ImageClassification-AI: Modelo S

Versão B **Data Augmentation**

Modelo de raiz que utiliza a técnica de Data Augmentation com o intuito de obter os melhores resultados possíveis. Por consequência, este modelo vai almejar otimizar ao máximo a sua arquitetura.

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 de valor desativar algumas mensagens de warning dadas pelo Tensorflow, deixando o este apenas mostrar 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 (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([[[[0.8156863 , 0.8627451 , 0.80784315],
         [0.8117647 , 0.85882354, 0.8039216 ],
         [0.8117647 , 0.85882354, 0.8039216 ],
         ...,
         [0.7294118 , 0.7647059 , 0.6901961 ],
         [0.72156864, 0.7647059 , 0.6901961 ],
         [0.7254902 , 0.76862746, 0.7019608 ]],

        [[0.8039216 , 0.8509804 , 0.79607844],
         [0.79607844, 0.84313726, 0.7882353 ],
         [0.79607844, 0.84313726, 0.7882353 ],
         ...,
         [0.7254902 , 0.7607843 , 0.6862745 ],
         [0.7176471 , 0.75686276, 0.6862745 ],
         [0.7176471 , 0.7647059 , 0.69803923]],

        [[0.8039216 , 0.8509804 , 0.79607844],
         [0.79607844, 0.84313726, 0.7882353 ],
         [0.8       , 0.84705883, 0.7921569 ],
         ...,
         [0.72156864, 0.75686276, 0.68235296],
         [0.7137255 , 0.7529412 , 0.68235296],
         [0.7137255 , 0.75686276, 0.69411767]],

        ...,

        [[0.3647059 , 0.4392157 , 0.39215687],
         [0.3882353 , 0.45882353, 0.4117647 ],
         [0.41568628, 0.48235294, 0.4392157 ],
         ...,
         [0.12156863, 0.21568628, 0.14117648],
         [0.1254902 , 0.22352941, 0.14509805],
         [0.14901961, 0.24705882, 0.17254902]],

        [[0.37254903, 0.4509804 , 0.40392157],
         [0.3764706 , 0.45490196, 0.40784314],
         [0.36078432, 0.4392157 , 0.39215687],
         ...,
         [0.10196079, 0.20784314, 0.12941177],
         [0.10588235, 0.20392157, 0.12156863],
```

```
         [0.14901961, 0.24705882, 0.17254902]],

        [[0.32941177, 0.41568628, 0.3647059 ],
         [0.30588236, 0.39215687, 0.34117648],
         [0.3254902 , 0.40784314, 0.35686275],
         ...,
         [0.07843138, 0.19215687, 0.11372549],
         [0.08627451, 0.1882353 , 0.10588235],
         [0.12156863, 0.21960784, 0.14117648]]],


       [[[0.54509807, 0.5294118 , 0.39607844],
         [0.50980395, 0.49019608, 0.35686275],
         [0.50980395, 0.49411765, 0.35686275],
         ...,
         [0.40392157, 0.36862746, 0.2509804 ],
         [0.38039216, 0.34509805, 0.22352941],
         [0.34117648, 0.3137255 , 0.1882353 ]],

        [[0.61960787, 0.5764706 , 0.44705883],
         [0.6156863 , 0.57254905, 0.44313726],
         [0.59607846, 0.5529412 , 0.42352942],
         ...,
         [0.43137255, 0.39215687, 0.27058825],
         [0.44313726, 0.40392157, 0.28627452],
         [0.4       , 0.3647059 , 0.23921569]],

        [[0.6901961 , 0.62352943, 0.49803922],
         [0.6901961 , 0.62352943, 0.49803922],
         [0.65882355, 0.5921569 , 0.46666667],
         ...,
         [0.46666667, 0.42745098, 0.30980393],
         [0.4862745 , 0.44313726, 0.3254902 ],
         [0.4627451 , 0.42352942, 0.3019608 ]],

        ...,

        [[0.05098039, 0.01568628, 0.01960784],
         [0.02352941, 0.        , 0.00392157],
         [0.01960784, 0.01176471, 0.01960784],
         ...,
         [0.24705882, 0.3764706 , 0.12156863],
         [0.39215687, 0.5529412 , 0.2       ],
         [0.46666667, 0.6431373 , 0.22352941]],

        [[0.02745098, 0.00392157, 0.00392157],
         [0.01568628, 0.        , 0.00392157],
         [0.01568628, 0.01176471, 0.01568628],
         ...,
         [0.28627452, 0.42745098, 0.16470589],
```

```
         [0.4117647 , 0.59607846, 0.20392157],
         [0.43137255, 0.62352943, 0.18431373]],

        [[0.01568628, 0.        , 0.        ],
         [0.00784314, 0.        , 0.        ],
         [0.00784314, 0.00392157, 0.00784314],
         ...,
         [0.3764706 , 0.5372549 , 0.21960784],
         [0.39215687, 0.5882353 , 0.18431373],
         [0.4117647 , 0.6       , 0.18431373]]],


       [[[0.3764706 , 0.54509807, 0.7137255 ],
         [0.3882353 , 0.5411765 , 0.7137255 ],
         [0.40392157, 0.5411765 , 0.70980394],
         ...,
         [0.3372549 , 0.49411765, 0.7058824 ],
         [0.33333334, 0.49019608, 0.6862745 ],
         [0.32156864, 0.49019608, 0.6784314 ]],

        [[0.36862746, 0.54901963, 0.7294118 ],
         [0.3764706 , 0.54901963, 0.7294118 ],
         [0.39215687, 0.54509807, 0.7254902 ],
         ...,
         [0.32941177, 0.5019608 , 0.69411767],
         [0.3254902 , 0.49803922, 0.6745098 ],
         [0.31764707, 0.49411765, 0.65882355]],

        [[0.3647059 , 0.54901963, 0.7372549 ],
         [0.38039216, 0.54901963, 0.7372549 ],
         [0.39215687, 0.54901963, 0.73333335],
         ...,
         [0.3254902 , 0.50980395, 0.6862745 ],
         [0.32156864, 0.5058824 , 0.6666667 ],
         [0.31764707, 0.49803922, 0.6431373 ]],

        ...,

        [[0.38431373, 0.5411765 , 0.7176471 ],
         [0.4       , 0.5411765 , 0.7176471 ],
         [0.40392157, 0.5372549 , 0.7176471 ],
         ...,
         [0.3372549 , 0.5019608 , 0.6901961 ],
         [0.32941177, 0.49411765, 0.68235296],
         [0.3254902 , 0.49019608, 0.6784314 ]],

        [[0.38431373, 0.5411765 , 0.7137255 ],
         [0.4       , 0.5411765 , 0.7176471 ],
         [0.40392157, 0.5372549 , 0.7176471 ],
         ...,
```

```
         [0.3372549 , 0.5019608 , 0.6901961 ],
         [0.32941177, 0.49411765, 0.68235296],
         [0.32941177, 0.49411765, 0.68235296]],

        [[0.38431373, 0.5411765 , 0.7137255 ],
         [0.4       , 0.5411765 , 0.7176471 ],
         [0.40392157, 0.5372549 , 0.7176471 ],
         ...,
         [0.3372549 , 0.5019608 , 0.6901961 ],
         [0.32941177, 0.49411765, 0.68235296],
         [0.32941177, 0.49411765, 0.68235296]]],


       ...,


       [[[0.14117648, 0.16078432, 0.07058824],
         [0.1254902 , 0.14509805, 0.07058824],
         [0.16470589, 0.18039216, 0.11764706],
         ...,
         [0.7490196 , 0.72156864, 0.7372549 ],
         [0.6901961 , 0.6627451 , 0.6784314 ],
         [0.8235294 , 0.79607844, 0.8117647 ]],

        [[0.13725491, 0.15686275, 0.06666667],
         [0.1254902 , 0.14509805, 0.07058824],
         [0.15294118, 0.16862746, 0.10588235],
         ...,
         [0.8392157 , 0.8156863 , 0.83137256],
         [0.78039217, 0.7607843 , 0.77254903],
         [0.90588236, 0.8862745 , 0.8980392 ]],

        [[0.10588235, 0.12941177, 0.03529412],
         [0.10196079, 0.11764706, 0.04313726],
         [0.09803922, 0.11372549, 0.05098039],
         ...,
         [0.78431374, 0.7764706 , 0.76862746],
         [0.74509805, 0.73333335, 0.7294118 ],
         [0.7607843 , 0.7490196 , 0.74509805]],

        ...,

        [[0.28627452, 0.27450982, 0.15686275],
         [0.30980393, 0.2901961 , 0.16862746],
         [0.36862746, 0.3254902 , 0.21176471],
         ...,
         [0.7647059 , 0.3372549 , 0.3137255 ],
         [0.78039217, 0.3764706 , 0.35686275],
         [0.75686276, 0.4       , 0.3764706 ]],
```

```
        [[0.21176471, 0.20784314, 0.05098039],
         [0.23137255, 0.23137255, 0.06666667],
         [0.27058825, 0.2627451 , 0.10588235],
         ...,
         [0.45490196, 0.40392157, 0.25882354],
         [0.45882353, 0.4117647 , 0.27058825],
         [0.44313726, 0.40392157, 0.27450982]],

        [[0.24705882, 0.24705882, 0.05490196],
         [0.23529412, 0.23921569, 0.04705882],
         [0.22352941, 0.23137255, 0.04313726],
         ...,
         [0.23529412, 0.23137255, 0.05098039],
         [0.23529412, 0.22745098, 0.04705882],
         [0.23921569, 0.22352941, 0.05490196]]],


       [[[0.08627451, 0.03137255, 0.02745098],
         [0.09019608, 0.03529412, 0.03137255],
         [0.09411765, 0.03529412, 0.03137255],
         ...,
         [0.5372549 , 0.5882353 , 0.43137255],
         [0.5686275 , 0.654902  , 0.3647059 ],
         [0.57254905, 0.6784314 , 0.3529412 ]],

        [[0.08627451, 0.03137255, 0.02745098],
         [0.09019608, 0.03529412, 0.03137255],
         [0.09019608, 0.03529412, 0.03137255],
         ...,
         [0.4509804 , 0.49019608, 0.37254903],
         [0.5764706 , 0.6392157 , 0.42352942],
         [0.5803922 , 0.67058825, 0.3647059 ]],

        [[0.08627451, 0.03137255, 0.02745098],
         [0.09019608, 0.03529412, 0.03137255],
         [0.09019608, 0.03529412, 0.03137255],
         ...,
         [0.30980393, 0.3372549 , 0.21960784],
         [0.5294118 , 0.57254905, 0.4117647 ],
         [0.57254905, 0.6509804 , 0.38039216]],

        ...,

        [[0.32156864, 0.38431373, 0.1882353 ],
         [0.30980393, 0.43137255, 0.09019608],
         [0.3764706 , 0.5019608 , 0.14117648],
         ...,
         [0.08235294, 0.04705882, 0.01568628],
         [0.18039216, 0.19607843, 0.0627451 ],
         [0.34509805, 0.41960785, 0.18039216]],
```

```
        [[0.32156864, 0.40784314, 0.1882353 ],
         [0.33333334, 0.45882353, 0.11764706],
         [0.3764706 , 0.49803922, 0.14117648],
         ...,
         [0.09803922, 0.03921569, 0.03137255],
         [0.08235294, 0.04313726, 0.01568628],
         [0.10980392, 0.09019608, 0.02745098]],

        [[0.31764707, 0.42352942, 0.18039216],
         [0.31764707, 0.4509804 , 0.09803922],
         [0.3764706 , 0.49411765, 0.13725491],
         ...,
         [0.09019608, 0.04313726, 0.02745098],
         [0.09411765, 0.03921569, 0.04705882],
         [0.09803922, 0.03529412, 0.03137255]]],


       [[[0.53333336, 0.68235296, 0.74509805],
         [0.5058824 , 0.6745098 , 0.7254902 ],
         [0.49411765, 0.6784314 , 0.7254902 ],
         ...,
         [0.5176471 , 0.6745098 , 0.7137255 ],
         [0.5176471 , 0.67058825, 0.70980394],
         [0.5176471 , 0.6784314 , 0.7176471 ]],

        [[0.5529412 , 0.7137255 , 0.77254903],
         [0.53333336, 0.7019608 , 0.75686276],
         [0.5254902 , 0.7058824 , 0.75686276],
         ...,
         [0.5647059 , 0.7254902 , 0.7647059 ],
         [0.56078434, 0.7176471 , 0.7607843 ],
         [0.56078434, 0.7254902 , 0.7647059 ]],

        [[0.54901963, 0.72156864, 0.77254903],
         [0.53333336, 0.7058824 , 0.75686276],
         [0.5411765 , 0.70980394, 0.7607843 ],
         ...,
         [0.59607846, 0.7529412 , 0.7921569 ],
         [0.5882353 , 0.74509805, 0.78431374],
         [0.5921569 , 0.7529412 , 0.7921569 ]],

        ...,

        [[0.30980393, 0.46666667, 0.20784314],
         [0.3529412 , 0.4509804 , 0.2509804 ],
         [0.37254903, 0.45490196, 0.24313726],
         ...,
         [0.2509804 , 0.5137255 , 0.16862746],
         [0.29411766, 0.5411765 , 0.21176471],
```

```
         [0.31764707, 0.5803922 , 0.25882354]],

        [[0.27058825, 0.4117647 , 0.13333334],
         [0.3137255 , 0.40392157, 0.16078432],
         [0.31764707, 0.39607844, 0.16470589],
         ...,
         [0.29803923, 0.5137255 , 0.17254902],
         [0.3019608 , 0.54509807, 0.20784314],
         [0.34117648, 0.5647059 , 0.25490198]],

        [[0.25882354, 0.38039216, 0.09803922],
         [0.3019608 , 0.3882353 , 0.11764706],
         [0.3137255 , 0.39607844, 0.14901961],
         ...,
         [0.2901961 , 0.45490196, 0.13333334],
         [0.28627452, 0.4862745 , 0.16470589],
         [0.34901962, 0.50980395, 0.21960784]]], dtype=float32)
```

2.4 - Imagens do dataset de treino

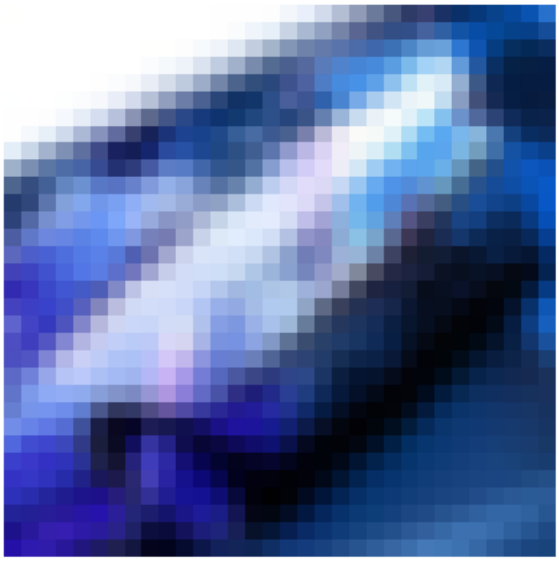
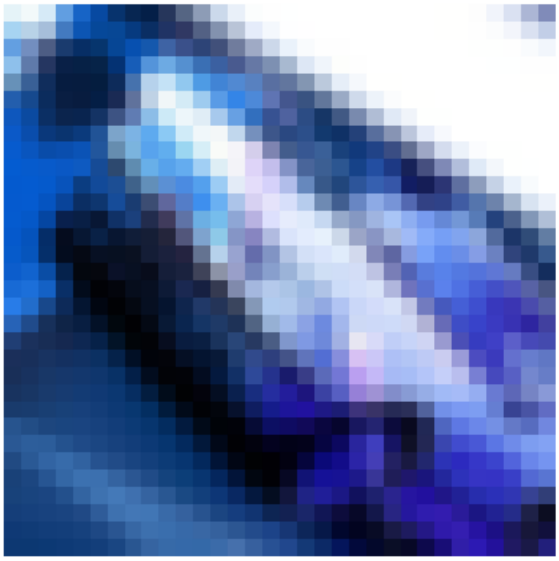
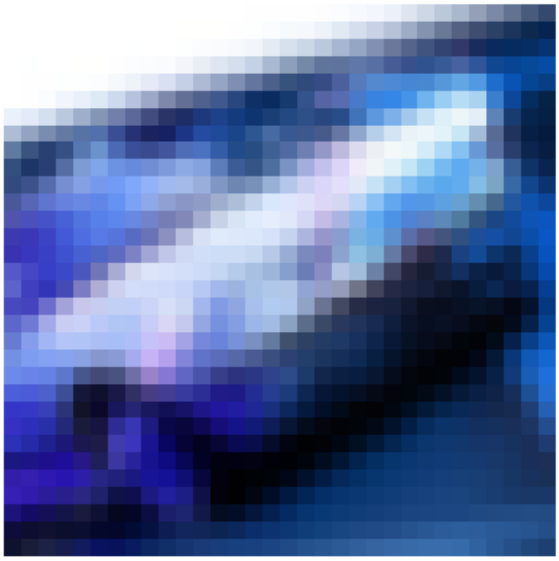
Visualização de dez imagens aleatórias do dataset de treino

```
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

A arquitetura deste modelo foi inspirada na arquitetura do modelo VGG16. Com isto, temos:

- Como supramencionado, a normalização dos valores de cada pixel da imagem
- A aplicação das operações de Data Augmentation
- Três blocos de layers convolucionais:
  - Cada um com duas camadas convolucionais:

  - A quantidade de filtros em cada camada convolucional vai aumentado progressivamente de 32 filtros até 128 e mantem-se constante dentro de cada bloco convolucional, isto é, dentro do mesmo bloco os filtros é utilizada a mesma quantidade de filtros para as ambas as camadas
  - É utilizada a função de ativação ReLu
  - No final de cada bloco convolucional 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)
  - Ainda sobre o final de cada bloco convolucional, é utilizado o Dropout() com o valor de 0.2, isto é, ou seja, no final de cada bloco convolucional são excluidos 20% (x * 100 %, sendo x o valor do paramêtro utilizado no Dropout) dos neurónios presentes naquele momento
  - 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
  - É utilizada uma outra camada densa, com 10 filtros (que equivalem ao número de classes presentes no 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 Block (2 layers)
x = layers.Conv2D(filters=32, kernel_size=3, padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.Activation("relu")(x)

x = layers.Conv2D(filters=32, kernel_size=3, padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.Activation("relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Dropout(0.2)(x)

# 2nd Convolutional Block (2 layers)
```

```
x = layers.Conv2D(filters=64, kernel_size=3, padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.Activation("relu")(x)

x = layers.Conv2D(filters=64, kernel_size=3, padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.Activation("relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Dropout(0.2)(x)

# 3rd Convolutional Block (2 layers)
x = layers.Conv2D(filters=128, kernel_size=3, padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.Activation("relu")(x)

x = layers.Conv2D(filters=128, kernel_size=3, padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.Activation("relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Dropout(0.2)(x)

# Classification Block
x = layers.Flatten()(x)
x = layers.Dense(128)(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=inputs, outputs=outputs)

model.summary()
```

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

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

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

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

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

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

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

batch_normalization_1 (Batc  (None, 32, 32, 32)        128
hNormalization)

activation_1 (Activation)    (None, 32, 32, 32)        0

max_pooling2d (MaxPooling2D  (None, 16, 16, 32)        0
)

dropout (Dropout)            (None, 16, 16, 32)        0

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

batch_normalization_2 (Batc  (None, 16, 16, 64)        256
hNormalization)

activation_2 (Activation)    (None, 16, 16, 64)        0

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

batch_normalization_3 (Batc  (None, 16, 16, 64)        256
hNormalization)

activation_3 (Activation)    (None, 16, 16, 64)        0

max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)          0
2D)

dropout_1 (Dropout)          (None, 8, 8, 64)          0

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

batch_normalization_4 (Batc  (None, 8, 8, 128)         512
hNormalization)

activation_4 (Activation)    (None, 8, 8, 128)         0

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

batch_normalization_5 (Batc  (None, 8, 8, 128)         512
hNormalization)

activation_5 (Activation)    (None, 8, 8, 128)         0

max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)         0
2D)
```

```
 dropout_2 (Dropout)          (None, 4, 4, 128)         0

 flatten (Flatten)            (None, 2048)              0

 dense (Dense)                (None, 128)               262272

 batch_normalization_6 (Batc  (None, 128)               512
 hNormalization)

 activation_6 (Activation)    (None, 128)               0

 dropout_3 (Dropout)          (None, 128)               0

 dense_1 (Dense)              (None, 10)                1290

=================================================================
Total params: 552,874
Trainable params: 551,722
Non-trainable params: 1,152
_________________________________________________________________
```

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.

Neste modelo foi utilizado como otimizador o Adam, com o principal objetivo de explorar mais otimizadores. Não é definido um learning rate a ser utilizado por este otimizador, sendo utilizado o rate por omissão, já que este já possui, de base, técnicas de otimização do learning rate.

```
model.compile(
    loss='categorical_crossentropy',
    optimizer=optimizers.Adam(),
    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_B_DA.keras',
        save_best_only = True,
        monitor='val_loss',
        verbose=1
    )
]

# Treinar o modelo
history = model.fit(train_dataset, epochs=100,
validation_data=val_dataset, callbacks=callbacks)

Epoch 1/100
1252/1252 [==============================] - ETA: 0s - loss: 1.7639 -
accuracy: 0.4021 - precision: 0.5872 - recall: 0.1771
Epoch 1: val_loss improved from inf to 1.63954, saving model to
models\IC_S.keras
1252/1252 [==============================] - 281s 220ms/step - loss:
1.7639 - accuracy: 0.4021 - precision: 0.5872 - recall: 0.1771 -
val_loss: 1.6395 - val_accuracy: 0.4432 - val_precision: 0.6019 -
val_recall: 0.3053
Epoch 2/100
1252/1252 [==============================] - ETA: 0s - loss: 1.3975 -
accuracy: 0.5238 - precision: 0.7100 - recall: 0.3050
Epoch 2: val_loss did not improve from 1.63954
```

```
1252/1252 [==============================] - 286s 228ms/step - loss:
1.3975 - accuracy: 0.5238 - precision: 0.7100 - recall: 0.3050 -
val_loss: 1.7155 - val_accuracy: 0.4534 - val_precision: 0.5343 -
val_recall: 0.3843
Epoch 3/100
1252/1252 [==============================] - ETA: 0s - loss: 1.2565 -
accuracy: 0.5768 - precision: 0.7433 - recall: 0.3943
Epoch 3: val_loss improved from 1.63954 to 1.18372, saving model to
models\IC_S.keras
1252/1252 [==============================] - 291s 232ms/step - loss:
1.2565 - accuracy: 0.5768 - precision: 0.7433 - recall: 0.3943 -
val_loss: 1.1837 - val_accuracy: 0.5962 - val_precision: 0.7178 -
val_recall: 0.4887
Epoch 4/100
1252/1252 [==============================] - ETA: 0s - loss: 1.1688 -
accuracy: 0.6098 - precision: 0.7652 - recall: 0.4504
Epoch 4: val_loss improved from 1.18372 to 1.16184, saving model to
models\IC_S.keras
1252/1252 [==============================] - 292s 233ms/step - loss:
1.1688 - accuracy: 0.6098 - precision: 0.7652 - recall: 0.4504 -
val_loss: 1.1618 - val_accuracy: 0.6141 - val_precision: 0.7318 -
val_recall: 0.5156
Epoch 5/100
1252/1252 [==============================] - ETA: 0s - loss: 1.1025 -
accuracy: 0.6367 - precision: 0.7737 - recall: 0.4861
Epoch 5: val_loss improved from 1.16184 to 0.93187, saving model to
models\IC_S.keras
1252/1252 [==============================] - 293s 234ms/step - loss:
1.1025 - accuracy: 0.6367 - precision: 0.7737 - recall: 0.4861 -
val_loss: 0.9319 - val_accuracy: 0.6905 - val_precision: 0.8084 -
val_recall: 0.5821
Epoch 6/100
1252/1252 [==============================] - ETA: 0s - loss: 1.0483 -
accuracy: 0.6548 - precision: 0.7886 - recall: 0.5203
Epoch 6: val_loss did not improve from 0.93187
1252/1252 [==============================] - 293s 234ms/step - loss:
1.0483 - accuracy: 0.6548 - precision: 0.7886 - recall: 0.5203 -
val_loss: 1.0226 - val_accuracy: 0.6652 - val_precision: 0.8007 -
val_recall: 0.5668
Epoch 7/100
1252/1252 [==============================] - ETA: 0s - loss: 0.9952 -
accuracy: 0.6735 - precision: 0.7976 - recall: 0.5504
Epoch 7: val_loss improved from 0.93187 to 0.86348, saving model to
models\IC_S.keras
1252/1252 [==============================] - 292s 234ms/step - loss:
0.9952 - accuracy: 0.6735 - precision: 0.7976 - recall: 0.5504 -
val_loss: 0.8635 - val_accuracy: 0.7137 - val_precision: 0.8072 -
val_recall: 0.6280
Epoch 8/100
```

```
1252/1252 [==============================] - ETA: 0s - loss: 0.9571 -
accuracy: 0.6859 - precision: 0.8019 - recall: 0.5712
Epoch 8: val_loss improved from 0.86348 to 0.86332, saving model to
models\IC_S.keras
1252/1252 [==============================] - 293s 234ms/step - loss:
0.9571 - accuracy: 0.6859 - precision: 0.8019 - recall: 0.5712 -
val_loss: 0.8633 - val_accuracy: 0.7108 - val_precision: 0.7946 -
val_recall: 0.6391
Epoch 9/100
1252/1252 [==============================] - ETA: 0s - loss: 0.9296 -
accuracy: 0.6985 - precision: 0.8073 - recall: 0.5875
Epoch 9: val_loss did not improve from 0.86332
1252/1252 [==============================] - 293s 234ms/step - loss:
0.9296 - accuracy: 0.6985 - precision: 0.8073 - recall: 0.5875 -
val_loss: 0.8901 - val_accuracy: 0.7032 - val_precision: 0.7947 -
val_recall: 0.6278
Epoch 10/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8967 -
accuracy: 0.7093 - precision: 0.8159 - recall: 0.6062
Epoch 10: val_loss improved from 0.86332 to 0.79270, saving model to
models\IC_S.keras
1252/1252 [==============================] - 293s 234ms/step - loss:
0.8967 - accuracy: 0.7093 - precision: 0.8159 - recall: 0.6062 -
val_loss: 0.7927 - val_accuracy: 0.7416 - val_precision: 0.8214 -
val_recall: 0.6660
Epoch 11/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8743 -
accuracy: 0.7155 - precision: 0.8194 - recall: 0.6161
Epoch 11: val_loss improved from 0.79270 to 0.76495, saving model to
models\IC_S.keras
1252/1252 [==============================] - 292s 234ms/step - loss:
0.8743 - accuracy: 0.7155 - precision: 0.8194 - recall: 0.6161 -
val_loss: 0.7650 - val_accuracy: 0.7458 - val_precision: 0.8272 -
val_recall: 0.6786
Epoch 12/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8522 -
accuracy: 0.7246 - precision: 0.8231 - recall: 0.6256
Epoch 12: val_loss improved from 0.76495 to 0.72196, saving model to
models\IC_S.keras
1252/1252 [==============================] - 292s 234ms/step - loss:
0.8522 - accuracy: 0.7246 - precision: 0.8231 - recall: 0.6256 -
val_loss: 0.7220 - val_accuracy: 0.7673 - val_precision: 0.8323 -
val_recall: 0.7096
Epoch 13/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8293 -
accuracy: 0.7333 - precision: 0.8269 - recall: 0.6393
Epoch 13: val_loss did not improve from 0.72196
1252/1252 [==============================] - 292s 233ms/step - loss:
0.8293 - accuracy: 0.7333 - precision: 0.8269 - recall: 0.6393 -
```

```
val_loss: 0.8363 - val_accuracy: 0.7286 - val_precision: 0.8035 -
val_recall: 0.6659
Epoch 14/100
1252/1252 [==============================] - ETA: 0s - loss: 0.8133 -
accuracy: 0.7400 - precision: 0.8333 - recall: 0.6488
Epoch 14: val_loss improved from 0.72196 to 0.71655, saving model to
models\IC_S.keras
1252/1252 [==============================] - 293s 234ms/step - loss:
0.8133 - accuracy: 0.7400 - precision: 0.8333 - recall: 0.6488 -
val_loss: 0.7165 - val_accuracy: 0.7660 - val_precision: 0.8335 -
val_recall: 0.7014
Epoch 15/100
1252/1252 [==============================] - ETA: 0s - loss: 0.7918 -
accuracy: 0.7448 - precision: 0.8345 - recall: 0.6578
Epoch 15: val_loss did not improve from 0.71655
1252/1252 [==============================] - 293s 234ms/step - loss:
0.7918 - accuracy: 0.7448 - precision: 0.8345 - recall: 0.6578 -
val_loss: 0.7891 - val_accuracy: 0.7424 - val_precision: 0.8080 -
val_recall: 0.6913
Epoch 16/100
1252/1252 [==============================] - ETA: 0s - loss: 0.7829 -
accuracy: 0.7514 - precision: 0.8374 - recall: 0.6638
Epoch 16: val_loss improved from 0.71655 to 0.62375, saving model to
models\IC_S.keras
1252/1252 [==============================] - 293s 234ms/step - loss:
0.7829 - accuracy: 0.7514 - precision: 0.8374 - recall: 0.6638 -
val_loss: 0.6237 - val_accuracy: 0.7995 - val_precision: 0.8610 -
val_recall: 0.7424
Epoch 17/100
1252/1252 [==============================] - ETA: 0s - loss: 0.7646 -
accuracy: 0.7548 - precision: 0.8396 - recall: 0.6702
Epoch 17: val_loss did not improve from 0.62375
1252/1252 [==============================] - 293s 234ms/step - loss:
0.7646 - accuracy: 0.7548 - precision: 0.8396 - recall: 0.6702 -
val_loss: 0.6292 - val_accuracy: 0.7961 - val_precision: 0.8589 -
val_recall: 0.7376
Epoch 18/100
1252/1252 [==============================] - ETA: 0s - loss: 0.7557 -
accuracy: 0.7580 - precision: 0.8428 - recall: 0.6788
Epoch 18: val_loss improved from 0.62375 to 0.60231, saving model to
models\IC_S.keras
1252/1252 [==============================] - 292s 234ms/step - loss:
0.7557 - accuracy: 0.7580 - precision: 0.8428 - recall: 0.6788 -
val_loss: 0.6023 - val_accuracy: 0.8072 - val_precision: 0.8609 -
val_recall: 0.7535
Epoch 19/100
1252/1252 [==============================] - ETA: 0s - loss: 0.7417 -
accuracy: 0.7614 - precision: 0.8421 - recall: 0.6839
Epoch 19: val_loss did not improve from 0.60231
```

```
1252/1252 [==============================] - 293s 234ms/step - loss:
0.7417 - accuracy: 0.7614 - precision: 0.8421 - recall: 0.6839 -
val_loss: 0.6238 - val_accuracy: 0.7986 - val_precision: 0.8571 -
val_recall: 0.7446
Epoch 20/100
1252/1252 [==============================] - ETA: 0s - loss: 0.7286 -
accuracy: 0.7646 - precision: 0.8447 - recall: 0.6888
Epoch 20: val_loss did not improve from 0.60231
1252/1252 [==============================] - 293s 234ms/step - loss:
0.7286 - accuracy: 0.7646 - precision: 0.8447 - recall: 0.6888 -
val_loss: 0.6187 - val_accuracy: 0.8012 - val_precision: 0.8572 -
val_recall: 0.7499
Epoch 21/100
1252/1252 [==============================] - ETA: 0s - loss: 0.7224 -
accuracy: 0.7702 - precision: 0.8480 - recall: 0.6928
Epoch 21: val_loss did not improve from 0.60231
1252/1252 [==============================] - 293s 234ms/step - loss:
0.7224 - accuracy: 0.7702 - precision: 0.8480 - recall: 0.6928 -
val_loss: 0.7055 - val_accuracy: 0.7757 - val_precision: 0.8349 -
val_recall: 0.7222
Epoch 22/100
1252/1252 [==============================] - ETA: 0s - loss: 0.7144 -
accuracy: 0.7734 - precision: 0.8514 - recall: 0.6975
Epoch 22: val_loss did not improve from 0.60231
1252/1252 [==============================] - 293s 234ms/step - loss:
0.7144 - accuracy: 0.7734 - precision: 0.8514 - recall: 0.6975 -
val_loss: 0.6570 - val_accuracy: 0.7877 - val_precision: 0.8437 -
val_recall: 0.7411
Epoch 23/100
1252/1252 [==============================] - ETA: 0s - loss: 0.7030 -
accuracy: 0.7755 - precision: 0.8511 - recall: 0.7036
Epoch 23: val_loss did not improve from 0.60231
1252/1252 [==============================] - 293s 234ms/step - loss:
0.7030 - accuracy: 0.7755 - precision: 0.8511 - recall: 0.7036 -
val_loss: 0.6923 - val_accuracy: 0.7802 - val_precision: 0.8364 -
val_recall: 0.7333
Epoch 24/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6940 -
accuracy: 0.7784 - precision: 0.8535 - recall: 0.7077
Epoch 24: val_loss did not improve from 0.60231
1252/1252 [==============================] - 292s 233ms/step - loss:
0.6940 - accuracy: 0.7784 - precision: 0.8535 - recall: 0.7077 -
val_loss: 0.6332 - val_accuracy: 0.7982 - val_precision: 0.8508 -
val_recall: 0.7522
Epoch 25/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6908 -
accuracy: 0.7768 - precision: 0.8498 - recall: 0.7061
Epoch 25: val_loss improved from 0.60231 to 0.58402, saving model to
models\IC_S.keras
```

```
1252/1252 [==============================] - 293s 234ms/step - loss:
0.6908 - accuracy: 0.7768 - precision: 0.8498 - recall: 0.7061 -
val_loss: 0.5840 - val_accuracy: 0.8144 - val_precision: 0.8674 -
val_recall: 0.7646
Epoch 26/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6812 -
accuracy: 0.7816 - precision: 0.8559 - recall: 0.7125
Epoch 26: val_loss did not improve from 0.58402
1252/1252 [==============================] - 293s 234ms/step - loss:
0.6812 - accuracy: 0.7816 - precision: 0.8559 - recall: 0.7125 -
val_loss: 0.6140 - val_accuracy: 0.7998 - val_precision: 0.8548 -
val_recall: 0.7582
Epoch 27/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6730 -
accuracy: 0.7854 - precision: 0.8565 - recall: 0.7177
Epoch 27: val_loss did not improve from 0.58402
1252/1252 [==============================] - 292s 233ms/step - loss:
0.6730 - accuracy: 0.7854 - precision: 0.8565 - recall: 0.7177 -
val_loss: 0.6421 - val_accuracy: 0.7974 - val_precision: 0.8500 -
val_recall: 0.7534
Epoch 28/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6719 -
accuracy: 0.7850 - precision: 0.8553 - recall: 0.7180
Epoch 28: val_loss improved from 0.58402 to 0.55699, saving model to
models\IC_S.keras
1252/1252 [==============================] - 294s 235ms/step - loss:
0.6719 - accuracy: 0.7850 - precision: 0.8553 - recall: 0.7180 -
val_loss: 0.5570 - val_accuracy: 0.8202 - val_precision: 0.8694 -
val_recall: 0.7797
Epoch 29/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6662 -
accuracy: 0.7854 - precision: 0.8562 - recall: 0.7192
Epoch 29: val_loss did not improve from 0.55699
1252/1252 [==============================] - 294s 235ms/step - loss:
0.6662 - accuracy: 0.7854 - precision: 0.8562 - recall: 0.7192 -
val_loss: 0.7221 - val_accuracy: 0.7777 - val_precision: 0.8212 -
val_recall: 0.7351
Epoch 30/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6535 -
accuracy: 0.7925 - precision: 0.8597 - recall: 0.7262
Epoch 30: val_loss did not improve from 0.55699
1252/1252 [==============================] - 293s 234ms/step - loss:
0.6535 - accuracy: 0.7925 - precision: 0.8597 - recall: 0.7262 -
val_loss: 0.5620 - val_accuracy: 0.8215 - val_precision: 0.8699 -
val_recall: 0.7827
Epoch 31/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6527 -
accuracy: 0.7926 - precision: 0.8592 - recall: 0.7268
Epoch 31: val_loss did not improve from 0.55699
```

```
1252/1252 [==============================] - 293s 234ms/step - loss:
0.6527 - accuracy: 0.7926 - precision: 0.8592 - recall: 0.7268 -
val_loss: 0.6987 - val_accuracy: 0.7787 - val_precision: 0.8282 -
val_recall: 0.7312
Epoch 32/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6466 -
accuracy: 0.7904 - precision: 0.8590 - recall: 0.7283
Epoch 32: val_loss did not improve from 0.55699
1252/1252 [==============================] - 293s 234ms/step - loss:
0.6466 - accuracy: 0.7904 - precision: 0.8590 - recall: 0.7283 -
val_loss: 0.6264 - val_accuracy: 0.8033 - val_precision: 0.8541 -
val_recall: 0.7622
Epoch 33/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6434 -
accuracy: 0.7913 - precision: 0.8576 - recall: 0.7295
Epoch 33: val_loss improved from 0.55699 to 0.55208, saving model to
models\IC_S.keras
1252/1252 [==============================] - 293s 234ms/step - loss:
0.6434 - accuracy: 0.7913 - precision: 0.8576 - recall: 0.7295 -
val_loss: 0.5521 - val_accuracy: 0.8226 - val_precision: 0.8735 -
val_recall: 0.7830
Epoch 34/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6342 -
accuracy: 0.7963 - precision: 0.8624 - recall: 0.7361
Epoch 34: val_loss did not improve from 0.55208
1252/1252 [==============================] - 293s 234ms/step - loss:
0.6342 - accuracy: 0.7963 - precision: 0.8624 - recall: 0.7361 -
val_loss: 0.5787 - val_accuracy: 0.8180 - val_precision: 0.8704 -
val_recall: 0.7745
Epoch 35/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6281 -
accuracy: 0.7991 - precision: 0.8657 - recall: 0.7384
Epoch 35: val_loss did not improve from 0.55208
1252/1252 [==============================] - 293s 234ms/step - loss:
0.6281 - accuracy: 0.7991 - precision: 0.8657 - recall: 0.7384 -
val_loss: 0.5932 - val_accuracy: 0.8109 - val_precision: 0.8576 -
val_recall: 0.7740
Epoch 36/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6291 -
accuracy: 0.7977 - precision: 0.8633 - recall: 0.7357
Epoch 36: val_loss did not improve from 0.55208
1252/1252 [==============================] - 294s 235ms/step - loss:
0.6291 - accuracy: 0.7977 - precision: 0.8633 - recall: 0.7357 -
val_loss: 0.5663 - val_accuracy: 0.8233 - val_precision: 0.8701 -
val_recall: 0.7825
Epoch 37/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6221 -
accuracy: 0.7999 - precision: 0.8641 - recall: 0.7409
Epoch 37: val_loss did not improve from 0.55208
```

```
1252/1252 [==============================] - 298s 238ms/step - loss:
0.6221 - accuracy: 0.7999 - precision: 0.8641 - recall: 0.7409 -
val_loss: 0.7173 - val_accuracy: 0.7759 - val_precision: 0.8323 -
val_recall: 0.7313
Epoch 38/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6162 -
accuracy: 0.8032 - precision: 0.8670 - recall: 0.7434
Epoch 38: val_loss did not improve from 0.55208
1252/1252 [==============================] - 269s 215ms/step - loss:
0.6162 - accuracy: 0.8032 - precision: 0.8670 - recall: 0.7434 -
val_loss: 0.5971 - val_accuracy: 0.8136 - val_precision: 0.8591 -
val_recall: 0.7748
Epoch 39/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6099 -
accuracy: 0.8040 - precision: 0.8672 - recall: 0.7463
Epoch 39: val_loss did not improve from 0.55208
1252/1252 [==============================] - 274s 219ms/step - loss:
0.6099 - accuracy: 0.8040 - precision: 0.8672 - recall: 0.7463 -
val_loss: 0.5952 - val_accuracy: 0.8088 - val_precision: 0.8528 -
val_recall: 0.7727
Epoch 40/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5971 -
accuracy: 0.8083 - precision: 0.8685 - recall: 0.7529
Epoch 40: val_loss did not improve from 0.55208
1252/1252 [==============================] - 303s 242ms/step - loss:
0.5971 - accuracy: 0.8083 - precision: 0.8685 - recall: 0.7529 -
val_loss: 0.5693 - val_accuracy: 0.8261 - val_precision: 0.8622 -
val_recall: 0.7915
Epoch 41/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5982 -
accuracy: 0.8085 - precision: 0.8707 - recall: 0.7537
Epoch 41: val_loss did not improve from 0.55208
1252/1252 [==============================] - 308s 246ms/step - loss:
0.5982 - accuracy: 0.8085 - precision: 0.8707 - recall: 0.7537 -
val_loss: 0.6286 - val_accuracy: 0.8065 - val_precision: 0.8458 -
val_recall: 0.7725
Epoch 42/100
1252/1252 [==============================] - ETA: 0s - loss: 0.6001 -
accuracy: 0.8073 - precision: 0.8688 - recall: 0.7507
Epoch 42: val_loss improved from 0.55208 to 0.54815, saving model to
models\IC_S.keras
1252/1252 [==============================] - 311s 248ms/step - loss:
0.6001 - accuracy: 0.8073 - precision: 0.8688 - recall: 0.7507 -
val_loss: 0.5482 - val_accuracy: 0.8303 - val_precision: 0.8725 -
val_recall: 0.7921
Epoch 43/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5931 -
accuracy: 0.8093 - precision: 0.8703 - recall: 0.7540
Epoch 43: val_loss improved from 0.54815 to 0.53795, saving model to
```

```
models\IC_S.keras
1252/1252 [==============================] - 311s 248ms/step - loss:
0.5931 - accuracy: 0.8093 - precision: 0.8703 - recall: 0.7540 -
val_loss: 0.5379 - val_accuracy: 0.8328 - val_precision: 0.8726 -
val_recall: 0.7999
Epoch 44/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5918 -
accuracy: 0.8077 - precision: 0.8678 - recall: 0.7536
Epoch 44: val_loss improved from 0.53795 to 0.53698, saving model to
models\IC_S.keras
1252/1252 [==============================] - 312s 249ms/step - loss:
0.5918 - accuracy: 0.8077 - precision: 0.8678 - recall: 0.7536 -
val_loss: 0.5370 - val_accuracy: 0.8303 - val_precision: 0.8735 -
val_recall: 0.7969
Epoch 45/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5889 -
accuracy: 0.8118 - precision: 0.8725 - recall: 0.7569
Epoch 45: val_loss did not improve from 0.53698
1252/1252 [==============================] - 308s 246ms/step - loss:
0.5889 - accuracy: 0.8118 - precision: 0.8725 - recall: 0.7569 -
val_loss: 0.5715 - val_accuracy: 0.8206 - val_precision: 0.8620 -
val_recall: 0.7870
Epoch 46/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5883 -
accuracy: 0.8089 - precision: 0.8673 - recall: 0.7547
Epoch 46: val_loss did not improve from 0.53698
1252/1252 [==============================] - 306s 244ms/step - loss:
0.5883 - accuracy: 0.8089 - precision: 0.8673 - recall: 0.7547 -
val_loss: 0.5729 - val_accuracy: 0.8234 - val_precision: 0.8681 -
val_recall: 0.7893
Epoch 47/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5759 -
accuracy: 0.8157 - precision: 0.8714 - recall: 0.7607
Epoch 47: val_loss did not improve from 0.53698
1252/1252 [==============================] - 311s 248ms/step - loss:
0.5759 - accuracy: 0.8157 - precision: 0.8714 - recall: 0.7607 -
val_loss: 0.5593 - val_accuracy: 0.8240 - val_precision: 0.8677 -
val_recall: 0.7905
Epoch 48/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5843 -
accuracy: 0.8130 - precision: 0.8712 - recall: 0.7589
Epoch 48: val_loss improved from 0.53698 to 0.53385, saving model to
models\IC_S.keras
1252/1252 [==============================] - 308s 246ms/step - loss:
0.5843 - accuracy: 0.8130 - precision: 0.8712 - recall: 0.7589 -
val_loss: 0.5339 - val_accuracy: 0.8329 - val_precision: 0.8728 -
val_recall: 0.8016
Epoch 49/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5749 -
```

```
accuracy: 0.8171 - precision: 0.8730 - recall: 0.7645
Epoch 49: val_loss improved from 0.53385 to 0.53191, saving model to
models\IC_S.keras
1252/1252 [==============================] - 313s 250ms/step - loss:
0.5749 - accuracy: 0.8171 - precision: 0.8730 - recall: 0.7645 -
val_loss: 0.5319 - val_accuracy: 0.8317 - val_precision: 0.8730 -
val_recall: 0.8030
Epoch 50/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5783 -
accuracy: 0.8158 - precision: 0.8714 - recall: 0.7610
Epoch 50: val_loss did not improve from 0.53191
1252/1252 [==============================] - 308s 246ms/step - loss:
0.5783 - accuracy: 0.8158 - precision: 0.8714 - recall: 0.7610 -
val_loss: 0.5462 - val_accuracy: 0.8285 - val_precision: 0.8692 -
val_recall: 0.7977
Epoch 51/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5644 -
accuracy: 0.8202 - precision: 0.8748 - recall: 0.7677
Epoch 51: val_loss improved from 0.53191 to 0.51866, saving model to
models\IC_S.keras
1252/1252 [==============================] - 287s 229ms/step - loss:
0.5644 - accuracy: 0.8202 - precision: 0.8748 - recall: 0.7677 -
val_loss: 0.5187 - val_accuracy: 0.8348 - val_precision: 0.8765 -
val_recall: 0.8042
Epoch 52/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5657 -
accuracy: 0.8192 - precision: 0.8751 - recall: 0.7672
Epoch 52: val_loss did not improve from 0.51866
1252/1252 [==============================] - 285s 227ms/step - loss:
0.5657 - accuracy: 0.8192 - precision: 0.8751 - recall: 0.7672 -
val_loss: 0.5625 - val_accuracy: 0.8281 - val_precision: 0.8659 -
val_recall: 0.7944
Epoch 53/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5650 -
accuracy: 0.8185 - precision: 0.8712 - recall: 0.7676
Epoch 53: val_loss did not improve from 0.51866
1252/1252 [==============================] - 286s 229ms/step - loss:
0.5650 - accuracy: 0.8185 - precision: 0.8712 - recall: 0.7676 -
val_loss: 0.5790 - val_accuracy: 0.8231 - val_precision: 0.8615 -
val_recall: 0.7932
Epoch 54/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5587 -
accuracy: 0.8210 - precision: 0.8772 - recall: 0.7699
Epoch 54: val_loss improved from 0.51866 to 0.50945, saving model to
models\IC_S.keras
1252/1252 [==============================] - 295s 235ms/step - loss:
0.5587 - accuracy: 0.8210 - precision: 0.8772 - recall: 0.7699 -
val_loss: 0.5094 - val_accuracy: 0.8436 - val_precision: 0.8804 -
val_recall: 0.8109
```

```
Epoch 55/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5594 -
accuracy: 0.8200 - precision: 0.8766 - recall: 0.7690
Epoch 55: val_loss did not improve from 0.50945
1252/1252 [==============================] - 288s 230ms/step - loss:
0.5594 - accuracy: 0.8200 - precision: 0.8766 - recall: 0.7690 -
val_loss: 0.5240 - val_accuracy: 0.8379 - val_precision: 0.8775 -
val_recall: 0.8045
Epoch 56/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5542 -
accuracy: 0.8230 - precision: 0.8755 - recall: 0.7727
Epoch 56: val_loss did not improve from 0.50945
1252/1252 [==============================] - 292s 233ms/step - loss:
0.5542 - accuracy: 0.8230 - precision: 0.8755 - recall: 0.7727 -
val_loss: 0.5277 - val_accuracy: 0.8370 - val_precision: 0.8695 -
val_recall: 0.8092
Epoch 57/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5455 -
accuracy: 0.8252 - precision: 0.8793 - recall: 0.7753
Epoch 57: val_loss did not improve from 0.50945
1252/1252 [==============================] - 286s 228ms/step - loss:
0.5455 - accuracy: 0.8252 - precision: 0.8793 - recall: 0.7753 -
val_loss: 0.5417 - val_accuracy: 0.8296 - val_precision: 0.8678 -
val_recall: 0.8027
Epoch 58/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5519 -
accuracy: 0.8230 - precision: 0.8767 - recall: 0.7733
Epoch 58: val_loss did not improve from 0.50945
1252/1252 [==============================] - 278s 222ms/step - loss:
0.5519 - accuracy: 0.8230 - precision: 0.8767 - recall: 0.7733 -
val_loss: 0.5615 - val_accuracy: 0.8284 - val_precision: 0.8692 -
val_recall: 0.7964
Epoch 59/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5452 -
accuracy: 0.8256 - precision: 0.8775 - recall: 0.7760
Epoch 59: val_loss improved from 0.50945 to 0.50583, saving model to
models\IC_S.keras
1252/1252 [==============================] - 290s 232ms/step - loss:
0.5452 - accuracy: 0.8256 - precision: 0.8775 - recall: 0.7760 -
val_loss: 0.5058 - val_accuracy: 0.8422 - val_precision: 0.8776 -
val_recall: 0.8174
Epoch 60/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5419 -
accuracy: 0.8266 - precision: 0.8788 - recall: 0.7764
Epoch 60: val_loss did not improve from 0.50583
1252/1252 [==============================] - 290s 232ms/step - loss:
0.5419 - accuracy: 0.8266 - precision: 0.8788 - recall: 0.7764 -
val_loss: 0.5911 - val_accuracy: 0.8221 - val_precision: 0.8575 -
val_recall: 0.7943
Epoch 61/100
```

```
1252/1252 [==============================] - ETA: 0s - loss: 0.5431 -
accuracy: 0.8261 - precision: 0.8788 - recall: 0.7776
Epoch 61: val_loss did not improve from 0.50583
1252/1252 [==============================] - 290s 232ms/step - loss:
0.5431 - accuracy: 0.8261 - precision: 0.8788 - recall: 0.7776 -
val_loss: 0.5226 - val_accuracy: 0.8381 - val_precision: 0.8757 -
val_recall: 0.8098
Epoch 62/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5455 -
accuracy: 0.8261 - precision: 0.8794 - recall: 0.7782
Epoch 62: val_loss did not improve from 0.50583
1252/1252 [==============================] - 282s 225ms/step - loss:
0.5455 - accuracy: 0.8261 - precision: 0.8794 - recall: 0.7782 -
val_loss: 0.5598 - val_accuracy: 0.8296 - val_precision: 0.8678 -
val_recall: 0.8030
Epoch 63/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5388 -
accuracy: 0.8276 - precision: 0.8794 - recall: 0.7815
Epoch 63: val_loss did not improve from 0.50583
1252/1252 [==============================] - 282s 225ms/step - loss:
0.5388 - accuracy: 0.8276 - precision: 0.8794 - recall: 0.7815 -
val_loss: 0.5084 - val_accuracy: 0.8466 - val_precision: 0.8804 -
val_recall: 0.8185
Epoch 64/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5346 -
accuracy: 0.8277 - precision: 0.8800 - recall: 0.7825
Epoch 64: val_loss did not improve from 0.50583
1252/1252 [==============================] - 290s 231ms/step - loss:
0.5346 - accuracy: 0.8277 - precision: 0.8800 - recall: 0.7825 -
val_loss: 0.5337 - val_accuracy: 0.8388 - val_precision: 0.8707 -
val_recall: 0.8108
Epoch 65/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5352 -
accuracy: 0.8275 - precision: 0.8813 - recall: 0.7800
Epoch 65: val_loss did not improve from 0.50583
1252/1252 [==============================] - 278s 222ms/step - loss:
0.5352 - accuracy: 0.8275 - precision: 0.8813 - recall: 0.7800 -
val_loss: 0.5825 - val_accuracy: 0.8207 - val_precision: 0.8583 -
val_recall: 0.7951
Epoch 66/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5331 -
accuracy: 0.8285 - precision: 0.8794 - recall: 0.7813
Epoch 66: val_loss improved from 0.50583 to 0.48664, saving model to
models\IC_S.keras
1252/1252 [==============================] - 292s 233ms/step - loss:
0.5331 - accuracy: 0.8285 - precision: 0.8794 - recall: 0.7813 -
val_loss: 0.4866 - val_accuracy: 0.8485 - val_precision: 0.8879 -
val_recall: 0.8220
Epoch 67/100
```

```
1252/1252 [==============================] - ETA: 0s - loss: 0.5307 -
accuracy: 0.8311 - precision: 0.8834 - recall: 0.7836
Epoch 67: val_loss did not improve from 0.48664
1252/1252 [==============================] - 292s 233ms/step - loss:
0.5307 - accuracy: 0.8311 - precision: 0.8834 - recall: 0.7836 -
val_loss: 0.5330 - val_accuracy: 0.8402 - val_precision: 0.8784 -
val_recall: 0.8099
Epoch 68/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5301 -
accuracy: 0.8320 - precision: 0.8822 - recall: 0.7843
Epoch 68: val_loss did not improve from 0.48664
1252/1252 [==============================] - 292s 233ms/step - loss:
0.5301 - accuracy: 0.8320 - precision: 0.8822 - recall: 0.7843 -
val_loss: 0.5303 - val_accuracy: 0.8365 - val_precision: 0.8717 -
val_recall: 0.8083
Epoch 69/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5245 -
accuracy: 0.8329 - precision: 0.8828 - recall: 0.7872
Epoch 69: val_loss did not improve from 0.48664
1252/1252 [==============================] - 293s 234ms/step - loss:
0.5245 - accuracy: 0.8329 - precision: 0.8828 - recall: 0.7872 -
val_loss: 0.5175 - val_accuracy: 0.8384 - val_precision: 0.8769 -
val_recall: 0.8096
Epoch 70/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5212 -
accuracy: 0.8336 - precision: 0.8820 - recall: 0.7883
Epoch 70: val_loss did not improve from 0.48664
1252/1252 [==============================] - 288s 230ms/step - loss:
0.5212 - accuracy: 0.8336 - precision: 0.8820 - recall: 0.7883 -
val_loss: 0.5191 - val_accuracy: 0.8421 - val_precision: 0.8764 -
val_recall: 0.8135
Epoch 71/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5153 -
accuracy: 0.8324 - precision: 0.8820 - recall: 0.7879
Epoch 71: val_loss did not improve from 0.48664
1252/1252 [==============================] - 297s 237ms/step - loss:
0.5153 - accuracy: 0.8324 - precision: 0.8820 - recall: 0.7879 -
val_loss: 0.4919 - val_accuracy: 0.8444 - val_precision: 0.8812 -
val_recall: 0.8160
Epoch 72/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5245 -
accuracy: 0.8329 - precision: 0.8836 - recall: 0.7859
Epoch 72: val_loss did not improve from 0.48664
1252/1252 [==============================] - 296s 236ms/step - loss:
0.5245 - accuracy: 0.8329 - precision: 0.8836 - recall: 0.7859 -
val_loss: 0.5128 - val_accuracy: 0.8409 - val_precision: 0.8758 -
val_recall: 0.8182
Epoch 73/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5140 -
```

```
accuracy: 0.8354 - precision: 0.8844 - recall: 0.7904
Epoch 73: val_loss did not improve from 0.48664
1252/1252 [==============================] - 294s 235ms/step - loss:
0.5140 - accuracy: 0.8354 - precision: 0.8844 - recall: 0.7904 -
val_loss: 0.5195 - val_accuracy: 0.8438 - val_precision: 0.8753 -
val_recall: 0.8195
Epoch 74/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5176 -
accuracy: 0.8335 - precision: 0.8835 - recall: 0.7893
Epoch 74: val_loss did not improve from 0.48664
1252/1252 [==============================] - 295s 236ms/step - loss:
0.5176 - accuracy: 0.8335 - precision: 0.8835 - recall: 0.7893 -
val_loss: 0.5859 - val_accuracy: 0.8217 - val_precision: 0.8589 -
val_recall: 0.7957
Epoch 75/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5183 -
accuracy: 0.8353 - precision: 0.8845 - recall: 0.7891
Epoch 75: val_loss did not improve from 0.48664
1252/1252 [==============================] - 291s 232ms/step - loss:
0.5183 - accuracy: 0.8353 - precision: 0.8845 - recall: 0.7891 -
val_loss: 0.5700 - val_accuracy: 0.8300 - val_precision: 0.8627 -
val_recall: 0.8040
Epoch 76/100
1252/1252 [==============================] - ETA: 0s - loss: 0.5132 -
accuracy: 0.8361 - precision: 0.8844 - recall: 0.7911
Epoch 76: val_loss did not improve from 0.48664
1252/1252 [==============================] - 293s 234ms/step - loss:
0.5132 - accuracy: 0.8361 - precision: 0.8844 - recall: 0.7911 -
val_loss: 0.4889 - val_accuracy: 0.8479 - val_precision: 0.8826 -
val_recall: 0.8258
```

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.

```python
# Carregar o modelo
model = keras.models.load_model('models/IC_S_B_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 [==============================] - 2s 7ms/step - loss: 0.4995
- accuracy: 0.8440 - precision: 0.8796 - recall: 0.8163
Test Accuracy: 0.843999981880188
Test Loss: 0.4994990825653076
Test Precision: 0.8796336054801941
Test Recall: 0.8162999749183655
```

4. Análise de resultados

4.1 Evolução das métricas durante o processo de treino

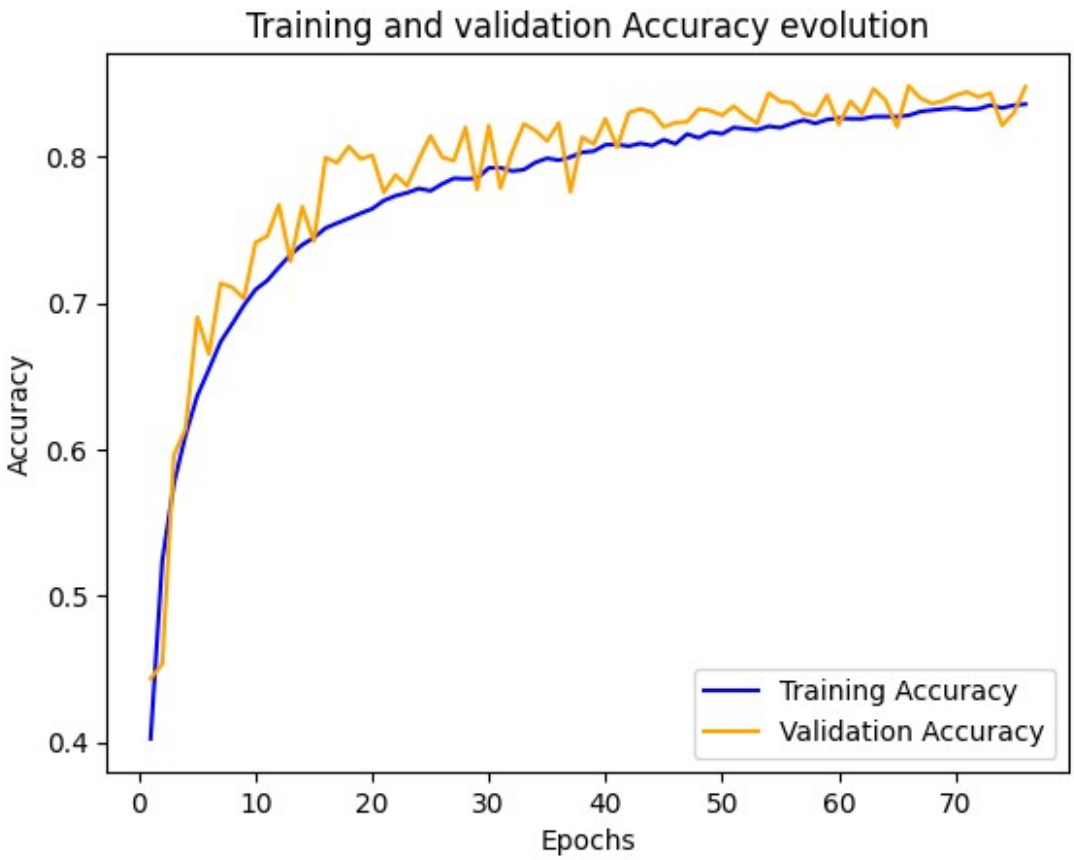
São utilizados gráficos para melhor compreender de que maneira as métricas, nomeadamente a accuracy, loss, precision e recall, foram evoluindo ao longo do processo de treino.

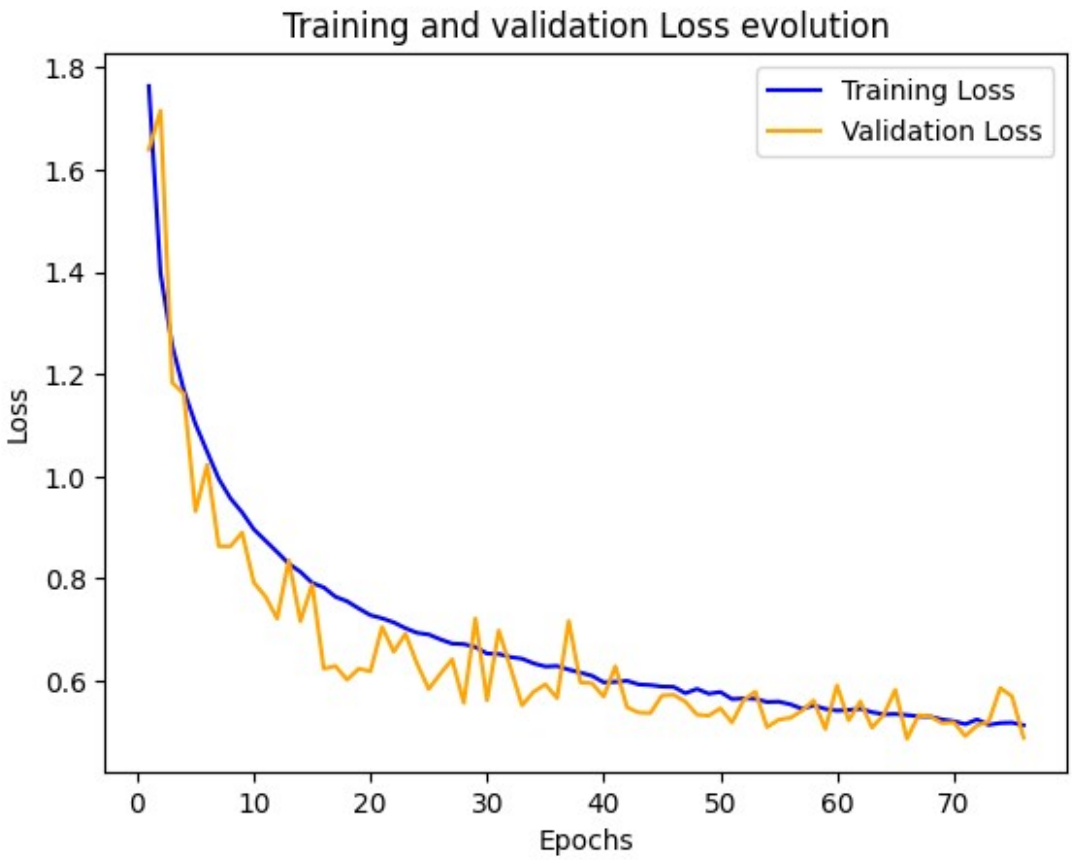
É possível visualizar que:

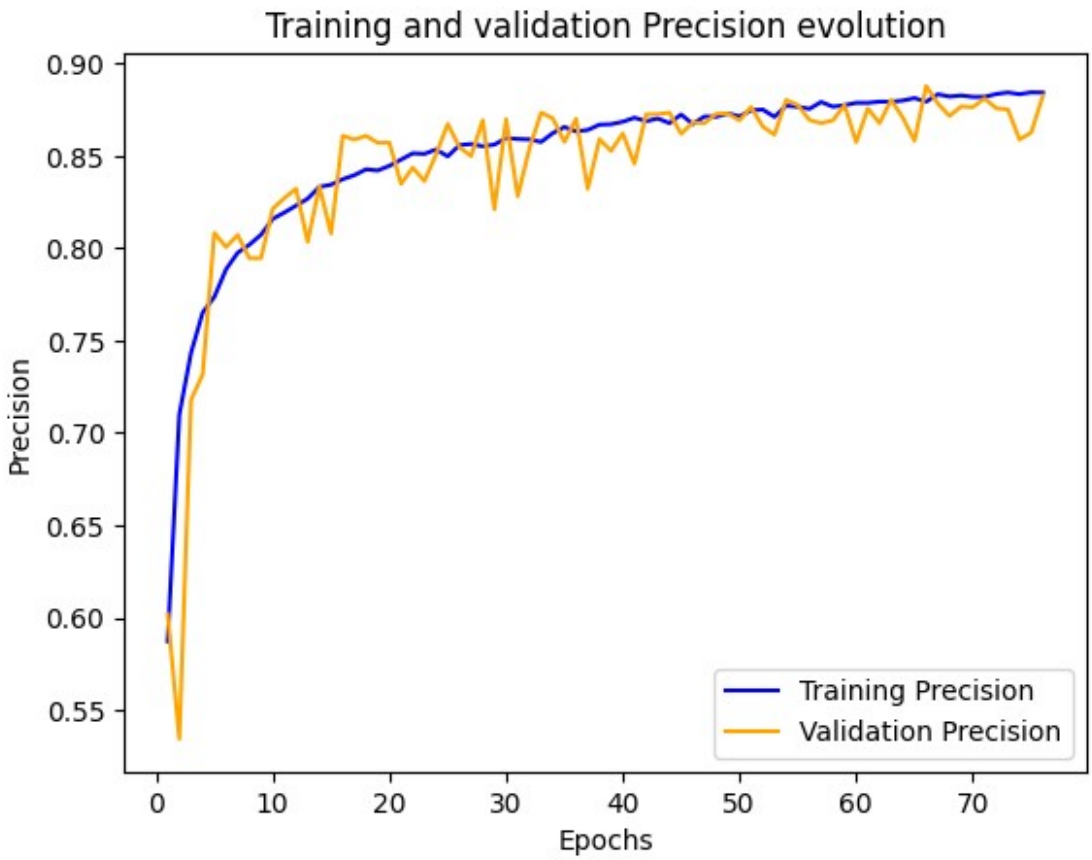
- A operação de data augmentation funcionou no sentido de combater o overfitting presente no modelo treinado sem Data Augmentation.
- É possível perceber que as imagens do dataset de validação não estão a ser corretamente representadas (linhas de validação inconstantes, com altos e baixos).
- O modelo está a sofrer de underfitting (é incapaz de capturar features suficientes das imagens de treino, o que tem como consequência uma estagnação ou mesmo deterioração da sua capacidade de classificação). Isto acontece porque as imagens do dataset de validação não oferecem informação suficiente para avaliar corretamente a capacidade de generalização do modelo.

```python
# 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
Accuracy
Epochs
0.4
0.5
0.6
0.7
0.8
0
10
20
30
40
50
60
70
Training Accuracy
Validation Accuracy


Training and validation Loss evolution
Training Loss
Validation Loss
Loss
Epochs
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0
10
20
30
40
50
60
70


Training and validation Precision evolution
Precision
Epochs
Training Precision
Validation Precision
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
10
20
30
40
50
60
70

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").

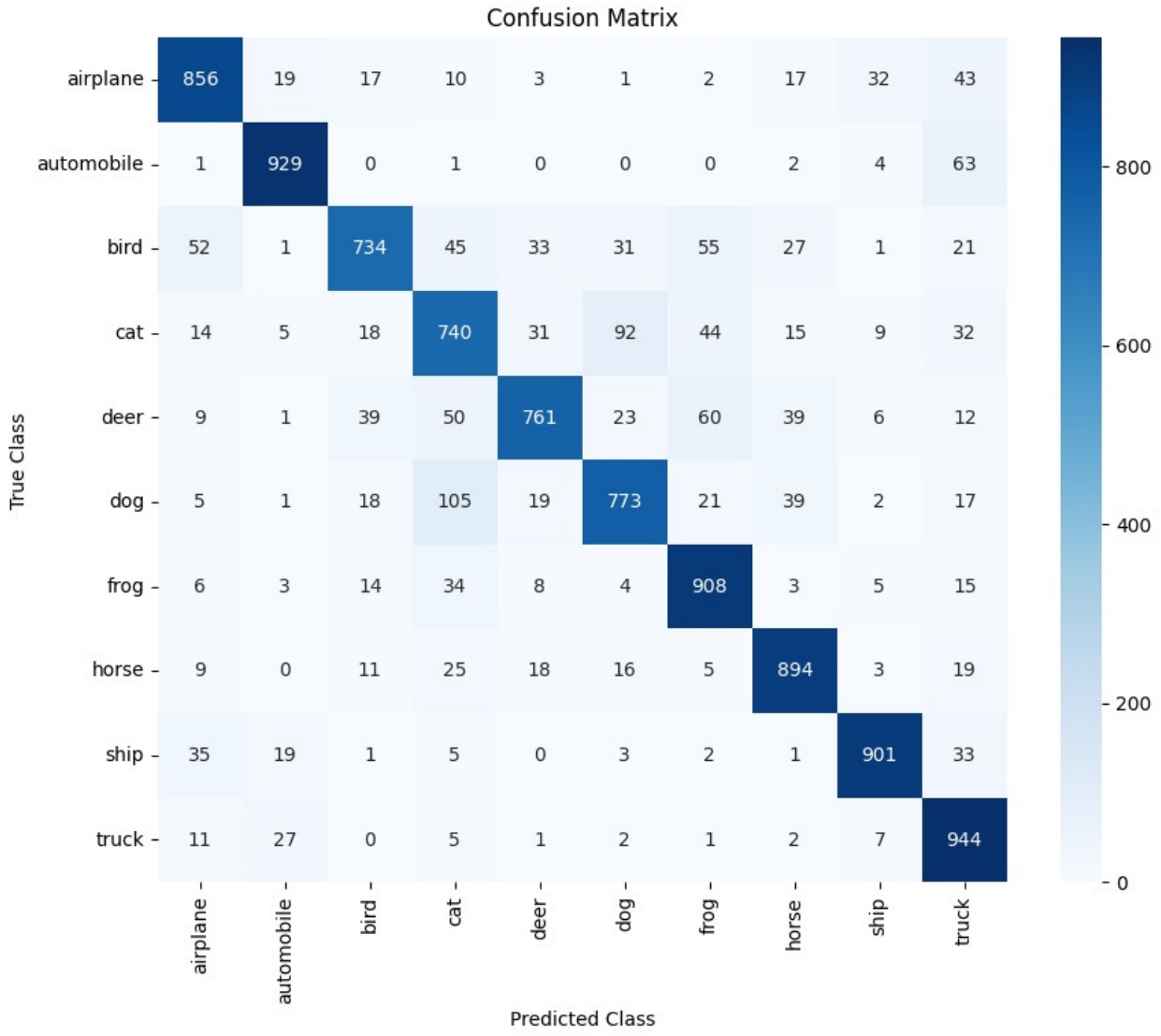
Com isto, podemos compreender que:

- O desempenho do modelo manteve-se, sendo que, algumas das métricas obtidas para certas classes sofreram alterações. Existem casos em que estas alterações melhoram a métrica, como é o caso da precision para a class Bird, e casos em que pioraram a métricas, como é o caso do recall para a classe Deer.
- Não existem classes onde o modelo é concretamente fraco na sua tarefa de prever mas, é possível visualizar que as seguintes classes obtiveram menos previsões corretas que as restantes:
  - Bird

  - Cat
  - Dog
  - Deer
- Com isto, é importante notar a ligeira melhoria no equilibro das previsões que o modelo é capaz de produzir.

```
# 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       0.86      0.86      0.86      1000
  automobile       0.92      0.93      0.93      1000
        bird       0.86      0.73      0.79      1000
         cat       0.73      0.74      0.73      1000
        deer       0.87      0.76      0.81      1000
         dog       0.82      0.77      0.79      1000
        frog       0.83      0.91      0.87      1000
       horse       0.86      0.89      0.88      1000
        ship       0.93      0.90      0.91      1000
       truck       0.79      0.94      0.86      1000

    accuracy                           0.84     10000
   macro avg       0.85      0.84      0.84     10000
weighted avg       0.85      0.84      0.84     10000
```

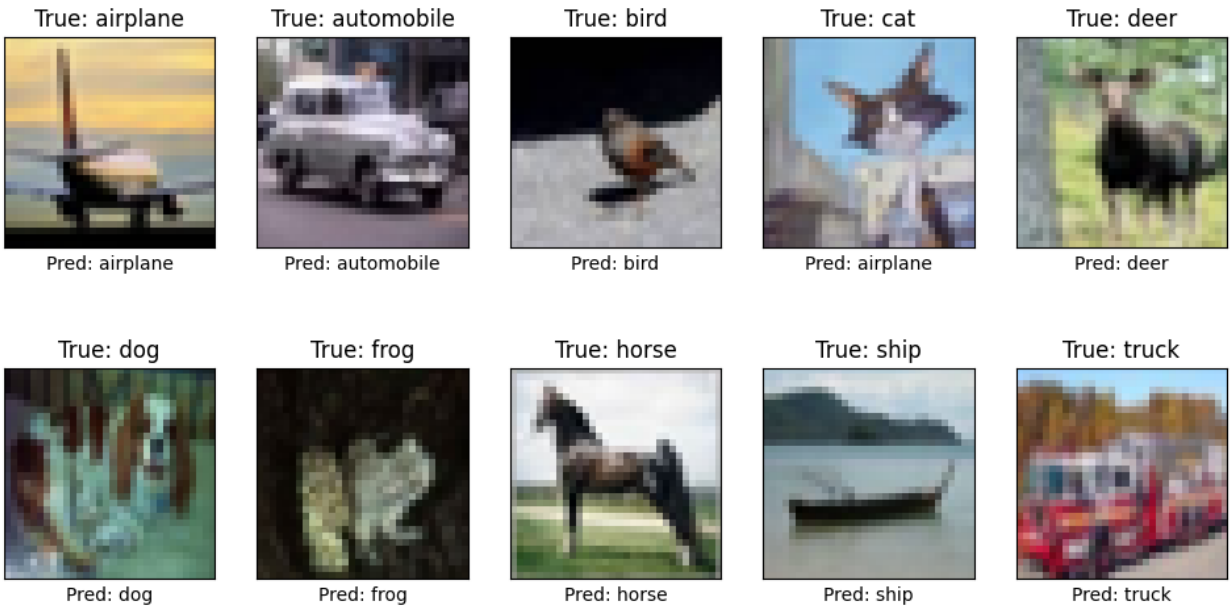


4.3 Visualização de previsões

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

As operações de Data Augmentation conseguiram, com sucesso, resolver o problema de overfitting presente na versão deste modelo treinada sem Data Augmentation. Apesar disso, foram introduzidos outros problemas neste modelo, nomeadamente:

- Underfitting
- Pequena má representação das imagens do dataset de validação

É possível perspectivar que o modelo possui algum espaço para melhoria, nomeadamente no que toca aos problemas supramencionados e às classes em que este possuí mais dificuldade na tarefa de previsão.

Compreendemos que, para melhorar este modelo, é necessário introduzir mais complexidade na arquitetura deste, de modo a resolver o problema de underfitting e de melhorar a capacidade de previsão para as classes onde este apresenta mais dificuldade (mais camadas ou mais filtros, mais capacidade do modelo aprender features de cada classe, o que reduz o underfitting e ajuda a tarefa de previsão). Aqui é, ainda, importante notar a introdução de mais complexidade na arquitetura do modelo pode comprometer a capacidade deste, sendo necessário realizar testes para obter a melhor arquitetura possível.

É importante realçar que, o modelo também foi treinado utilizando um scheduler para o learning rate e o Optuna, um algoritmo de otimização dos hyperparameters. Os resultados que foram obtidos nestes treinos não foram satisfatórios e, foi tomada a decisão de remover estes dois do modelo final. Em ambos os casos o modelo não melhorava significativamente, sendo que, no caso do scheduler, verificámos que o modelo perdia demasiada capacidade de convergência e, no caso do Optuna, o modelo não era capaz de obter resultados significativamente melhores no que toca às métricas de classificação. Tanto o scheduler como o Optuna aumentavam substancialmente o tempo de treino do modelo, o que contribui-o consideravelmente para a decisão de não utilizar estes. Por fim, no caso específico do Optuna é importante realçar que foram, também, feitas experiências utilizando um timeout entre dez e trinta minutos, com o intuito de reduzir o tempo necessário para o treino do modelo, que em nenhum caso foi capaz de obter um resultado superior ao resultado sem utilizar o Optuna (isto acontecia devido ao facto de que, com a Data Augmentation, o Optuna precisava de bastante mais tempo para conseguir acabar uma trial, o que prejudicava significativamente os resultados que este obtia para os hyperparameters pois, não era feito mais do que uma ou duas trials, o que tinha como consequência uma redução considerável dos resultados obtidos pelo 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