```
#Used to mount google drive to colab so that files can be accessed
from google drive
from google.colab import drive
drive.mount('/content/drive')
```

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

```
#Importing necessary libraries for the project
import tensorflow as tf
from keras.utils import image_dataset_from_directory
from keras.applications.vgg16 import VGG16
from tensorflow import keras
from keras import layers

# Directories
#specifying the directories where the datasets of train, validation
and test are stored

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
)

# 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']
```

```
# Build the model
# Create the base model from the pre-trained model VGG16 with weights
from ImageNet
# excluding the top layers

# Fine-tuning the model
conv_base = VGG16(weights="imagenet", include_top=False)
conv_base.trainable = True #defining that the convolutional base is
trainable
for layer in conv_base.layers[:-2]: #excluding the last two layers
    layer.trainable = False #defining that the layers are not
trainable

# defining the input shape, considering the image redimensioned to
150x150 with 3 channels(RGB)
inputs = keras.Input(shape=(150, 150, 3))
x = keras.applications.vgg16.preprocess_input(inputs)  # Apply input
value scaling
x = conv_base(x)
x = layers.Flatten()(x)
```

```
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(10, activation="softmax")(x)
model = keras.Model(inputs, outputs)

# Compile the model
model.compile(
    loss="sparse_categorical_crossentropy",
    optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
    metrics=["accuracy"]
)

# Adding ModelCheckpoint callback
# its for saving the best model based on the validation loss
# and has a patience of 5 epochs to avoid overfitting
checkpoint_cb = tf.keras.callbacks.ModelCheckpoint(

'/content/drive/MyDrive/ProjetoIA/models/best_modelT_TL_FT_without_DA.
h5',
    save_best_only=True,
    monitor='val_loss',
    mode='min',
    verbose=1
)
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                           patience=5,
                           verbose=1)

# Train the model with fine-tuning and checkpointing
history = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=[checkpoint_cb,early_stop]
)

# Save the final model
model.save('/content/drive/MyDrive/ProjetoIA/models/CNN_modelT_TL_FT_w
ithout_DA.h5')
```

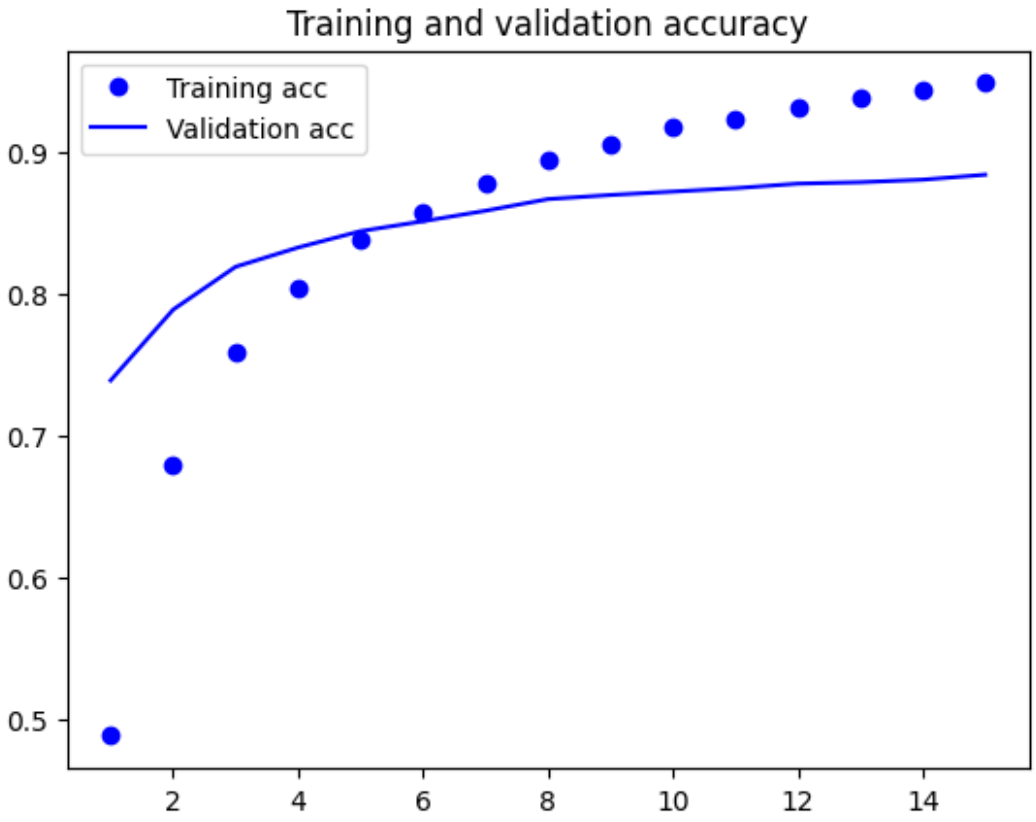
```
Epoch 1/30
   1251/Unknown - 394s 314ms/step - loss: 4.1475 - accuracy: 0.4888
Epoch 1: val_loss improved from inf to 0.93821, saving model to
/content/drive/MyDrive/ProjetoIA/models/best_modelT_TL_FT_without_DA.h
5
1251/1251 [==============================] - 480s 383ms/step - loss:
4.1475 - accuracy: 0.4888 - val_loss: 0.9382 - val_accuracy: 0.7393
Epoch 2/30
1251/1251 [==============================] - ETA: 0s - loss: 1.2252 -
accuracy: 0.6802
```

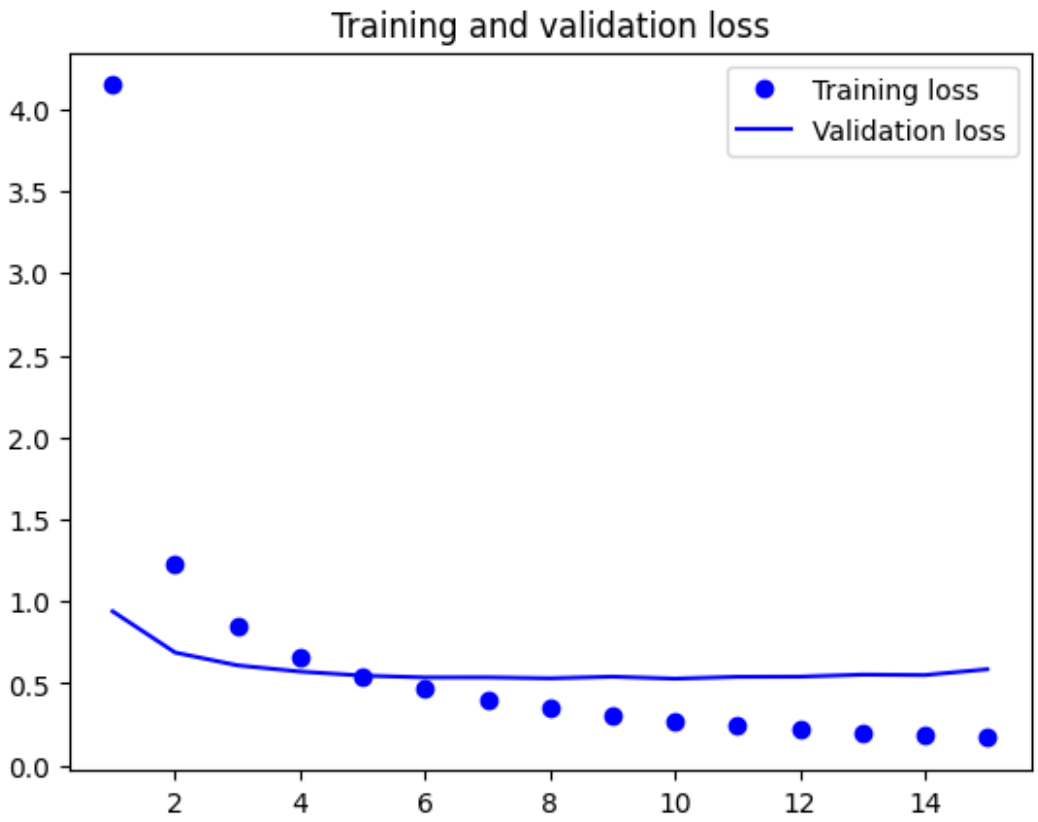
```
Epoch 2: val_loss improved from 0.93821 to 0.68712, saving model to
/content/drive/MyDrive/ProjetoIA/models/best_modelT_TL_FT_without_DA.h
5
1251/1251 [==============================] - 472s 377ms/step - loss:
1.2252 - accuracy: 0.6802 - val_loss: 0.6871 - val_accuracy: 0.7892
Epoch 3/30
1251/1251 [==============================] - ETA: 0s - loss: 0.8453 -
accuracy: 0.7585
Epoch 3: val_loss improved from 0.68712 to 0.60882, saving model to
/content/drive/MyDrive/ProjetoIA/models/best_modelT_TL_FT_without_DA.h
5
1251/1251 [==============================] - 473s 377ms/step - loss:
0.8453 - accuracy: 0.7585 - val_loss: 0.6088 - val_accuracy: 0.8194
Epoch 4/30
1251/1251 [==============================] - ETA: 0s - loss: 0.6609 -
accuracy: 0.8045
Epoch 4: val_loss improved from 0.60882 to 0.57089, saving model to
/content/drive/MyDrive/ProjetoIA/models/best_modelT_TL_FT_without_DA.h
5
1251/1251 [==============================] - 475s 379ms/step - loss:
0.6609 - accuracy: 0.8045 - val_loss: 0.5709 - val_accuracy: 0.8330
Epoch 5/30
1251/1251 [==============================] - ETA: 0s - loss: 0.5401 -
accuracy: 0.8383
Epoch 5: val_loss improved from 0.57089 to 0.54661, saving model to
/content/drive/MyDrive/ProjetoIA/models/best_modelT_TL_FT_without_DA.h
5
1251/1251 [==============================] - 479s 382ms/step - loss:
0.5401 - accuracy: 0.8383 - val_loss: 0.5466 - val_accuracy: 0.8445
Epoch 6/30
1251/1251 [==============================] - ETA: 0s - loss: 0.4627 -
accuracy: 0.8583
Epoch 6: val_loss improved from 0.54661 to 0.53518, saving model to
/content/drive/MyDrive/ProjetoIA/models/best_modelT_TL_FT_without_DA.h
5
1251/1251 [==============================] - 476s 380ms/step - loss:
0.4627 - accuracy: 0.8583 - val_loss: 0.5352 - val_accuracy: 0.8515
Epoch 7/30
1251/1251 [==============================] - ETA: 0s - loss: 0.3982 -
accuracy: 0.8785
Epoch 7: val_loss did not improve from 0.53518
1251/1251 [==============================] - 477s 381ms/step - loss:
0.3982 - accuracy: 0.8785 - val_loss: 0.5356 - val_accuracy: 0.8589
Epoch 8/30
1251/1251 [==============================] - ETA: 0s - loss: 0.3488 -
accuracy: 0.8947
Epoch 8: val_loss improved from 0.53518 to 0.52985, saving model to
/content/drive/MyDrive/ProjetoIA/models/best_modelT_TL_FT_without_DA.h
5
```

```
1251/1251 [==============================] - 470s 376ms/step - loss:
0.3488 - accuracy: 0.8947 - val_loss: 0.5298 - val_accuracy: 0.8671
Epoch 9/30
1251/1251 [==============================] - ETA: 0s - loss: 0.3025 -
accuracy: 0.9061
Epoch 9: val_loss did not improve from 0.52985
1251/1251 [==============================] - 479s 382ms/step - loss:
0.3025 - accuracy: 0.9061 - val_loss: 0.5398 - val_accuracy: 0.8700
Epoch 10/30
1251/1251 [==============================] - ETA: 0s - loss: 0.2721 -
accuracy: 0.9174
Epoch 10: val_loss improved from 0.52985 to 0.52764, saving model
to
/content/drive/MyDrive/ProjetoIA/models/best_modelT_TL_FT_without_DA.h
5
1251/1251 [==============================] - 478s 382ms/step - loss:
0.2721 - accuracy: 0.9174 - val_loss: 0.5276 - val_accuracy: 0.8724
Epoch 11/30
1251/1251 [==============================] - ETA: 0s - loss: 0.2460 -
accuracy: 0.9234
Epoch 11: val_loss did not improve from 0.52764
1251/1251 [==============================] - 471s 376ms/step - loss:
0.2460 - accuracy: 0.9234 - val_loss: 0.5390 - val_accuracy: 0.8748
Epoch 12/30
1251/1251 [==============================] - ETA: 0s - loss: 0.2209 -
accuracy: 0.9316
Epoch 12: val_loss did not improve from 0.52764
1251/1251 [==============================] - 478s 382ms/step - loss:
0.2209 - accuracy: 0.9316 - val_loss: 0.5399 - val_accuracy: 0.8780
Epoch 13/30
1251/1251 [==============================] - ETA: 0s - loss: 0.1978 -
accuracy: 0.9383
Epoch 13: val_loss did not improve from 0.52764
1251/1251 [==============================] - 475s 379ms/step - loss:
0.1978 - accuracy: 0.9383 - val_loss: 0.5523 - val_accuracy: 0.8790
Epoch 14/30
1251/1251 [==============================] - ETA: 0s - loss: 0.1816 -
accuracy: 0.9439
Epoch 14: val_loss did not improve from 0.52764
1251/1251 [==============================] - 478s 382ms/step - loss:
0.1816 - accuracy: 0.9439 - val_loss: 0.5497 - val_accuracy: 0.8808
Epoch 15/30
1251/1251 [==============================] - ETA: 0s - loss: 0.1667 -
accuracy: 0.9488
Epoch 15: val_loss did not improve from 0.52764
1251/1251 [==============================] - 478s 382ms/step - loss:
0.1667 - accuracy: 0.9488 - val_loss: 0.5850 - val_accuracy: 0.8843
Epoch 15: early stopping
```

```
#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()
```






```
#Assessing the model's performance
test_loss, test_acc = model.evaluate(validation_dataset)#devolve um
tuplo, sao objetos individuais
print('test_acc:', test_acc)

313/313 [==============================] - 88s 281ms/step - loss:
0.5850 - accuracy: 0.8843
test_acc: 0.8842999935150146
```

Foram avaliados 313 batches no conjunto de validação Perda calculada no conjunto de validação 0.5850 Previsão do conjunto de validação com 88.42% das previsões foram corretas.