```python
import numpy as np
from tensorflow.keras import layers, Model, Input
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, 
ReduceLROnPlateau
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
from tensorflow import keras

# Diretórios de dados
base_dir = '../../Imagens/'

train_dir = os.path.join(base_dir, 'train/train5')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')
```

Load dos dados

- O valor 1./255 significa que vamos dividir cada valor de pixel por 255. Isso resulta em reescalar os valores de pixel de cada imagem para o intervalo de 0 a 1.

```python
# Configuração do ImageDataGenerator
datagen = ImageDataGenerator(rescale=1./255)

IMG_SIZE = 32
BATCH_SIZE = 32
num_classes = 10

train_dataset = datagen.flow_from_directory(
    train_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',

)

validation_dataset = datagen.flow_from_directory(
    validation_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',

)

test_dataset = datagen.flow_from_directory(
    test_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
```

```
)

Found 40000 images belonging to 10 classes.
Found 10000 images belonging to 10 classes.
Found 10000 images belonging to 10 classes.
```

Arquitetura

- A data agumentation é usada para melhorar a generalização e robustez do modelo ao introduzir variações nos dados de treino, como *rescalling*, *zoom* etc...

Regularização L2

- Aplicamos nas camadas convolucionais e densas para penalizar os pesos grandes na função de *loss* durante o treino, o que ajuda a evitar *overfitting* ao reduzir a complexidade do modelo

Otimizador

- O modelo é compilado com o otimizador Adam, que foi o que mais utilizamos nada aulas (optimizer=Adam(learning_rate=0.0001)), e que é amplamente utilizado devido à sua eficiência em ajustar as taxas de aprendizagem de forma adaptativa para cada parâmetro da rede neuronal.

Função Loss

- A função de *loss* escolhida foi a categorical_crossentropy, adequada para problemas de classificação multiclasse

Métrica da avaliação

- A métrica de avaliação durante o treino foi a acurácia (metrics=['accuracy']), que mede a proporção de *predicts* corretas em relação ao total de previsões

Layers Convolucionis

- Cada uma das camadas convolucionais (Conv2D) é seguida pela ativação ReLU (activation='relu'), utilizando padding do tipo 'same' para manter o tamanho da saída igual ao da entrada (padding='same').

- Além disso, aplicámos regularização do kernel através do kernel_regularizer=l2(0.0001), que aplica regularização L2 para ajudar a evitar overfitting como já explicado anteriormente.
- Após cada camada convolucional, aplicámos normalização de batch (BatchNormalization) para acelerar o treino e melhorar a estabilidade do modelo.
- Posteriormente, é utilizado um pooling máximo (MaxPooling2D) com uma janela de (2, 2) para reduzir a dimensionalidade dos dados e extrair características mais importantes da imagem.

Camada Flatten

- Transforma a saída das camadas convolucionais em um vetor unidimensional para ligar a parte convolucional à *fully connected* da rede

Camadas Fully Connected

- A primeira camada densa (Dense) com 512 unidades, ativação ReLU, e regularização do kernel através do kernel_regularizer=l2(0.001). Normalização de batch (BatchNormalization) e dropout de 30% (Dropout(0.3)) para a regularização e prevenção do *overfitting*.

```
# Definindo o input
inputs = Input(shape=(32, 32, 3))

# Primeira camada convolucional
x = layers.Conv2D(32, (3, 3), activation='relu', padding='same',
kernel_regularizer=l2(0.001))(inputs)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D((2, 2))(x)
x = layers.Dropout(0.3)(x)

# Segunda camada convolucional
x = layers.Conv2D(64, (3, 3), activation='relu', padding='same',
kernel_regularizer=l2(0.001))(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D((2, 2))(x)
x = layers.Dropout(0.4)(x)

# Terceira camada convolucional
x = layers.Conv2D(128, (3, 3), activation='relu', padding='same',
kernel_regularizer=l2(0.001))(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D((2, 2))(x)
x = layers.Dropout(0.4)(x)

# Quarta camada convolucional
x = layers.Conv2D(256, (3, 3), activation='relu', padding='same',
kernel_regularizer=l2(0.001))(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D((2, 2))(x)
x = layers.Dropout(0.4)(x)

# Camada de Flatten
x = layers.Flatten()(x)

# Camada totalmente conectada
x = layers.Dense(512, activation='relu', kernel_regularizer=l2(0.001))
(x)
x = layers.BatchNormalization()(x)
x = layers.Dropout(0.5)(x)

# Camada de saída
```

```python
outputs = layers.Dense(10, activation='softmax')(x)  # Supondo 10 classes

# Definindo o modelo
model = Model(inputs=inputs, outputs=outputs)

# Compilando o modelo
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

```
WARNING:tensorflow:From c:\Users\MvCrespo\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\backend.py:1398: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From c:\Users\MvCrespo\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\normalization\batch_normalization.py:979: The name tf.nn.fused_batch_norm is deprecated. Please use tf.compat.v1.nn.fused_batch_norm instead.
```

```python
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 3)]       0         
                                                                 
 conv2d (Conv2D)             (None, 32, 32, 32)        896       
                                                                 
 batch_normalization (Batch  (None, 32, 32, 32)        128       
 Normalization)                                                  
                                                                 
 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0         
 D)                                                              
                                                                 
 dropout (Dropout)           (None, 16, 16, 32)        0         
                                                                 
 conv2d_1 (Conv2D)           (None, 16, 16, 64)        18496     
                                                                 
 batch_normalization_1 (Bat  (None, 16, 16, 64)        256       
 chNormalization)                                                
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 8, 8, 64)          0         
 g2D)                                                            
                                                                 
 dropout_1 (Dropout)         (None, 8, 8, 64)          0         
```

```
 conv2d_2 (Conv2D)           (None, 8, 8, 128)         73856

 batch_normalization_2 (Bat  (None, 8, 8, 128)         512
 chNormalization)

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

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

 conv2d_3 (Conv2D)           (None, 4, 4, 256)         295168

 batch_normalization_3 (Bat  (None, 4, 4, 256)         1024
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 2, 2, 256)         0
 g2D)

 dropout_3 (Dropout)         (None, 2, 2, 256)         0

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 512)               524800

 batch_normalization_4 (Bat  (None, 512)               2048
 chNormalization)

 dropout_4 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 922314 (3.52 MB)
Trainable params: 920330 (3.51 MB)
Non-trainable params: 1984 (7.75 KB)
_________________________________________________________________
```

Callbacks

- Uma mais valia para os treinos que fizemos foi o early_stopping que, se o val_loss nao mudar durante 10 epocas ele parar de treinar e fica com os melhores pesos que teve
- ReduceLr vai reduzindo o lr se mantiver o val_loss com 0.1 durante 5 vezes

```
# Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1,
patience=5, min_lr=0.000001)
```

```python
# Treinar o modelo
history = model.fit(train_dataset,
epochs=100 ,validation_data=validation_dataset,
callbacks=[early_stopping, reduce_lr])
```

```
Epoch 1/100
WARNING:tensorflow:From c:\Users\MvCrespo\AppData\Local\Programs\
Python\Python311\Lib\site-packages\keras\src\utils\tf_utils.py:492:
The name tf.ragged.RaggedTensorValue is deprecated. Please use
tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From c:\Users\MvCrespo\AppData\Local\Programs\
Python\Python311\Lib\site-packages\keras\src\engine\
base_layer_utils.py:384: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

1250/1250 [==============================] - 48s 37ms/step - loss:
2.9140 - accuracy: 0.3518 - val_loss: 2.7124 - val_accuracy: 0.3726 -
lr: 0.0010
Epoch 2/100
1250/1250 [==============================] - 32s 26ms/step - loss:
2.1544 - accuracy: 0.4887 - val_loss: 2.0665 - val_accuracy: 0.4852 -
lr: 0.0010
Epoch 3/100
1250/1250 [==============================] - 33s 26ms/step - loss:
1.8887 - accuracy: 0.5470 - val_loss: 1.8090 - val_accuracy: 0.5859 -
lr: 0.0010
Epoch 4/100
1250/1250 [==============================] - 31s 25ms/step - loss:
1.7853 - accuracy: 0.5810 - val_loss: 1.8056 - val_accuracy: 0.5750 -
lr: 0.0010
Epoch 5/100
1250/1250 [==============================] - 32s 25ms/step - loss:
1.7666 - accuracy: 0.6015 - val_loss: 2.1665 - val_accuracy: 0.5122 -
lr: 0.0010
Epoch 6/100
1250/1250 [==============================] - 31s 25ms/step - loss:
1.7488 - accuracy: 0.6225 - val_loss: 1.5829 - val_accuracy: 0.6864 -
lr: 0.0010
Epoch 7/100
1250/1250 [==============================] - 31s 25ms/step - loss:
1.7280 - accuracy: 0.6334 - val_loss: 1.6542 - val_accuracy: 0.6604 -
lr: 0.0010
Epoch 8/100
1250/1250 [==============================] - 33s 26ms/step - loss:
1.7065 - accuracy: 0.6406 - val_loss: 1.6110 - val_accuracy: 0.6680 -
lr: 0.0010
Epoch 9/100
1250/1250 [==============================] - 33s 27ms/step - loss:
```

```
1.6784 - accuracy: 0.6471 - val_loss: 2.2231 - val_accuracy: 0.4950 -
lr: 0.0010
Epoch 10/100
1250/1250 [==============================] - 32s 26ms/step - loss:
1.6604 - accuracy: 0.6539 - val_loss: 1.8445 - val_accuracy: 0.5724 -
lr: 0.0010
Epoch 11/100
1250/1250 [==============================] - 32s 25ms/step - loss:
1.6477 - accuracy: 0.6562 - val_loss: 1.5473 - val_accuracy: 0.6860 -
lr: 0.0010
Epoch 12/100
1250/1250 [==============================] - 32s 25ms/step - loss:
1.6367 - accuracy: 0.6643 - val_loss: 1.5036 - val_accuracy: 0.7075 -
lr: 0.0010
Epoch 13/100
1250/1250 [==============================] - 32s 25ms/step - loss:
1.6152 - accuracy: 0.6665 - val_loss: 1.7178 - val_accuracy: 0.6281 -
lr: 0.0010
Epoch 14/100
1250/1250 [==============================] - 32s 25ms/step - loss:
1.6154 - accuracy: 0.6678 - val_loss: 1.6076 - val_accuracy: 0.6715 -
lr: 0.0010
Epoch 15/100
1250/1250 [==============================] - 32s 25ms/step - loss:
1.5981 - accuracy: 0.6735 - val_loss: 1.8258 - val_accuracy: 0.5889 -
lr: 0.0010
Epoch 16/100
1250/1250 [==============================] - 32s 25ms/step - loss:
1.5854 - accuracy: 0.6708 - val_loss: 1.4636 - val_accuracy: 0.7138 -
lr: 0.0010
Epoch 17/100
1250/1250 [==============================] - 32s 25ms/step - loss:
1.5880 - accuracy: 0.6740 - val_loss: 1.5357 - val_accuracy: 0.6842 -
lr: 0.0010
Epoch 18/100
1250/1250 [==============================] - 31s 25ms/step - loss:
1.5764 - accuracy: 0.6787 - val_loss: 1.6040 - val_accuracy: 0.6631 -
lr: 0.0010
Epoch 19/100
1250/1250 [==============================] - 30s 24ms/step - loss:
1.5670 - accuracy: 0.6781 - val_loss: 1.3600 - val_accuracy: 0.7568 -
lr: 0.0010
Epoch 20/100
1250/1250 [==============================] - 30s 24ms/step - loss:
1.5617 - accuracy: 0.6814 - val_loss: 1.7715 - val_accuracy: 0.6172 -
lr: 0.0010
Epoch 21/100
1250/1250 [==============================] - 31s 24ms/step - loss:
1.5553 - accuracy: 0.6826 - val_loss: 1.4628 - val_accuracy: 0.7157 -
```

```
lr: 0.0010
Epoch 22/100
1250/1250 [==============================] - 31s 24ms/step - loss:
1.5571 - accuracy: 0.6828 - val_loss: 1.4431 - val_accuracy: 0.7189 -
lr: 0.0010
Epoch 23/100
1250/1250 [==============================] - 31s 24ms/step - loss:
1.5414 - accuracy: 0.6873 - val_loss: 1.3747 - val_accuracy: 0.7500 -
lr: 0.0010
Epoch 24/100
1250/1250 [==============================] - 31s 24ms/step - loss:
1.5481 - accuracy: 0.6866 - val_loss: 1.4082 - val_accuracy: 0.7261 -
lr: 0.0010
Epoch 25/100
1250/1250 [==============================] - 30s 24ms/step - loss:
1.3726 - accuracy: 0.7293 - val_loss: 1.1754 - val_accuracy: 0.7785 -
lr: 1.0000e-04
Epoch 26/100
1250/1250 [==============================] - 30s 24ms/step - loss:
1.2508 - accuracy: 0.7453 - val_loss: 1.1112 - val_accuracy: 0.7799 -
lr: 1.0000e-04
Epoch 27/100
1250/1250 [==============================] - 31s 25ms/step - loss:
1.1728 - accuracy: 0.7535 - val_loss: 1.0376 - val_accuracy: 0.7893 -
lr: 1.0000e-04
Epoch 28/100
1250/1250 [==============================] - 33s 26ms/step - loss:
1.1135 - accuracy: 0.7570 - val_loss: 0.9844 - val_accuracy: 0.7972 -
lr: 1.0000e-04
Epoch 29/100
1250/1250 [==============================] - 31s 24ms/step - loss:
1.0697 - accuracy: 0.7633 - val_loss: 0.9623 - val_accuracy: 0.7933 -
lr: 1.0000e-04
Epoch 30/100
1250/1250 [==============================] - 31s 24ms/step - loss:
1.0298 - accuracy: 0.7668 - val_loss: 0.8961 - val_accuracy: 0.8062 -
lr: 1.0000e-04
Epoch 31/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.9974 - accuracy: 0.7704 - val_loss: 0.8868 - val_accuracy: 0.8052 -
lr: 1.0000e-04
Epoch 32/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.9724 - accuracy: 0.7743 - val_loss: 0.8555 - val_accuracy: 0.8115 -
lr: 1.0000e-04
Epoch 33/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.9421 - accuracy: 0.7799 - val_loss: 0.8172 - val_accuracy: 0.8216 -
lr: 1.0000e-04
```

```
Epoch 34/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.9285 - accuracy: 0.7800 - val_loss: 0.8668 - val_accuracy: 0.8007 -
lr: 1.0000e-04
Epoch 35/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.9071 - accuracy: 0.7860 - val_loss: 0.8423 - val_accuracy: 0.8068 -
lr: 1.0000e-04
Epoch 36/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8914 - accuracy: 0.7872 - val_loss: 0.7927 - val_accuracy: 0.8192 -
lr: 1.0000e-04
Epoch 37/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8757 - accuracy: 0.7864 - val_loss: 0.7892 - val_accuracy: 0.8177 -
lr: 1.0000e-04
Epoch 38/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8658 - accuracy: 0.7887 - val_loss: 0.8273 - val_accuracy: 0.8022 -
lr: 1.0000e-04
Epoch 39/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8540 - accuracy: 0.7920 - val_loss: 0.7632 - val_accuracy: 0.8219 -
lr: 1.0000e-04
Epoch 40/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8399 - accuracy: 0.7956 - val_loss: 0.7822 - val_accuracy: 0.8160 -
lr: 1.0000e-04
Epoch 41/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8328 - accuracy: 0.7966 - val_loss: 0.7462 - val_accuracy: 0.8224 -
lr: 1.0000e-04
Epoch 42/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8155 - accuracy: 0.8001 - val_loss: 0.7619 - val_accuracy: 0.8185 -
lr: 1.0000e-04
Epoch 43/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8174 - accuracy: 0.7983 - val_loss: 0.7540 - val_accuracy: 0.8198 -
lr: 1.0000e-04
Epoch 44/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8143 - accuracy: 0.7991 - val_loss: 0.7396 - val_accuracy: 0.8203 -
lr: 1.0000e-04
Epoch 45/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.8013 - accuracy: 0.8005 - val_loss: 0.7688 - val_accuracy: 0.8124 -
lr: 1.0000e-04
Epoch 46/100
```

```
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7963 - accuracy: 0.8023 - val_loss: 0.7277 - val_accuracy: 0.8242 -
lr: 1.0000e-04
Epoch 47/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7891 - accuracy: 0.8037 - val_loss: 0.7178 - val_accuracy: 0.8280 -
lr: 1.0000e-04
Epoch 48/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7884 - accuracy: 0.8045 - val_loss: 0.7105 - val_accuracy: 0.8355 -
lr: 1.0000e-04
Epoch 49/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7860 - accuracy: 0.8059 - val_loss: 0.7217 - val_accuracy: 0.8236 -
lr: 1.0000e-04
Epoch 50/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7785 - accuracy: 0.8059 - val_loss: 0.7253 - val_accuracy: 0.8231 -
lr: 1.0000e-04
Epoch 51/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7723 - accuracy: 0.8070 - val_loss: 0.6998 - val_accuracy: 0.8328 -
lr: 1.0000e-04
Epoch 52/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7606 - accuracy: 0.8110 - val_loss: 0.7349 - val_accuracy: 0.8232 -
lr: 1.0000e-04
Epoch 53/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7656 - accuracy: 0.8081 - val_loss: 0.7068 - val_accuracy: 0.8284 -
lr: 1.0000e-04
Epoch 54/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7604 - accuracy: 0.8092 - val_loss: 0.7112 - val_accuracy: 0.8262 -
lr: 1.0000e-04
Epoch 55/100
1250/1250 [==============================] - 33s 27ms/step - loss:
0.7609 - accuracy: 0.8104 - val_loss: 0.7380 - val_accuracy: 0.8203 -
lr: 1.0000e-04
Epoch 56/100
1250/1250 [==============================] - 37s 30ms/step - loss:
0.7528 - accuracy: 0.8104 - val_loss: 0.6989 - val_accuracy: 0.8347 -
lr: 1.0000e-04
Epoch 57/100
1250/1250 [==============================] - 32s 26ms/step - loss:
0.7429 - accuracy: 0.8130 - val_loss: 0.7007 - val_accuracy: 0.8306 -
lr: 1.0000e-04
Epoch 58/100
1250/1250 [==============================] - 31s 25ms/step - loss:
```

```
0.7480 - accuracy: 0.8139 - val_loss: 0.7067 - val_accuracy: 0.8311 -
lr: 1.0000e-04
Epoch 59/100
1250/1250 [==============================] - 31s 24ms/step - loss:
0.7437 - accuracy: 0.8152 - val_loss: 0.7088 - val_accuracy: 0.8293 -
lr: 1.0000e-04
Epoch 60/100
1250/1250 [==============================] - 30s 24ms/step - loss:
0.7406 - accuracy: 0.8139 - val_loss: 0.7195 - val_accuracy: 0.8252 -
lr: 1.0000e-04
Epoch 61/100
1250/1250 [==============================] - 30s 24ms/step - loss:
0.7393 - accuracy: 0.8153 - val_loss: 0.6903 - val_accuracy: 0.8342 -
lr: 1.0000e-04
Epoch 62/100
1250/1250 [==============================] - 31s 24ms/step - loss:
0.7372 - accuracy: 0.8153 - val_loss: 0.6846 - val_accuracy: 0.8356 -
lr: 1.0000e-04
Epoch 63/100
1250/1250 [==============================] - 31s 24ms/step - loss:
0.7345 - accuracy: 0.8145 - val_loss: 0.6892 - val_accuracy: 0.8324 -
lr: 1.0000e-04
Epoch 64/100
1250/1250 [==============================] - 33s 27ms/step - loss:
0.7335 - accuracy: 0.8170 - val_loss: 0.7053 - val_accuracy: 0.8274 -
lr: 1.0000e-04
Epoch 65/100
1250/1250 [==============================] - 35s 28ms/step - loss:
0.7273 - accuracy: 0.8189 - val_loss: 0.6785 - val_accuracy: 0.8381 -
lr: 1.0000e-04
Epoch 66/100
1250/1250 [==============================] - 34s 27ms/step - loss:
0.7330 - accuracy: 0.8164 - val_loss: 0.6917 - val_accuracy: 0.8320 -
lr: 1.0000e-04
Epoch 67/100
1250/1250 [==============================] - 34s 27ms/step - loss:
0.7276 - accuracy: 0.8177 - val_loss: 0.6942 - val_accuracy: 0.8344 -
lr: 1.0000e-04
Epoch 68/100
1250/1250 [==============================] - 34s 27ms/step - loss:
0.7263 - accuracy: 0.8184 - val_loss: 0.7128 - val_accuracy: 0.8261 -
lr: 1.0000e-04
Epoch 69/100
1250/1250 [==============================] - 34s 27ms/step - loss:
0.7254 - accuracy: 0.8195 - val_loss: 0.7090 - val_accuracy: 0.8262 -
lr: 1.0000e-04
Epoch 70/100
1250/1250 [==============================] - 34s 27ms/step - loss:
0.7199 - accuracy: 0.8225 - val_loss: 0.6753 - val_accuracy: 0.8389 -
```

```
lr: 1.0000e-04
Epoch 71/100
1250/1250 [==============================] - 34s 27ms/step - loss:
0.7265 - accuracy: 0.8179 - val_loss: 0.6977 - val_accuracy: 0.8309 -
lr: 1.0000e-04
Epoch 72/100
1250/1250 [==============================] - 34s 27ms/step - loss:
0.7218 - accuracy: 0.8213 - val_loss: 0.7126 - val_accuracy: 0.8239 -
lr: 1.0000e-04
Epoch 73/100
1250/1250 [==============================] - 34s 27ms/step - loss:
0.7199 - accuracy: 0.8215 - val_loss: 0.7213 - val_accuracy: 0.8208 -
lr: 1.0000e-04
Epoch 74/100
1250/1250 [==============================] - 34s 27ms/step - loss:
0.7201 - accuracy: 0.8203 - val_loss: 0.7033 - val_accuracy: 0.8269 -
lr: 1.0000e-04
Epoch 75/100
1250/1250 [==============================] - 32s 26ms/step - loss:
0.7226 - accuracy: 0.8209 - val_loss: 0.6666 - val_accuracy: 0.8450 -
lr: 1.0000e-04
Epoch 76/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7165 - accuracy: 0.8217 - val_loss: 0.7116 - val_accuracy: 0.8272 -
lr: 1.0000e-04
Epoch 77/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7187 - accuracy: 0.8195 - val_loss: 0.7160 - val_accuracy: 0.8244 -
lr: 1.0000e-04
Epoch 78/100
1250/1250 [==============================] - 32s 26ms/step - loss:
0.7139 - accuracy: 0.8240 - val_loss: 0.6947 - val_accuracy: 0.8333 -
lr: 1.0000e-04
Epoch 79/100
1250/1250 [==============================] - 32s 25ms/step - loss:
0.7127 - accuracy: 0.8244 - val_loss: 0.6845 - val_accuracy: 0.8350 -
lr: 1.0000e-04
Epoch 80/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.7149 - accuracy: 0.8234 - val_loss: 0.6738 - val_accuracy: 0.8388 -
lr: 1.0000e-04
Epoch 81/100
1250/1250 [==============================] - 32s 25ms/step - loss:
0.6862 - accuracy: 0.8314 - val_loss: 0.6734 - val_accuracy: 0.8388 -
lr: 1.0000e-05
Epoch 82/100
1250/1250 [==============================] - 32s 26ms/step - loss:
0.6779 - accuracy: 0.8342 - val_loss: 0.6627 - val_accuracy: 0.8421 -
lr: 1.0000e-05
```

```
Epoch 83/100
1250/1250 [==============================] - 38s 30ms/step - loss:
0.6690 - accuracy: 0.8373 - val_loss: 0.6643 - val_accuracy: 0.8411 -
lr: 1.0000e-05
Epoch 84/100
1250/1250 [==============================] - 35s 28ms/step - loss:
0.6717 - accuracy: 0.8350 - val_loss: 0.6636 - val_accuracy: 0.8406 -
lr: 1.0000e-05
Epoch 85/100
1250/1250 [==============================] - 32s 26ms/step - loss:
0.6669 - accuracy: 0.8375 - val_loss: 0.6579 - val_accuracy: 0.8435 -
lr: 1.0000e-05
Epoch 86/100
1250/1250 [==============================] - 32s 26ms/step - loss:
0.6541 - accuracy: 0.8421 - val_loss: 0.6580 - val_accuracy: 0.8432 -
lr: 1.0000e-05
Epoch 87/100
1250/1250 [==============================] - 31s 25ms/step - loss:
0.6594 - accuracy: 0.8379 - val_loss: 0.6611 - val_accuracy: 0.8406 -
lr: 1.0000e-05
Epoch 88/100
1250/1250 [==============================] - 32s 26ms/step - loss:
0.6489 - accuracy: 0.8417 - val_loss: 0.6553 - val_accuracy: 0.8426 -
lr: 1.0000e-05
Epoch 89/100
1250/1250 [==============================] - 32s 25ms/step - loss:
0.6470 - accuracy: 0.8413 - val_loss: 0.6536 - val_accuracy: 0.8438 -
lr: 1.0000e-05
Epoch 90/100
1250/1250 [==============================] - 32s 26ms/step - loss:
0.6476 - accuracy: 0.8419 - val_loss: 0.6571 - val_accuracy: 0.8436 -
lr: 1.0000e-05
Epoch 91/100
1250/1250 [==============================] - 33s 26ms/step - loss:
0.6416 - accuracy: 0.8428 - val_loss: 0.6491 - val_accuracy: 0.8446 -
lr: 1.0000e-05
Epoch 92/100
1250/1250 [==============================] - 33s 27ms/step - loss:
0.6327 - accuracy: 0.8476 - val_loss: 0.6480 - val_accuracy: 0.8455 -
lr: 1.0000e-05
Epoch 93/100
1250/1250 [==============================] - 34s 28ms/step - loss:
0.6403 - accuracy: 0.8460 - val_loss: 0.6524 - val_accuracy: 0.8427 -
lr: 1.0000e-05
Epoch 94/100
1250/1250 [==============================] - 35s 28ms/step - loss:
0.6382 - accuracy: 0.8429 - val_loss: 0.6494 - val_accuracy: 0.8445 -
lr: 1.0000e-05
Epoch 95/100
```

```
1250/1250 [==============================] - 36s 29ms/step - loss: 0.6354 - accuracy: 0.8432 - val_loss: 0.6514 - val_accuracy: 0.8434 - lr: 1.0000e-05
Epoch 96/100
1250/1250 [==============================] - 35s 28ms/step - loss: 0.6278 - accuracy: 0.8468 - val_loss: 0.6411 - val_accuracy: 0.8464 - lr: 1.0000e-05
Epoch 97/100
1250/1250 [==============================] - 33s 26ms/step - loss: 0.6335 - accuracy: 0.8456 - val_loss: 0.6453 - val_accuracy: 0.8440 - lr: 1.0000e-05
Epoch 98/100
1250/1250 [==============================] - 33s 26ms/step - loss: 0.6281 - accuracy: 0.8478 - val_loss: 0.6484 - val_accuracy: 0.8439 - lr: 1.0000e-05
Epoch 99/100
1250/1250 [==============================] - 33s 26ms/step - loss: 0.6195 - accuracy: 0.8503 - val_loss: 0.6478 - val_accuracy: 0.8431 - lr: 1.0000e-05
Epoch 100/100
1250/1250 [==============================] - 32s 26ms/step - loss: 0.6183 - accuracy: 0.8493 - val_loss: 0.6394 - val_accuracy: 0.8435 - lr: 1.0000e-05
```

```
#Saving the model
model.save('From_Scratch_Sem_DataAugmentation.h5')
```

```
c:\Users\MvCrespo\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
model = keras.models.load_model('From_Scratch_Sem_DataAugmentation.h5')
# Validacao da Rede
val_loss, val_acc = model.evaluate(validation_dataset)
print('val_acc:', val_acc)

# Avaliar o modelo
test_loss, test_acc = model.evaluate(test_dataset)
print(f'Test accuracy: {test_acc}')
```

```
313/313 [==============================] - 4s 12ms/step - loss: 0.6394 - accuracy: 0.8435
val_acc: 0.843500018119812
313/313 [==============================] - 4s 11ms/step - loss: 0.6599 - accuracy: 0.8439
Test accuracy: 0.8439000248908997
```

```python
# Plotando os resultados
import matplotlib.pyplot as plt

def plot_training_history(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    epochs = range(len(acc))

    plt.figure(figsize=(12, 4))
    plt.subplot(1, 2, 1)
    plt.plot(epochs, acc, 'bo-', label='Training accuracy')
    plt.plot(epochs, val_acc, 'ro-', label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(epochs, loss, 'bo-', label='Training loss')
    plt.plot(epochs, val_loss, 'ro-', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()

    plt.show()

plot_training_history(history)
```