```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
import tensorflow as tf
from keras.utils import image_dataset_from_directory
# Directories
train_dirs = [
    '/content/drive/MyDrive/ProjetoIA/dataset/train1',
    '/content/drive/MyDrive/ProjetoIA/dataset/train2',
    '/content/drive/MyDrive/ProjetoIA/dataset/train4',
    '/content/drive/MyDrive/ProjetoIA/dataset/train5'
]
validation_dir = '/content/drive/MyDrive/ProjetoIA/dataset/train3'
test_dir = '/content/drive/MyDrive/ProjetoIA/dataset/test'

# Parameters
IMG_SIZE = 150
BATCH_SIZE = 32

# Function to load datasets from multiple directories and concatenate them
def load_and_concatenate_datasets(directories, img_size, batch_size):
    datasets = []
    for directory in directories:
        dataset = image_dataset_from_directory(
            directory,
            image_size=(img_size, img_size),
            batch_size=batch_size
        )
        datasets.append(dataset)
    return datasets

# Load train datasets and concatenate
train_datasets = load_and_concatenate_datasets(train_dirs, IMG_SIZE, BATCH_SIZE)
train_dataset = tf.data.Dataset.sample_from_datasets(train_datasets)

# Load validation and test datasets
validation_dataset = image_dataset_from_directory(
    validation_dir,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE
)
test_dataset = image_dataset_from_directory(
    test_dir,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE
)
```

```python
# Extract class names from one of the datasets
example_dataset = image_dataset_from_directory(
    train_dirs[0],
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE
)
class_names = example_dataset.class_names
print(class_names)
```

```
Found 10400 files belonging to 10 classes.
Found 9600 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 1 classes.
Found 10400 files belonging to 10 classes.
['000_airplane', '001_automobile', '002_bird', '003_cat', '004_deer',
'005_dog', '006_frog', '007_horse', '008_ship', '009_truck']
```

```python
#The shape of each batch

for data_batch, labels_batch in train_dataset:
  print('data batch shape:', data_batch.shape)
  print('labels batch shape:', labels_batch.shape)
  break
```

```
data batch shape: (32, 150, 150, 3)
labels batch shape: (32,)
```

```python
#Creating the neural network
from tensorflow import keras
from keras import layers
from keras import models
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
x = layers.Dense(512, activation="relu")(x)
outputs = layers.Dense(10, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 150, 150, 3)]     0         
                                                                 
 rescaling (Rescaling)       (None, 150, 150, 3)       0         
                                                                 
 conv2d (Conv2D)             (None, 148, 148, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 36, 36, 64)        0         
 g2D)                                                            
                                                                 
 conv2d_2 (Conv2D)           (None, 34, 34, 64)        36928     
                                                                 
 max_pooling2d_2 (MaxPoolin  (None, 17, 17, 64)        0         
 g2D)                                                            
                                                                 
 conv2d_3 (Conv2D)           (None, 15, 15, 128)       73856     
                                                                 
 max_pooling2d_3 (MaxPoolin  (None, 7, 7, 128)         0         
 g2D)                                                            
                                                                 
 conv2d_4 (Conv2D)           (None, 5, 5, 128)         147584    
                                                                 
 max_pooling2d_4 (MaxPoolin  (None, 2, 2, 128)         0         
 g2D)                                                            
                                                                 
 flatten (Flatten)           (None, 512)               0         
                                                                 
 dense (Dense)               (None, 512)               262656    
                                                                 
 dense_1 (Dense)             (None, 10)                5130      
                                                                 
=================================================================
Total params: 545546 (2.08 MB)
Trainable params: 545546 (2.08 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
checkpoint_callback = 
ModelCheckpoint(filepath='/content/drive/MyDrive/ProjetoIA/models/
modelS_2_best.h5',
                                      monitor='val_loss',
                                      save_best_only=True,
                                      save_weights_only=False,
                                      verbose=1)

early_stopping_callback = EarlyStopping(monitor='val_loss',
                                        patience=5,
                                        verbose=1)

import tensorflow as tf
from keras.optimizers import Adam

model.compile(optimizer=Adam(learning_rate=1e-4),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

#Training the model
#history guarda todos os parametros gerados durante o treino
history = model.fit(
  train_dataset,
  epochs=50,
  validation_data=validation_dataset,# n usa para melhorar so para 
mostrar ao utilizador
  callbacks=[checkpoint_callback, early_stopping_callback]
  )
```

```
Epoch 1/50
   1251/Unknown - 1205s 960ms/step - loss: 1.7948 - accuracy: 0.3425
Epoch 1: val_loss improved from inf to 1.50047, saving model to 
/content/drive/MyDrive/ProjetoIA/models/modelS_2_best.h5

/usr/local/lib/python3.10/dist-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(

1251/1251 [==============================] - 1582s 1s/step - loss: 
1.7948 - accuracy: 0.3425 - val_loss: 1.5005 - val_accuracy: 0.4568
Epoch 2/50
1251/1251 [==============================] - ETA: 0s - loss: 1.4521 - 
accuracy: 0.4764
Epoch 2: val_loss improved from 1.50047 to 1.40763, saving model to 
/content/drive/MyDrive/ProjetoIA/models/modelS_2_best.h5
1251/1251 [==============================] - 66s 53ms/step - loss: 
1.4521 - accuracy: 0.4764 - val_loss: 1.4076 - val_accuracy: 0.4989
Epoch 3/50
1249/1251 [============================>.] - ETA: 0s - loss: 1.3110 - 
accuracy: 0.5354
```

```
Epoch 3: val_loss improved from 1.40763 to 1.24342, saving model to
/content/drive/MyDrive/ProjetoIA/models/modelS_2_best.h5
1251/1251 [==============================] - 66s 52ms/step - loss:
1.3109 - accuracy: 0.5354 - val_loss: 1.2434 - val_accuracy: 0.5639
Epoch 4/50
1250/1251 [============================>.] - ETA: 0s - loss: 1.2126 -
accuracy: 0.5735
Epoch 4: val_loss improved from 1.24342 to 1.19051, saving model to
/content/drive/MyDrive/ProjetoIA/models/modelS_2_best.h5
1251/1251 [==============================] - 65s 52ms/step - loss:
1.2127 - accuracy: 0.5735 - val_loss: 1.1905 - val_accuracy: 0.5858
Epoch 5/50
1249/1251 [============================>.] - ETA: 0s - loss: 1.1272 -
accuracy: 0.6048
Epoch 5: val_loss improved from 1.19051 to 1.11760, saving model to
/content/drive/MyDrive/ProjetoIA/models/modelS_2_best.h5
1251/1251 [==============================] - 66s 52ms/step - loss:
1.1271 - accuracy: 0.6049 - val_loss: 1.1176 - val_accuracy: 0.6078
Epoch 6/50
1251/1251 [==============================] - ETA: 0s - loss: 1.0581 -
accuracy: 0.6325
Epoch 6: val_loss did not improve from 1.11760
1251/1251 [==============================] - 65s 52ms/step - loss:
1.0581 - accuracy: 0.6325 - val_loss: 1.1258 - val_accuracy: 0.6172
Epoch 7/50
1251/1251 [==============================] - ETA: 0s - loss: 0.9986 -
accuracy: 0.6549
Epoch 7: val_loss improved from 1.11760 to 1.09571, saving model to
/content/drive/MyDrive/ProjetoIA/models/modelS_2_best.h5
1251/1251 [==============================] - 66s 53ms/step - loss:
0.9986 - accuracy: 0.6549 - val_loss: 1.0957 - val_accuracy: 0.6234
Epoch 8/50
1251/1251 [==============================] - ETA: 0s - loss: 0.9398 -
accuracy: 0.6743
Epoch 8: val_loss improved from 1.09571 to 1.01861, saving model to
/content/drive/MyDrive/ProjetoIA/models/modelS_2_best.h5
1251/1251 [==============================] - 66s 52ms/step - loss:
0.9398 - accuracy: 0.6743 - val_loss: 1.0186 - val_accuracy: 0.6475
Epoch 9/50
1251/1251 [==============================] - ETA: 0s - loss: 0.8878 -
accuracy: 0.6933
Epoch 9: val_loss did not improve from 1.01861
1251/1251 [==============================] - 66s 52ms/step - loss:
0.8878 - accuracy: 0.6933 - val_loss: 1.0619 - val_accuracy: 0.6378
Epoch 10/50
1251/1251 [==============================] - ETA: 0s - loss: 0.8422 -
accuracy: 0.7070
Epoch 10: val_loss improved from 1.01861 to 1.01104, saving model
to /content/drive/MyDrive/ProjetoIA/models/modelS_2_best.h5
```

```
1251/1251 [==============================] - 66s 52ms/step - loss:
0.8422 - accuracy: 0.7070 - val_loss: 1.0110 - val_accuracy: 0.6565
Epoch 11/50
1251/1251 [==============================] - ETA: 0s - loss: 0.7913 -
accuracy: 0.7286
Epoch 11: val_loss improved from 1.01104 to 0.96920, saving model
to /content/drive/MyDrive/ProjetoIA/models/modelS_2_best.h5
1251/1251 [==============================] - 65s 52ms/step - loss:
0.7913 - accuracy: 0.7286 - val_loss: 0.9692 - val_accuracy: 0.6683
Epoch 12/50
1251/1251 [==============================] - ETA: 0s - loss: 0.7422 -
accuracy: 0.7432
Epoch 12: val_loss did not improve from 0.96920
1251/1251 [==============================] - 65s 52ms/step - loss:
0.7422 - accuracy: 0.7432 - val_loss: 1.0190 - val_accuracy: 0.6576
Epoch 13/50
1249/1251 [============================>.] - ETA: 0s - loss: 0.6979 -
accuracy: 0.7611
Epoch 13: val_loss did not improve from 0.96920
1251/1251 [==============================] - 65s 52ms/step - loss:
0.6976 - accuracy: 0.7613 - val_loss: 0.9715 - val_accuracy: 0.6766
Epoch 14/50
1251/1251 [==============================] - ETA: 0s - loss: 0.6495 -
accuracy: 0.7785
Epoch 14: val_loss did not improve from 0.96920
1251/1251 [==============================] - 64s 51ms/step - loss:
0.6495 - accuracy: 0.7785 - val_loss: 0.9842 - val_accuracy: 0.6728
Epoch 15/50
1251/1251 [==============================] - ETA: 0s - loss: 0.6091 -
accuracy: 0.7913
Epoch 15: val_loss did not improve from 0.96920
1251/1251 [==============================] - 65s 51ms/step - loss:
0.6091 - accuracy: 0.7913 - val_loss: 1.1129 - val_accuracy: 0.6510
Epoch 16/50
1251/1251 [==============================] - ETA: 0s - loss: 0.5609 -
accuracy: 0.8076
Epoch 16: val_loss did not improve from 0.96920
1251/1251 [==============================] - 64s 51ms/step - loss:
0.5609 - accuracy: 0.8076 - val_loss: 1.0342 - val_accuracy: 0.6717
Epoch 16: early stopping
```

```python
#Loading IF NESCESARY
from tensorflow import keras
model =
keras.models.load_model('/content/drive/MyDrive/ProjetoIA/models/model
S_2_best.h5')
```

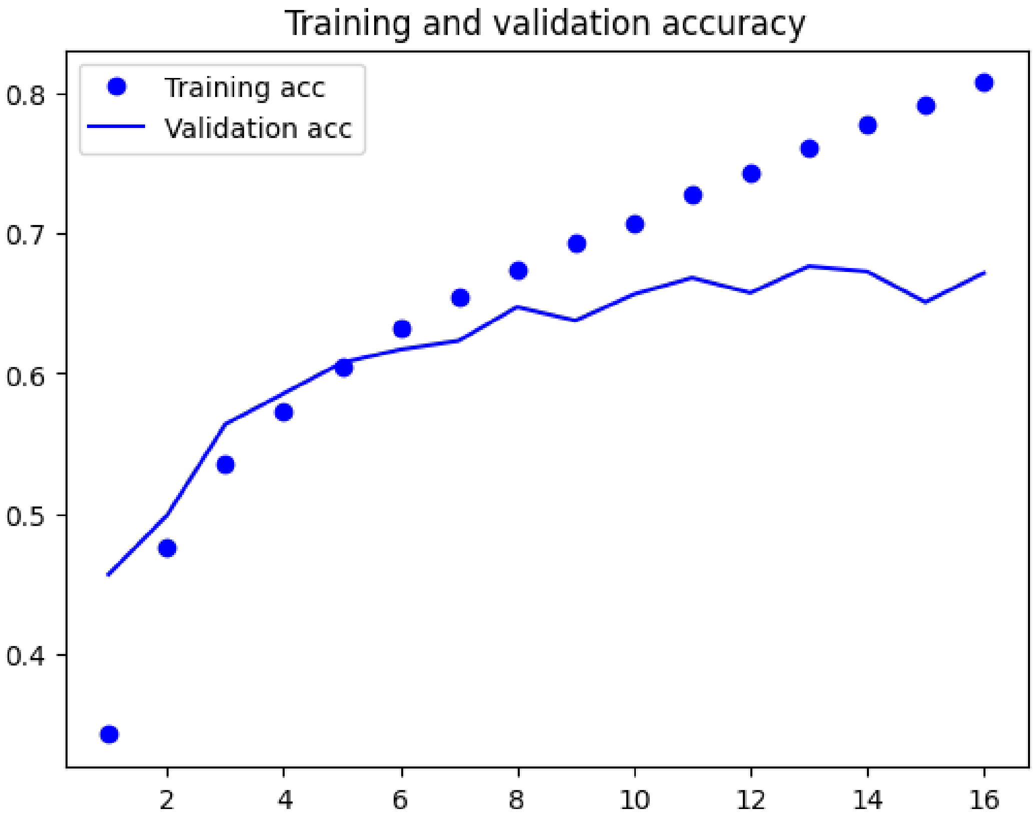
```
313/313 [==============================] - 65s 207ms/step - loss:
1.5914 - accuracy: 0.6758
val_acc: 0.6758000254631042
```

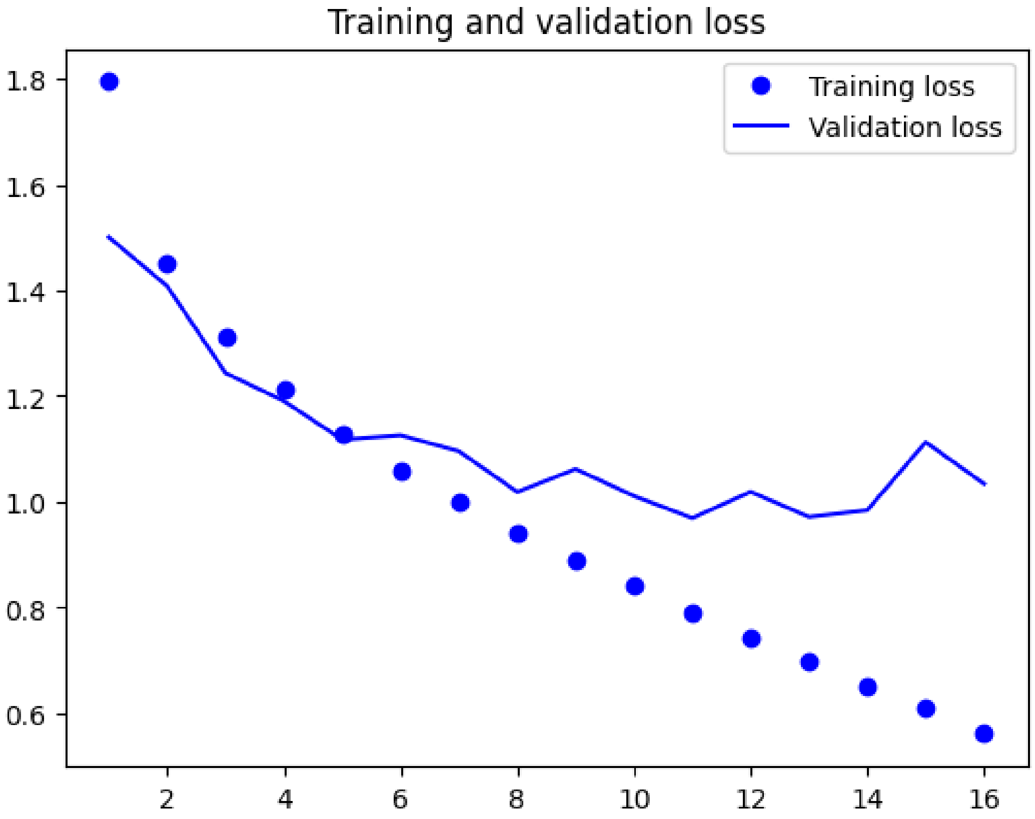
```
#and testing the model
val_loss, val_acc = model.evaluate(validation_dataset)
print('val_acc:', val_acc)
```

```
313/313 [==============================] - 13s 41ms/step - loss:
1.0342 - accuracy: 0.6717
val_acc: 0.6717000007629395
```

```
#Displaying curves of loss and accuracy

import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```


Training and validation accuracy
Training acc
Validation acc
0.8
0.7
0.6
0.5
0.4
2
4
6
8
10
12
14
16


Training and validation loss
Training loss
Validation loss
1.8
1.6
1.4
1.2
1.0
0.8
0.6
2
4
6
8
10
12
14
16