 , 0.79607844, 0.9843137 ],
         [0.5764706 , 0.7921569 , 0.9843137 ],
         ...,
         [0.5568628 , 0.8039216 , 1.        ],
         [0.5568628 , 0.8039216 , 0.99607843],
```

```
         [0.5647059 , 0.8       , 0.9843137 ]],

        [[0.57254905, 0.79607844, 0.972549  ],
         [0.58431375, 0.8117647 , 0.99215686],
         [0.5803922 , 0.8039216 , 0.99215686],
         ...,
         [0.5647059 , 0.8117647 , 1.        ],
         [0.5686275 , 0.80784315, 1.        ],
         [0.5803922 , 0.80784315, 0.99215686]]],


       [[[0.99607843, 0.99215686, 0.99607843],
         [0.95686275, 0.9647059 , 0.92156863],
         [0.9411765 , 0.9490196 , 0.90588236],
         ...,
         [0.95686275, 0.95686275, 0.9372549 ],
         [0.95686275, 0.95686275, 0.9254902 ],
         [0.9607843 , 0.9607843 , 0.9411765 ]],

        [[0.98039216, 0.98039216, 0.9843137 ],
         [0.5529412 , 0.58431375, 0.5058824 ],
         [0.3764706 , 0.42745098, 0.32941177],
         ...,
         [0.4627451 , 0.46666667, 0.42352942],
         [0.46666667, 0.4745098 , 0.41568628],
         [0.49019608, 0.49411765, 0.45490196]],

        [[0.9490196 , 0.95686275, 0.9490196 ],
         [0.42745098, 0.48235294, 0.3647059 ],
         [0.2784314 , 0.36078432, 0.20784314],
         ...,
         [0.3764706 , 0.3882353 , 0.3254902 ],
         [0.35686275, 0.36862746, 0.29411766],
         [0.4       , 0.40784314, 0.34901962]],

        ...,

        [[0.5803922 , 0.5764706 , 0.5764706 ],
         [0.5058824 , 0.49019608, 0.49019608],
         [0.50980395, 0.49019608, 0.49019608],
         ...,
         [0.49019608, 0.49411765, 0.49411765],
         [0.6       , 0.60784316, 0.6039216 ],
         [0.96862745, 0.9764706 , 0.972549  ]],

        [[0.69803923, 0.69411767, 0.69803923],
         [0.6392157 , 0.61960787, 0.6313726 ],
         [0.6431373 , 0.61960787, 0.6313726 ],
         ...,
         [0.6313726 , 0.6392157 , 0.63529414],
```

```
         [0.70980394, 0.7176471 , 0.7137255 ],
         [0.972549  , 0.98039216, 0.9764706 ]],

        [[0.94509804, 0.94509804, 0.94509804],
         [0.94509804, 0.92941177, 0.9411765 ],
         [0.94509804, 0.92941177, 0.94509804],
         ...,
         [0.92941177, 0.9372549 , 0.93333334],
         [0.9372549 , 0.94509804, 0.9411765 ],
         [0.98039216, 0.9843137 , 0.9843137 ]]],


       [[[0.13725491, 0.44705883, 0.27058825],
         [0.14901961, 0.39607844, 0.22745098],
         [0.19215687, 0.4       , 0.22352941],
         ...,
         [0.11764706, 0.47058824, 0.2509804 ],
         [0.14117648, 0.47843137, 0.27450982],
         [0.19215687, 0.43137255, 0.24313726]],

        [[0.18039216, 0.49803922, 0.3137255 ],
         [0.16078432, 0.42745098, 0.24705882],
         [0.17254902, 0.39607844, 0.21960784],
         ...,
         [0.12941177, 0.43529412, 0.23137255],
         [0.11764706, 0.44313726, 0.24313726],
         [0.19215687, 0.41960785, 0.24705882]],

        [[0.16862746, 0.47843137, 0.29411766],
         [0.15294118, 0.42745098, 0.24705882],
         [0.16078432, 0.38039216, 0.21176471],
         ...,
         [0.16862746, 0.41568628, 0.23137255],
         [0.13725491, 0.44705883, 0.2509804 ],
         [0.16078432, 0.41960785, 0.23921569]],

        ...,

        [[0.5411765 , 0.6745098 , 0.627451  ],
         [0.50980395, 0.627451  , 0.5568628 ],
         [0.52156866, 0.62352943, 0.5529412 ],
         ...,
         [0.70980394, 0.69411767, 0.5803922 ],
         [0.7058824 , 0.69411767, 0.58431375],
         [0.69411767, 0.69803923, 0.6039216 ]],

        [[0.50980395, 0.6156863 , 0.5686275 ],
         [0.5176471 , 0.627451  , 0.5647059 ],
         [0.52156866, 0.627451  , 0.5764706 ],
         ...,
```

```
         [0.7294118 , 0.7372549 , 0.6117647 ],
         [0.7254902 , 0.73333335, 0.6156863 ],
         [0.7372549 , 0.74509805, 0.654902  ]],

        [[0.54509807, 0.6627451 , 0.6       ],
         [0.54901963, 0.6745098 , 0.6117647 ],
         [0.5529412 , 0.6901961 , 0.6509804 ],
         ...,
         [0.6156863 , 0.6509804 , 0.5176471 ],
         [0.6117647 , 0.63529414, 0.5058824 ],
         [0.64705884, 0.6392157 , 0.5294118 ]]],


       ...,


       [[[0.3254902 , 0.6313726 , 0.7647059 ],
         [0.32941177, 0.6313726 , 0.7647059 ],
         [0.33333334, 0.6392157 , 0.76862746],
         ...,
         [0.34117648, 0.627451  , 0.7411765 ],
         [0.32941177, 0.61960787, 0.7372549 ],
         [0.31764707, 0.6117647 , 0.73333335]],

        [[0.3372549 , 0.6509804 , 0.7764706 ],
         [0.34117648, 0.6509804 , 0.7764706 ],
         [0.34509805, 0.654902  , 0.78039217],
         ...,
         [0.37254903, 0.654902  , 0.75686276],
         [0.36862746, 0.6509804 , 0.75686276],
         [0.34509805, 0.63529414, 0.7490196 ]],

        [[0.34509805, 0.654902  , 0.7764706 ],
         [0.34509805, 0.654902  , 0.77254903],
         [0.3529412 , 0.65882355, 0.7764706 ],
         ...,
         [0.3647059 , 0.64705884, 0.7529412 ],
         [0.35686275, 0.6431373 , 0.7490196 ],
         [0.34117648, 0.6313726 , 0.74509805]],

        ...,

        [[0.53333336, 0.627451  , 0.54901963],
         [0.5372549 , 0.627451  , 0.5529412 ],
         [0.5372549 , 0.627451  , 0.54901963],
         ...,
         [0.5372549 , 0.627451  , 0.5411765 ],
         [0.5372549 , 0.62352943, 0.5411765 ],
         [0.53333336, 0.61960787, 0.5372549 ]],
```

```
        [[0.5294118 , 0.62352943, 0.54901963],
         [0.5294118 , 0.61960787, 0.54901963],
         [0.5372549 , 0.62352943, 0.54901963],
         ...,
         [0.53333336, 0.6313726 , 0.54901963],
         [0.5372549 , 0.63529414, 0.5529412 ],
         [0.5372549 , 0.63529414, 0.5529412 ]],

        [[0.50980395, 0.6117647 , 0.54509807],
         [0.5137255 , 0.6156863 , 0.54901963],
         [0.5254902 , 0.627451  , 0.54901963],
         ...,
         [0.5137255 , 0.6156863 , 0.5294118 ],
         [0.50980395, 0.6039216 , 0.5254902 ],
         [0.50980395, 0.60784316, 0.5294118 ]]],


       [[[0.4       , 0.30980393, 0.22352941],
         [0.3882353 , 0.29803923, 0.21176471],
         [0.39215687, 0.3019608 , 0.21568628],
         ...,
         [0.34117648, 0.2       , 0.07450981],
         [0.34509805, 0.2       , 0.07058824],
         [0.3529412 , 0.2       , 0.06666667]],

        [[0.31764707, 0.19607843, 0.07843138],
         [0.32156864, 0.19607843, 0.08235294],
         [0.32156864, 0.19607843, 0.08235294],
         ...,
         [0.34117648, 0.2       , 0.07450981],
         [0.34117648, 0.19607843, 0.07058824],
         [0.34901962, 0.19607843, 0.06666667]],

        [[0.34509805, 0.19215687, 0.05882353],
         [0.34509805, 0.19215687, 0.0627451 ],
         [0.34117648, 0.19215687, 0.0627451 ],
         ...,
         [0.3372549 , 0.19607843, 0.07058824],
         [0.34509805, 0.2       , 0.07450981],
         [0.3529412 , 0.2       , 0.06666667]],

        ...,

        [[0.43529412, 0.23921569, 0.08235294],
         [0.40392157, 0.23137255, 0.07843138],
         [0.6117647 , 0.47843137, 0.3372549 ],
         ...,
         [0.4392157 , 0.24705882, 0.08627451],
         [0.4509804 , 0.25882354, 0.09411765],
         [0.43529412, 0.24705882, 0.07450981]],
```

```
        [[0.43137255, 0.23921569, 0.08627451],
         [0.4117647 , 0.23921569, 0.08235294],
         [0.4627451 , 0.3019608 , 0.16078432],
         ...,
         [0.44705883, 0.2509804 , 0.09019608],
         [0.43137255, 0.2509804 , 0.09019608],
         [0.43529412, 0.26666668, 0.10980392]],

        [[0.42745098, 0.23921569, 0.09019608],
         [0.4       , 0.22745098, 0.06666667],
         [0.41568628, 0.23529412, 0.09019608],
         ...,
         [0.45882353, 0.26666668, 0.10196079],
         [0.43137255, 0.2627451 , 0.10980392],
         [0.38431373, 0.2509804 , 0.11764706]]],


       [[[0.5921569 , 0.6156863 , 0.63529414],
         [0.5803922 , 0.62352943, 0.627451  ],
         [0.54901963, 0.5921569 , 0.5921569 ],
         ...,
         [0.06666667, 0.07058824, 0.04705882],
         [0.02745098, 0.03137255, 0.01176471],
         [0.02745098, 0.03529412, 0.01568628]],

        [[0.5647059 , 0.5882353 , 0.6392157 ],
         [0.5764706 , 0.6156863 , 0.64705884],
         [0.58431375, 0.62352943, 0.63529414],
         ...,
         [0.04705882, 0.04313726, 0.03529412],
         [0.05098039, 0.04705882, 0.03529412],
         [0.06666667, 0.0627451 , 0.05490196]],

        [[0.59607846, 0.6313726 , 0.6666667 ],
         [0.5921569 , 0.6392157 , 0.6745098 ],
         [0.59607846, 0.6313726 , 0.6509804 ],
         ...,
         [0.1254902 , 0.11372549, 0.09411765],
         [0.14901961, 0.13725491, 0.12156863],
         [0.15686275, 0.14509805, 0.12941177]],

        ...,

        [[0.3647059 , 0.4117647 , 0.4117647 ],
         [0.36862746, 0.41568628, 0.41960785],
         [0.38431373, 0.42745098, 0.4392157 ],
         ...,
         [0.27450982, 0.3137255 , 0.2901961 ],
         [0.12941177, 0.16862746, 0.12941177],
```

```
         [0.16078432, 0.20392157, 0.16862746]],

        [[0.39215687, 0.43529412, 0.4509804 ],
         [0.34509805, 0.3882353 , 0.4       ],
         [0.34901962, 0.39215687, 0.40392157],
         ...,
         [0.22352941, 0.2627451 , 0.24705882],
         [0.21568628, 0.25882354, 0.22352941],
         [0.2784314 , 0.32156864, 0.29411766]],

        [[0.3529412 , 0.39607844, 0.42745098],
         [0.34117648, 0.38039216, 0.40784314],
         [0.3529412 , 0.39607844, 0.41568628],
         ...,
         [0.27450982, 0.3254902 , 0.3137255 ],
         [0.31764707, 0.36862746, 0.34117648],
         [0.34509805, 0.39607844, 0.37254903]]], 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)
        plt.title(class_names[np.argmax(label_batch[i])])
        plt.imshow(data_batch[i].numpy().astype('uint8'))
        plt.xticks([])
        plt.yticks([])
    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

Apesar deste modelo ter o objetivo de seguir aquilo que foi lecionado ao longo do semestre, é necessário alterar a arquitetura do modelo treinado durante as aulas devido ao facto de estarem a ser utilizadas imagens mais pequenas. Caso isso não fosse precavido, os feature maps criados pelo modelo iriam ter tamanhos inválidos.

Na arquitetura deste modelo temos:

- A aplicação das operações de Data Augmentation

- Como supramencionado, a normalização dos valores de cada pixel da imagem
- Três blocos de layers convolucionais:
  - Cada um com uma layer convulocional
  - A quantidade de filtros em cada camada vai aumentado progressivamente de 32 filtros até 128
  - É utilizada a função de ativação ReLu
  - No final de cada bloco é feito o MaxPooling do feature map até aquele momento, com um filtro de 2x2 (que irá reduzir o tamanho de feature map em metade e, no caso de o valor ser decimal, irá arrendondar o tamanho às unidades)
  - É utilizada a técnica de regularização BatchNormalization com o intuito de manter consistente a distribuição dos valores que saem dos outputs de cada layer e que entram na próxima
- Bloco de classificação:
  - É utilizado o Flatten para transformar os valores obtidos até aqui num vetor 1D
  - É utilizada uma camada densa com 128 filtros que, irá receber os valores da ultima camada convolucional aos quais vai aplicar a BatchNormalization e a técnica de Dropout (que consiste em excluir, aleatóriamente, x * 100 % dos neurónios anteriores, sendo x o valor que passamos por paramêtro). O Dropout é utilizado especialmente para combater o overfitting.
  - É utilizada uma outra camada densa, com 10 filtros (relativos à quantidade de classes do problema), 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, combater 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 paramêtros que este envolve.

```
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))

x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)

# 1st Convolutional Layer
x = layers.Conv2D(filters=32, kernel_size=3)(x)
x = layers.BatchNormalization()(x)
x = layers.Activation("relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)

# 2nd Convolutional Layer
x = layers.Conv2D(filters=64, kernel_size=3)(x)
x = layers.BatchNormalization()(x)
x = layers.Activation("relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)

# 3rd Convolutional Layer
```

```
x = layers.Conv2D(filters=128, kernel_size=3)(x)
x = layers.BatchNormalization()(x)
x = layers.Activation("relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Flatten()(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.BatchNormalization()(x)
x = layers.Dropout(0.5)(x)

outputs = layers.Dense(10, activation="softmax",
kernel_regularizer=regularizers.l2(0.01))(x)
model = keras.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, 30, 30, 32)        896

 batch_normalization (BatchN  (None, 30, 30, 32)       128
 ormalization)

 activation (Activation)     (None, 30, 30, 32)        0

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 batch_normalization_1 (Batc  (None, 13, 13, 64)       256
 hNormalization)

 activation_1 (Activation)   (None, 13, 13, 64)        0

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 128)         73856

 batch_normalization_2 (Batc  (None, 4, 4, 128)        512
 hNormalization)
```

```
 activation_2 (Activation)   (None, 4, 4, 128)          0

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)         0
 2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 128)               65664

 batch_normalization_3 (Batc  (None, 128)              512
 hNormalization)

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 161,610
Trainable params: 160,906
Non-trainable params: 704
_________________________________________________________________
```

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 metricas de acerto (neste caso o "CategorialAccuracy" em vez do Accuracy normal devido ao contexto do problema), precisão e recall. É, ainda, importante referir que inicialmente era para ser incluida uma métrica de calculo 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 metrica. Realizaram-se experiências utilizando a métrica F1-Score do Tensorflow Addons mas, os resultados não foram satisfatórios.

Como este modelo visa se aproximar ao máximo, dentro dos possíveis, aos exemplos lecionados ao longo do semestre foi escolhido o optimizador RMSprop com um learning rate de 0.0001.

```
# Compilar o modelo
model.compile(
    loss='categorical_crossentropy',
    optimizer=optimizers.RMSprop(learning_rate=1e-4),
    metrics=[
                tf.keras.metrics.CategoricalAccuracy(name='accuracy'),
                tf.keras.metrics.Precision(name='precision'),
                tf.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 necessáriamente 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) utilzando:

- O dataset de treino
- 100 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_S_A.keras',
        save_best_only = True,
        monitor='val_loss',
        verbose=1
    )
]


# Train the model
history = model.fit(train_dataset,
epochs=100,validation_data=val_dataset, callbacks=callbacks)
```

```
Epoch 1/100
1251/1252 [============================>.] - ETA: 0s - loss: 2.3120 -
accuracy: 0.2974 - precision: 0.3810 - recall: 0.1657
Epoch 1: val_loss improved from inf to 1.79692, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 62s 47ms/step - loss:
2.3118 - accuracy: 0.2975 - precision: 0.3811 - recall: 0.1657 -
val_loss: 1.7969 - val_accuracy: 0.4294 - val_precision: 0.5710 -
```

```
val_recall: 0.2843
Epoch 2/100
1252/1252 [==============================] - ETA: 0s - loss: 1.8580 -
accuracy: 0.3909 - precision: 0.5160 - recall: 0.2331
Epoch 2: val_loss improved from 1.79692 to 1.66222, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 59s 47ms/step - loss:
1.8580 - accuracy: 0.3909 - precision: 0.5160 - recall: 0.2331 -
val_loss: 1.6622 - val_accuracy: 0.4727 - val_precision: 0.6241 -
val_recall: 0.3355
Epoch 3/100
1251/1252 [==============================>.] - ETA: 0s - loss: 1.6608 -
accuracy: 0.4410 - precision: 0.5885 - recall: 0.2649
Epoch 3: val_loss improved from 1.66222 to 1.56694, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 59s 47ms/step - loss:
1.6609 - accuracy: 0.4410 - precision: 0.5886 - recall: 0.2649 -
val_loss: 1.5669 - val_accuracy: 0.4897 - val_precision: 0.6248 -
val_recall: 0.3561
Epoch 4/100
1252/1252 [==============================] - ETA: 0s - loss: 1.5545 -
accuracy: 0.4705 - precision: 0.6331 - recall: 0.2869
Epoch 4: val_loss improved from 1.56694 to 1.43259, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 59s 47ms/step - loss:
1.5545 - accuracy: 0.4705 - precision: 0.6331 - recall: 0.2869 -
val_loss: 1.4326 - val_accuracy: 0.5284 - val_precision: 0.6738 -
val_recall: 0.3911
Epoch 5/100
1251/1252 [==============================>.] - ETA: 0s - loss: 1.4691 -
accuracy: 0.4998 - precision: 0.6686 - recall: 0.3074
Epoch 5: val_loss improved from 1.43259 to 1.37356, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 60s 48ms/step - loss:
1.4691 - accuracy: 0.4999 - precision: 0.6685 - recall: 0.3074 -
val_loss: 1.3736 - val_accuracy: 0.5412 - val_precision: 0.6864 -
val_recall: 0.4085
Epoch 6/100
1252/1252 [==============================] - ETA: 0s - loss: 1.4109 -
accuracy: 0.5218 - precision: 0.6921 - recall: 0.3219
Epoch 6: val_loss improved from 1.37356 to 1.26439, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 60s 48ms/step - loss:
1.4109 - accuracy: 0.5218 - precision: 0.6921 - recall: 0.3219 -
val_loss: 1.2644 - val_accuracy: 0.5764 - val_precision: 0.7271 -
val_recall: 0.4375
Epoch 7/100
1252/1252 [==============================] - ETA: 0s - loss: 1.3606 -
accuracy: 0.5381 - precision: 0.7128 - recall: 0.3441
```

```
Epoch 7: val_loss improved from 1.26439 to 1.26237, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 59s 47ms/step - loss:
1.3606 - accuracy: 0.5381 - precision: 0.7128 - recall: 0.3441 -
val_loss: 1.2624 - val_accuracy: 0.5777 - val_precision: 0.7258 -
val_recall: 0.4407
Epoch 8/100
1252/1252 [==============================] - ETA: 0s - loss: 1.3263 -
accuracy: 0.5534 - precision: 0.7251 - recall: 0.3598
Epoch 8: val_loss did not improve from 1.26237
1252/1252 [==============================] - 59s 47ms/step - loss:
1.3263 - accuracy: 0.5534 - precision: 0.7251 - recall: 0.3598 -
val_loss: 1.4077 - val_accuracy: 0.5422 - val_precision: 0.6692 -
val_recall: 0.4391
Epoch 9/100
1252/1252 [==============================] - ETA: 0s - loss: 1.2924 -
accuracy: 0.5608 - precision: 0.7339 - recall: 0.3727
Epoch 9: val_loss did not improve from 1.26237
1252/1252 [==============================] - 59s 47ms/step - loss:
1.2924 - accuracy: 0.5608 - precision: 0.7339 - recall: 0.3727 -
val_loss: 1.3265 - val_accuracy: 0.5581 - val_precision: 0.6852 -
val_recall: 0.4583
Epoch 10/100
1252/1252 [==============================] - ETA: 0s - loss: 1.2602 -
accuracy: 0.5739 - precision: 0.7413 - recall: 0.3891
Epoch 10: val_loss improved from 1.26237 to 1.24451, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 59s 47ms/step - loss:
1.2602 - accuracy: 0.5739 - precision: 0.7413 - recall: 0.3891 -
val_loss: 1.2445 - val_accuracy: 0.5851 - val_precision: 0.7102 -
val_recall: 0.4668
Epoch 11/100
1252/1252 [==============================] - ETA: 0s - loss: 1.2358 -
accuracy: 0.5842 - precision: 0.7511 - recall: 0.4001
Epoch 11: val_loss improved from 1.24451 to 1.22891, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 60s 48ms/step - loss:
1.2358 - accuracy: 0.5842 - precision: 0.7511 - recall: 0.4001 -
val_loss: 1.2289 - val_accuracy: 0.5885 - val_precision: 0.7186 -
val_recall: 0.4795
Epoch 12/100
1252/1252 [==============================] - ETA: 0s - loss: 1.2120 -
accuracy: 0.5951 - precision: 0.7552 - recall: 0.4164
Epoch 12: val_loss did not improve from 1.22891
1252/1252 [==============================] - 59s 47ms/step - loss:
1.2120 - accuracy: 0.5951 - precision: 0.7552 - recall: 0.4164 -
val_loss: 1.2794 - val_accuracy: 0.5756 - val_precision: 0.6850 -
val_recall: 0.4804
Epoch 13/100
```

```
1251/1252 [============================>.] - ETA: 0s - loss: 1.1920 -
accuracy: 0.5997 - precision: 0.7572 - recall: 0.4280
Epoch 13: val_loss did not improve from 1.22891
1252/1252 [==============================] - 59s 47ms/step - loss:
1.1922 - accuracy: 0.5997 - precision: 0.7571 - recall: 0.4279 -
val_loss: 1.2466 - val_accuracy: 0.5895 - val_precision: 0.7048 -
val_recall: 0.4993
Epoch 14/100
1252/1252 [==============================] - ETA: 0s - loss: 1.1742 -
accuracy: 0.6064 - precision: 0.7619 - recall: 0.4414
Epoch 14: val_loss improved from 1.22891 to 1.13636, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 59s 47ms/step - loss:
1.1742 - accuracy: 0.6064 - precision: 0.7619 - recall: 0.4414 -
val_loss: 1.1364 - val_accuracy: 0.6203 - val_precision: 0.7306 -
val_recall: 0.5069
Epoch 15/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.1617 -
accuracy: 0.6095 - precision: 0.7636 - recall: 0.4464
Epoch 15: val_loss did not improve from 1.13636
1252/1252 [==============================] - 59s 47ms/step - loss:
1.1616 - accuracy: 0.6096 - precision: 0.7637 - recall: 0.4465 -
val_loss: 1.2847 - val_accuracy: 0.5821 - val_precision: 0.6763 -
val_recall: 0.4911
Epoch 16/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.1488 -
accuracy: 0.6162 - precision: 0.7646 - recall: 0.4550
Epoch 16: val_loss did not improve from 1.13636
1252/1252 [==============================] - 59s 47ms/step - loss:
1.1486 - accuracy: 0.6163 - precision: 0.7647 - recall: 0.4551 -
val_loss: 1.1452 - val_accuracy: 0.6171 - val_precision: 0.7303 -
val_recall: 0.5122
Epoch 17/100
1252/1252 [==============================] - ETA: 0s - loss: 1.1253 -
accuracy: 0.6229 - precision: 0.7731 - recall: 0.4664
Epoch 17: val_loss did not improve from 1.13636
1252/1252 [==============================] - 59s 47ms/step - loss:
1.1253 - accuracy: 0.6229 - precision: 0.7731 - recall: 0.4664 -
val_loss: 1.1369 - val_accuracy: 0.6244 - val_precision: 0.7318 -
val_recall: 0.5280
Epoch 18/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.1171 -
accuracy: 0.6257 - precision: 0.7741 - recall: 0.4729
Epoch 18: val_loss improved from 1.13636 to 1.05086, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 59s 47ms/step - loss:
1.1172 - accuracy: 0.6256 - precision: 0.7741 - recall: 0.4728 -
val_loss: 1.0509 - val_accuracy: 0.6501 - val_precision: 0.7617 -
val_recall: 0.5476
```

```
Epoch 19/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.1074 -
accuracy: 0.6304 - precision: 0.7738 - recall: 0.4779
Epoch 19: val_loss did not improve from 1.05086
1252/1252 [==============================] - 59s 47ms/step - loss:
1.1074 - accuracy: 0.6303 - precision: 0.7738 - recall: 0.4779 -
val_loss: 1.0765 - val_accuracy: 0.6404 - val_precision: 0.7512 -
val_recall: 0.5443
Epoch 20/100
1252/1252 [==============================] - ETA: 0s - loss: 1.0894 -
accuracy: 0.6360 - precision: 0.7785 - recall: 0.4887
Epoch 20: val_loss did not improve from 1.05086
1252/1252 [==============================] - 61s 48ms/step - loss:
1.0894 - accuracy: 0.6360 - precision: 0.7785 - recall: 0.4887 -
val_loss: 1.0734 - val_accuracy: 0.6429 - val_precision: 0.7456 -
val_recall: 0.5512
Epoch 21/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.0797 -
accuracy: 0.6382 - precision: 0.7776 - recall: 0.4923
Epoch 21: val_loss did not improve from 1.05086
1252/1252 [==============================] - 62s 49ms/step - loss:
1.0797 - accuracy: 0.6382 - precision: 0.7776 - recall: 0.4924 -
val_loss: 1.0961 - val_accuracy: 0.6391 - val_precision: 0.7385 -
val_recall: 0.5469
Epoch 22/100
1252/1252 [==============================] - ETA: 0s - loss: 1.0627 -
accuracy: 0.6471 - precision: 0.7845 - recall: 0.5021
Epoch 22: val_loss improved from 1.05086 to 1.02700, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 61s 49ms/step - loss:
1.0627 - accuracy: 0.6471 - precision: 0.7845 - recall: 0.5021 -
val_loss: 1.0270 - val_accuracy: 0.6565 - val_precision: 0.7661 -
val_recall: 0.5603
Epoch 23/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.0592 -
accuracy: 0.6474 - precision: 0.7826 - recall: 0.5075
Epoch 23: val_loss did not improve from 1.02700
1252/1252 [==============================] - 61s 48ms/step - loss:
1.0592 - accuracy: 0.6474 - precision: 0.7825 - recall: 0.5074 -
val_loss: 1.1055 - val_accuracy: 0.6364 - val_precision: 0.7387 -
val_recall: 0.5494
Epoch 24/100
1252/1252 [==============================] - ETA: 0s - loss: 1.0468 -
accuracy: 0.6500 - precision: 0.7869 - recall: 0.5118
Epoch 24: val_loss did not improve from 1.02700
1252/1252 [==============================] - 60s 48ms/step - loss:
1.0468 - accuracy: 0.6500 - precision: 0.7869 - recall: 0.5118 -
val_loss: 1.1324 - val_accuracy: 0.6326 - val_precision: 0.7215 -
val_recall: 0.5487
```

```
Epoch 25/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.0391 -
accuracy: 0.6540 - precision: 0.7858 - recall: 0.5165
Epoch 25: val_loss improved from 1.02700 to 0.99407, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 60s 48ms/step - loss:
1.0391 - accuracy: 0.6540 - precision: 0.7857 - recall: 0.5165 -
val_loss: 0.9941 - val_accuracy: 0.6713 - val_precision: 0.7770 -
val_recall: 0.5746
Epoch 26/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.0296 -
accuracy: 0.6590 - precision: 0.7876 - recall: 0.5238
Epoch 26: val_loss did not improve from 0.99407
1252/1252 [==============================] - 60s 48ms/step - loss:
1.0296 - accuracy: 0.6590 - precision: 0.7877 - recall: 0.5239 -
val_loss: 1.0722 - val_accuracy: 0.6502 - val_precision: 0.7494 -
val_recall: 0.5628
Epoch 27/100
1252/1252 [==============================] - ETA: 0s - loss: 1.0262 -
accuracy: 0.6615 - precision: 0.7900 - recall: 0.5269
Epoch 27: val_loss did not improve from 0.99407
1252/1252 [==============================] - 60s 48ms/step - loss:
1.0262 - accuracy: 0.6615 - precision: 0.7900 - recall: 0.5269 -
val_loss: 1.0253 - val_accuracy: 0.6597 - val_precision: 0.7644 -
val_recall: 0.5719
Epoch 28/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.0110 -
accuracy: 0.6623 - precision: 0.7917 - recall: 0.5332
Epoch 28: val_loss did not improve from 0.99407
1252/1252 [==============================] - 60s 47ms/step - loss:
1.0108 - accuracy: 0.6623 - precision: 0.7917 - recall: 0.5332 -
val_loss: 1.1424 - val_accuracy: 0.6251 - val_precision: 0.7103 -
val_recall: 0.5496
Epoch 29/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.0000 -
accuracy: 0.6690 - precision: 0.7955 - recall: 0.5392
Epoch 29: val_loss did not improve from 0.99407
1252/1252 [==============================] - 60s 48ms/step - loss:
1.0000 - accuracy: 0.6690 - precision: 0.7954 - recall: 0.5393 -
val_loss: 1.0055 - val_accuracy: 0.6680 - val_precision: 0.7621 -
val_recall: 0.5831
Epoch 30/100
1251/1252 [============================>.] - ETA: 0s - loss: 1.0029 -
accuracy: 0.6676 - precision: 0.7914 - recall: 0.5402
Epoch 30: val_loss did not improve from 0.99407
1252/1252 [==============================] - 60s 47ms/step - loss:
1.0028 - accuracy: 0.6676 - precision: 0.7913 - recall: 0.5401 -
val_loss: 1.0215 - val_accuracy: 0.6659 - val_precision: 0.7601 -
val_recall: 0.5861
```

```
Epoch 31/100
1251/1252 [============================>.] - ETA: 0s - loss: 0.9847 -
accuracy: 0.6737 - precision: 0.7968 - recall: 0.5459
Epoch 31: val_loss improved from 0.99407 to 0.96206, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 60s 48ms/step - loss:
0.9846 - accuracy: 0.6737 - precision: 0.7968 - recall: 0.5459 -
val_loss: 0.9621 - val_accuracy: 0.6803 - val_precision: 0.7755 -
val_recall: 0.5985
Epoch 32/100
1251/1252 [============================>.] - ETA: 0s - loss: 0.9814 -
accuracy: 0.6756 - precision: 0.7981 - recall: 0.5493
Epoch 32: val_loss did not improve from 0.96206
1252/1252 [==============================] - 59s 47ms/step - loss:
0.9815 - accuracy: 0.6756 - precision: 0.7981 - recall: 0.5492 -
val_loss: 1.0109 - val_accuracy: 0.6684 - val_precision: 0.7629 -
val_recall: 0.5866
Epoch 33/100
1251/1252 [============================>.] - ETA: 0s - loss: 0.9774 -
accuracy: 0.6771 - precision: 0.7979 - recall: 0.5539
Epoch 33: val_loss did not improve from 0.96206
1252/1252 [==============================] - 61s 49ms/step - loss:
0.9774 - accuracy: 0.6771 - precision: 0.7979 - recall: 0.5540 -
val_loss: 1.1185 - val_accuracy: 0.6341 - val_precision: 0.7149 -
val_recall: 0.5646
Epoch 34/100
1252/1252 [==============================] - ETA: 0s - loss: 0.9641 -
accuracy: 0.6802 - precision: 0.7997 - recall: 0.5572
Epoch 34: val_loss did not improve from 0.96206
1252/1252 [==============================] - 65s 52ms/step - loss:
0.9641 - accuracy: 0.6802 - precision: 0.7997 - recall: 0.5572 -
val_loss: 1.0814 - val_accuracy: 0.6474 - val_precision: 0.7321 -
val_recall: 0.5760
Epoch 35/100
1252/1252 [==============================] - ETA: 0s - loss: 0.9583 -
accuracy: 0.6821 - precision: 0.7997 - recall: 0.5606
Epoch 35: val_loss did not improve from 0.96206
1252/1252 [==============================] - 65s 52ms/step - loss:
0.9583 - accuracy: 0.6821 - precision: 0.7997 - recall: 0.5606 -
val_loss: 0.9798 - val_accuracy: 0.6796 - val_precision: 0.7724 -
val_recall: 0.5984
Epoch 36/100
1251/1252 [============================>.] - ETA: 0s - loss: 0.9525 -
accuracy: 0.6846 - precision: 0.7999 - recall: 0.5655
Epoch 36: val_loss improved from 0.96206 to 0.95651, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 65s 52ms/step - loss:
0.9527 - accuracy: 0.6846 - precision: 0.7999 - recall: 0.5654 -
val_loss: 0.9565 - val_accuracy: 0.6780 - val_precision: 0.7665 -
```

```
val_recall: 0.5958
Epoch 37/100
1252/1252 [==============================] - ETA: 0s - loss: 0.9475 -
accuracy: 0.6855 - precision: 0.8031 - recall: 0.5681
Epoch 37: val_loss improved from 0.95651 to 0.93947, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 65s 52ms/step - loss:
0.9475 - accuracy: 0.6855 - precision: 0.8031 - recall: 0.5681 -
val_loss: 0.9395 - val_accuracy: 0.6887 - val_precision: 0.7790 -
val_recall: 0.6076
Epoch 38/100
1251/1252 [==============================>.] - ETA: 0s - loss: 0.9415 -
accuracy: 0.6867 - precision: 0.8039 - recall: 0.5699
Epoch 38: val_loss did not improve from 0.93947
1252/1252 [==============================] - 66s 53ms/step - loss:
0.9417 - accuracy: 0.6866 - precision: 0.8039 - recall: 0.5698 -
val_loss: 1.0346 - val_accuracy: 0.6656 - val_precision: 0.7468 -
val_recall: 0.5900
Epoch 39/100
1252/1252 [==============================] - ETA: 0s - loss: 0.9362 -
accuracy: 0.6894 - precision: 0.8055 - recall: 0.5748
Epoch 39: val_loss did not improve from 0.93947
1252/1252 [==============================] - 66s 53ms/step - loss:
0.9362 - accuracy: 0.6894 - precision: 0.8055 - recall: 0.5748 -
val_loss: 0.9416 - val_accuracy: 0.6902 - val_precision: 0.7778 -
val_recall: 0.6161
Epoch 40/100
1251/1252 [==============================>.] - ETA: 0s - loss: 0.9308 -
accuracy: 0.6904 - precision: 0.8067 - recall: 0.5805
Epoch 40: val_loss did not improve from 0.93947
1252/1252 [==============================] - 65s 52ms/step - loss:
0.9308 - accuracy: 0.6904 - precision: 0.8067 - recall: 0.5805 -
val_loss: 0.9919 - val_accuracy: 0.6717 - val_precision: 0.7586 -
val_recall: 0.6012
Epoch 41/100
1251/1252 [==============================>.] - ETA: 0s - loss: 0.9283 -
accuracy: 0.6938 - precision: 0.8079 - recall: 0.5795
Epoch 41: val_loss improved from 0.93947 to 0.92732, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 66s 53ms/step - loss:
0.9285 - accuracy: 0.6937 - precision: 0.8078 - recall: 0.5794 -
val_loss: 0.9273 - val_accuracy: 0.6943 - val_precision: 0.7827 -
val_recall: 0.6158
Epoch 42/100
1251/1252 [==============================>.] - ETA: 0s - loss: 0.9170 -
accuracy: 0.6977 - precision: 0.8129 - recall: 0.5872
Epoch 42: val_loss improved from 0.92732 to 0.91486, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 65s 52ms/step - loss:
```

```
0.9171 - accuracy: 0.6977 - precision: 0.8129 - recall: 0.5872 -
val_loss: 0.9149 - val_accuracy: 0.6991 - val_precision: 0.7904 -
val_recall: 0.6231
Epoch 43/100
1251/1252 [============================>.] - ETA: 0s - loss: 0.9142 -
accuracy: 0.6963 - precision: 0.8085 - recall: 0.5869
Epoch 43: val_loss did not improve from 0.91486
1252/1252 [==============================] - 64s 51ms/step - loss:
0.9140 - accuracy: 0.6964 - precision: 0.8086 - recall: 0.5869 -
val_loss: 0.9898 - val_accuracy: 0.6807 - val_precision: 0.7714 -
val_recall: 0.6067
Epoch 44/100
1251/1252 [============================>.] - ETA: 0s - loss: 0.9008 -
accuracy: 0.7006 - precision: 0.8109 - recall: 0.5932
Epoch 44: val_loss did not improve from 0.91486
1252/1252 [==============================] - 66s 52ms/step - loss:
0.9008 - accuracy: 0.7006 - precision: 0.8110 - recall: 0.5932 -
val_loss: 0.9610 - val_accuracy: 0.6825 - val_precision: 0.7706 -
val_recall: 0.6143
Epoch 45/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8983 -
accuracy: 0.7031 - precision: 0.8124 - recall: 0.5959
Epoch 45: val_loss improved from 0.91486 to 0.87935, saving model to
models\IC_S_A.keras
1252/1252 [==============================] - 64s 51ms/step - loss:
0.8983 - accuracy: 0.7031 - precision: 0.8124 - recall: 0.5959 -
val_loss: 0.8793 - val_accuracy: 0.7111 - val_precision: 0.7953 -
val_recall: 0.6326
Epoch 46/100
1251/1252 [============================>.] - ETA: 0s - loss: 0.8973 -
accuracy: 0.7033 - precision: 0.8085 - recall: 0.5970
Epoch 46: val_loss did not improve from 0.87935
1252/1252 [==============================] - 65s 52ms/step - loss:
0.8973 - accuracy: 0.7033 - precision: 0.8085 - recall: 0.5971 -
val_loss: 0.9491 - val_accuracy: 0.6905 - val_precision: 0.7709 -
val_recall: 0.6195
Epoch 47/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8928 -
accuracy: 0.7025 - precision: 0.8095 - recall: 0.5954
Epoch 47: val_loss did not improve from 0.87935
1252/1252 [==============================] - 65s 52ms/step - loss:
0.8928 - accuracy: 0.7025 - precision: 0.8095 - recall: 0.5954 -
val_loss: 0.9937 - val_accuracy: 0.6796 - val_precision: 0.7659 -
val_recall: 0.6066
Epoch 48/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8869 -
accuracy: 0.7043 - precision: 0.8127 - recall: 0.6008
Epoch 48: val_loss did not improve from 0.87935
1252/1252 [==============================] - 64s 51ms/step - loss:
```

```
0.8869 - accuracy: 0.7043 - precision: 0.8127 - recall: 0.6008 -
val_loss: 0.9399 - val_accuracy: 0.6930 - val_precision: 0.7721 -
val_recall: 0.6208
Epoch 49/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8791 -
accuracy: 0.7086 - precision: 0.8142 - recall: 0.6046
Epoch 49: val_loss did not improve from 0.87935
1252/1252 [==============================] - 65s 52ms/step - loss:
0.8791 - accuracy: 0.7086 - precision: 0.8142 - recall: 0.6046 -
val_loss: 0.9410 - val_accuracy: 0.6936 - val_precision: 0.7755 -
val_recall: 0.6256
Epoch 50/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8789 -
accuracy: 0.7097 - precision: 0.8151 - recall: 0.6079
Epoch 50: val_loss did not improve from 0.87935
1252/1252 [==============================] - 64s 51ms/step - loss:
0.8789 - accuracy: 0.7097 - precision: 0.8151 - recall: 0.6079 -
val_loss: 0.9009 - val_accuracy: 0.7037 - val_precision: 0.7839 -
val_recall: 0.6328
Epoch 51/100
1251/1252 [==============================>.] - ETA: 0s - loss: 0.8678 -
accuracy: 0.7155 - precision: 0.8189 - recall: 0.6137
Epoch 51: val_loss did not improve from 0.87935
1252/1252 [==============================] - 66s 53ms/step - loss:
0.8678 - accuracy: 0.7155 - precision: 0.8189 - recall: 0.6137 -
val_loss: 0.9150 - val_accuracy: 0.6987 - val_precision: 0.7807 -
val_recall: 0.6284
Epoch 52/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8704 -
accuracy: 0.7128 - precision: 0.8154 - recall: 0.6102
Epoch 52: val_loss did not improve from 0.87935
1252/1252 [==============================] - 66s 52ms/step - loss:
0.8704 - accuracy: 0.7128 - precision: 0.8154 - recall: 0.6102 -
val_loss: 1.0569 - val_accuracy: 0.6560 - val_precision: 0.7311 -
val_recall: 0.5911
Epoch 53/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8625 -
accuracy: 0.7151 - precision: 0.8178 - recall: 0.6142
Epoch 53: val_loss did not improve from 0.87935
1252/1252 [==============================] - 65s 52ms/step - loss:
0.8625 - accuracy: 0.7151 - precision: 0.8178 - recall: 0.6142 -
val_loss: 1.0141 - val_accuracy: 0.6693 - val_precision: 0.7535 -
val_recall: 0.6057
Epoch 54/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8589 -
accuracy: 0.7168 - precision: 0.8181 - recall: 0.6174
Epoch 54: val_loss did not improve from 0.87935
1252/1252 [==============================] - 64s 51ms/step - loss:
0.8589 - accuracy: 0.7168 - precision: 0.8181 - recall: 0.6174 -
```

```
val_loss: 0.9051 - val_accuracy: 0.7048 - val_precision: 0.7811 -
val_recall: 0.6354
Epoch 55/100
1251/1252 [============================>.] - ETA: 0s - loss: 0.8538 -
accuracy: 0.7157 - precision: 0.8182 - recall: 0.6157
Epoch 55: val_loss did not improve from 0.87935
1252/1252 [==============================] - 66s 53ms/step - loss:
0.8537 - accuracy: 0.7157 - precision: 0.8182 - recall: 0.6157 -
val_loss: 0.9329 - val_accuracy: 0.6906 - val_precision: 0.7725 -
val_recall: 0.6178
```

3.4 Avaliação

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

```
# Carregar o modelo
model = keras.models.load_model('models/IC_S_A_DA.keras')

# Avaliar o modelo utilizando o dataset de testes
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 [==============================] - 17s 6ms/step - loss: 0.8677
- accuracy: 0.7142 - precision: 0.7944 - recall: 0.6363
Test Accuracy: 0.7142000198364258
Test Loss: 0.867665708065033
Test Precision: 0.7943820357322693
Test Recall: 0.6363000273704529
```

4. Análise de resultados

4.1 Evolução das métricas durante o processo de treino

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

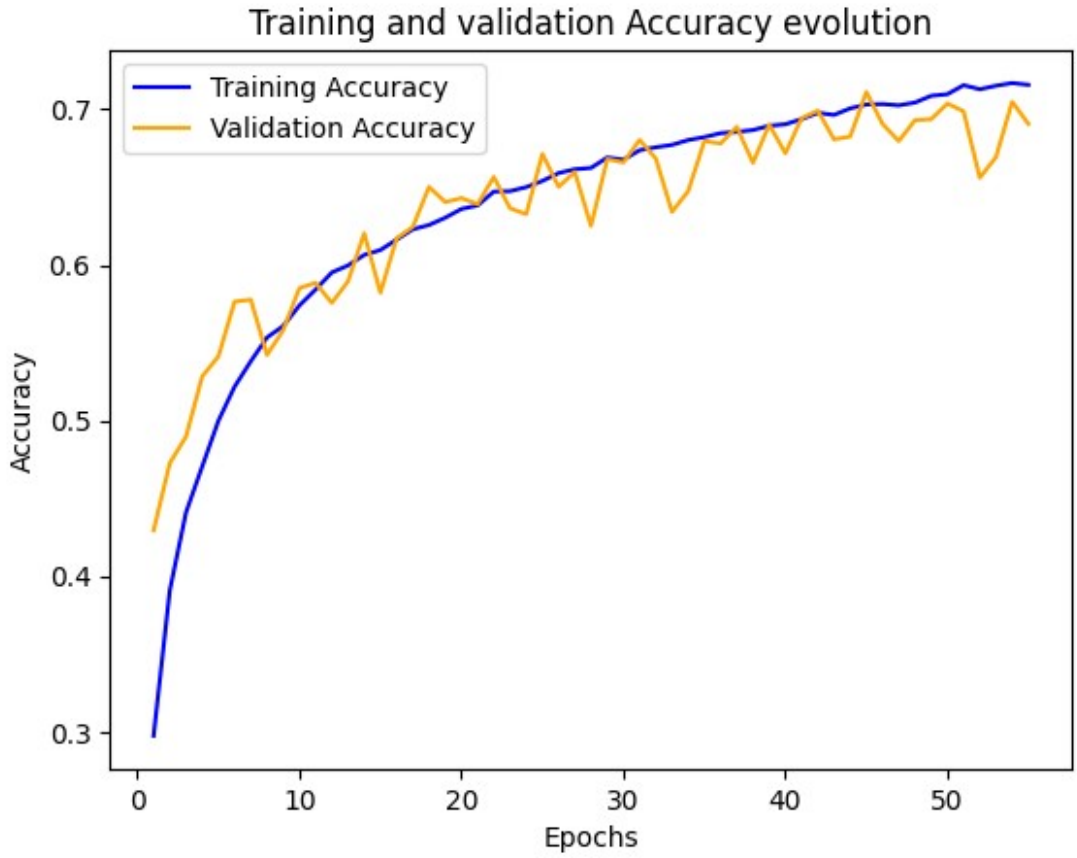
É possível visualizar que:

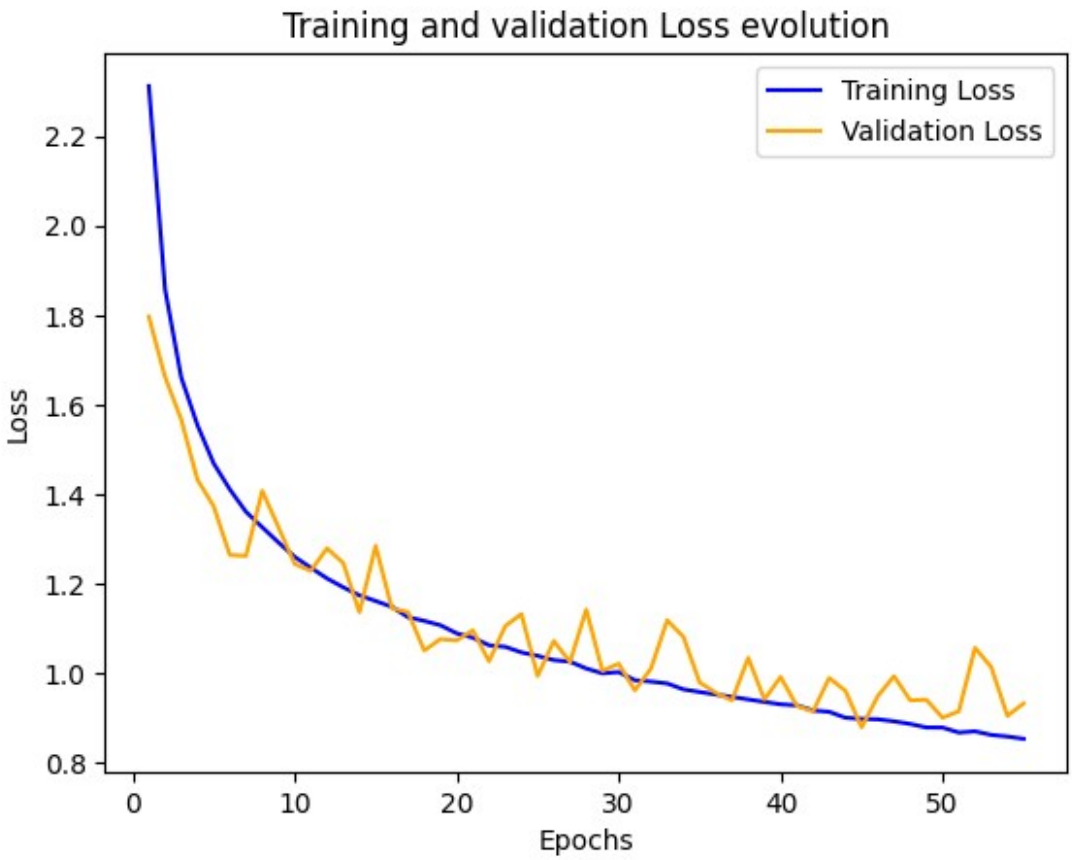
- O overfitting que este modelo sem Data Augmentation apresentava foi combatido com algum grau de sucesso
- O modelo apresenta sinais de má representação das imagens no dataset de validação (linhas bastante inconstantes e instáveis). Isto acontece porque as imagens do dataset de

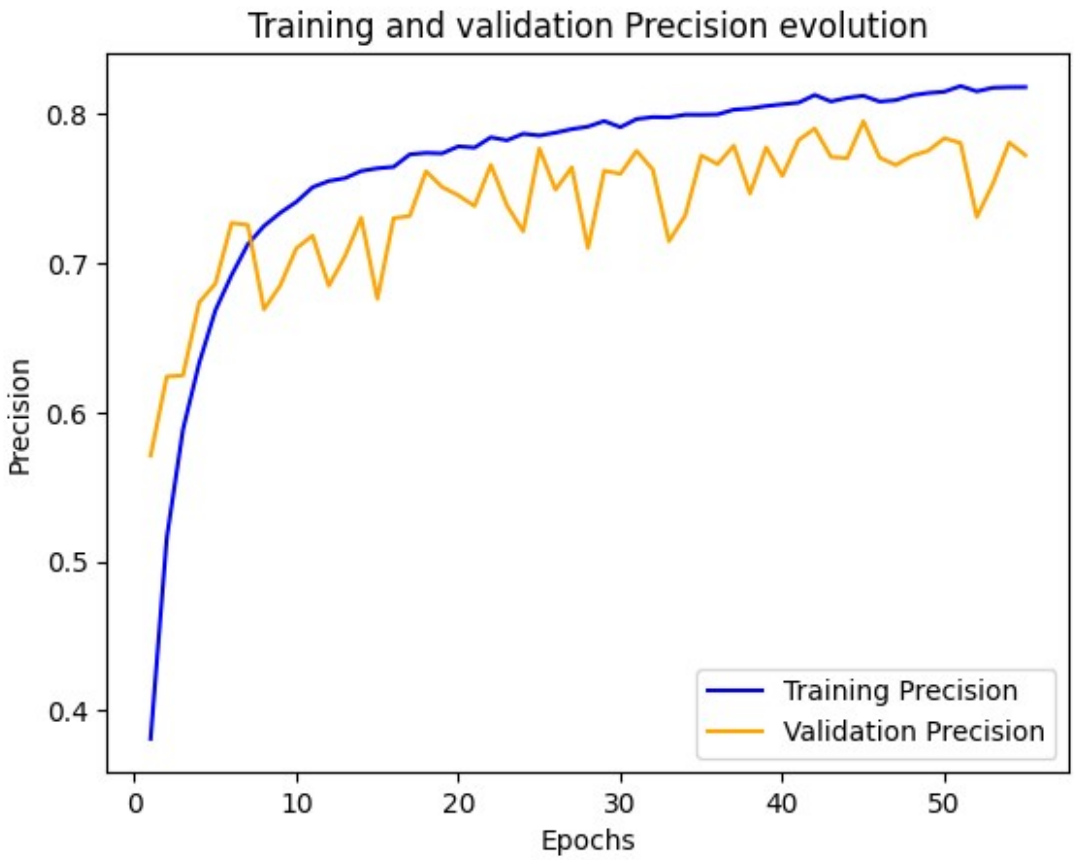
validação não oferecem informação suficiente para avaliar corretamente a capacidade de generalização do modelo.

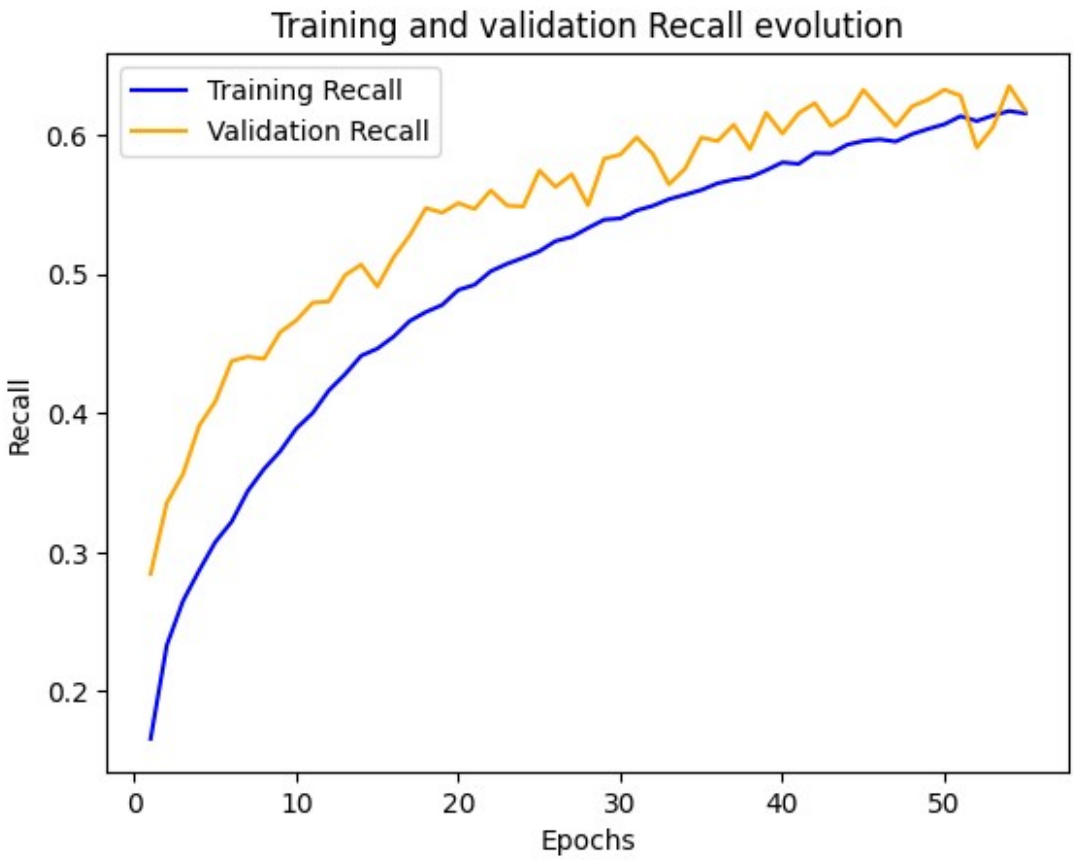
```
# Buscar as métricas
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
precision = history.history['precision']
val_precision = history.history['val_precision']
recall = history.history['recall']
val_recall = 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 Accuracy evolution
Training Accuracy
Validation Accuracy
Accuracy
0.3
0.4
0.5
0.6
0.7
0
10
20
30
40
50
Epochs


Training and validation Loss evolution
Training Loss
Validation Loss
Loss
Epochs
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0
10
20
30
40
50


Training and validation Precision evolution
Precision
Epochs
0.4
0.5
0.6
0.7
0.8
0
10
20
30
40
50
Training Precision
Validation Precision



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, construida 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 compreender que:

- As métricas com valores preocupantes no modelo sem Data Augmentation (precision, f1-score e recall) melhoraram, mas ainda não se obteve valores aos quais seria possíveis considerar de bons
- O modelo identifica bem as seguintes classes:
  - Automobile
  - Frog
  - Ship
  - Truck

  - Horse
- O modelo mostra dificuldades em identificar as seguintes classes:
  - Cat, apesar de ter ocorrido uma melhoria
  - Dog, apesar de ter ocorrido uma melhoria
  - Bird, piorou quando comparado com o modelo sem Data Augmentation

```
# 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()
```

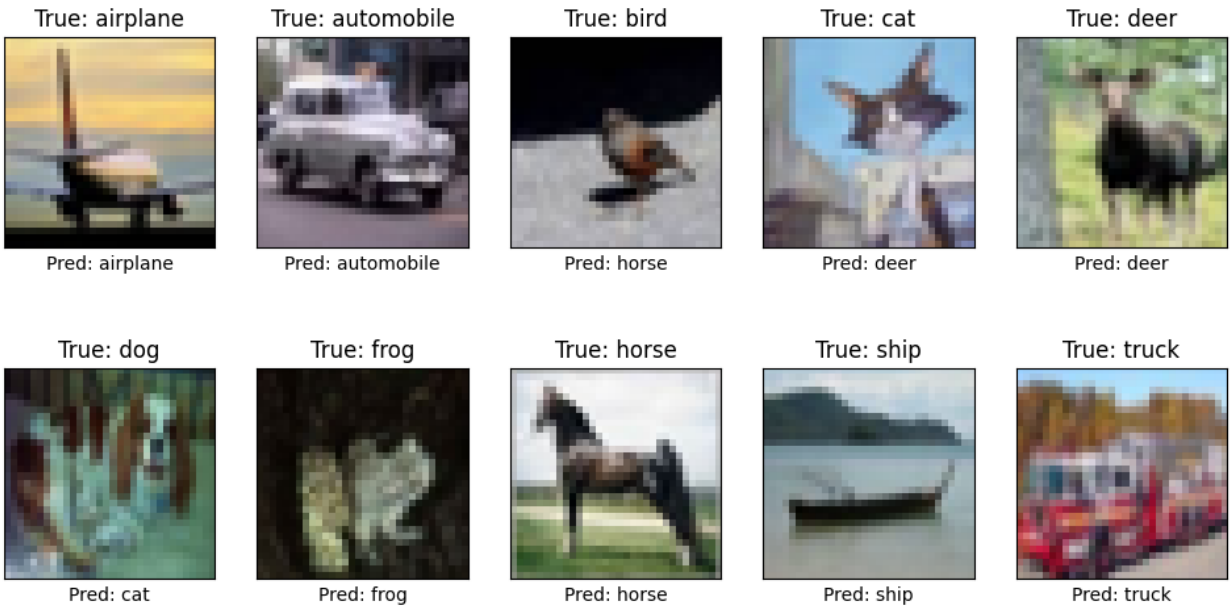
```
313/313 [==============================] - 2s 5ms/step
              precision    recall  f1-score   support

    airplane       0.80      0.73      0.76      1000
  automobile       0.83      0.86      0.85      1000
        bird       0.75      0.48      0.59      1000
         cat       0.57      0.55      0.56      1000
        deer       0.67      0.61      0.64      1000
         dog       0.74      0.56      0.64      1000
        frog       0.57      0.88      0.70      1000
       horse       0.71      0.81      0.76      1000
        ship       0.85      0.82      0.84      1000
       truck       0.73      0.83      0.78      1000

    accuracy                           0.71     10000
   macro avg       0.72      0.71      0.71     10000
weighted avg       0.72      0.71      0.71     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()
```



Conclusões

O problema de overfitting presente no modelo sem Data Augmentation foi mitigado. Com isto, houve uma melhoria de 1% na taxa de acerto do modelo, quando comparado com a sua versão sem Data Augmentation.

Apesar disto não é possível afirmar que o modelo melhorou significativamente ao acrescentar as operações de Data Augmentation.

Por fim, é importante realçar que a utilização das operações de Data Augmentation criou um problema na representação das imagens no dataset de validação. Para combater este problema é necessário fazer uma reflexão sobre alguns aspectos fundamentais do tais como a complexidade do modelo.

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