Librerias

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import os
from keras.src.engine.data_adapter import train_validation_split
```

Procesamiento de los datos

```
BATCH_SIZE = 32
IMG_SIZE = (160,160)
PATH = '/content/drive/MyDrive/Images/computer_parts'

train_dir = os.path.join(PATH, 'train')

train_dataset = tf.keras.utils.image_dataset_from_directory(train_dir,
                                                            shuffle = True,
                                                            batch_size = BATCH_SIZE,
                                                            image_size = IMG_SIZE)

test_dir = os.path.join(PATH, 'test')

test_dataset = tf.keras.utils.image_dataset_from_directory(train_dir,
                                                            shuffle = True,
                                                            batch_size = BATCH_SIZE,
                                                            image_size = IMG_SIZE)
```

```
Found 853 files belonging to 3 classes.
Found 853 files belonging to 3 classes.
```

```
train_dataset.take(1)
```

```
<_TakeDataset element_spec=(TensorSpec(shape=(None, 160, 160, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.int32, name=None))>
```

```
class_name = train_dataset.class_names

plt.figure(figsize = (10,10))
for image, label in train_dataset.take(1):
  for i in range(9):
    ax=plt.subplot(3,3,i+1)
    plt.imshow(image[i].numpy().astype("uint8"))
    plt.title(class_name[label[i]])
    plt.axis("off")
```


```
val_batches = tf.data.experimental.cardinality(train_dataset)
test_batches = tf.data.experimental.cardinality(test_dataset)
validation_dataset = train_dataset.skip(val_batches // 5)

print('Numero de batches para validation_dataset = %d' % tf.data.experimental.cardinality(validation_dataset))
print('Numero de batches para test_dataset = %d' % test_batches)
```

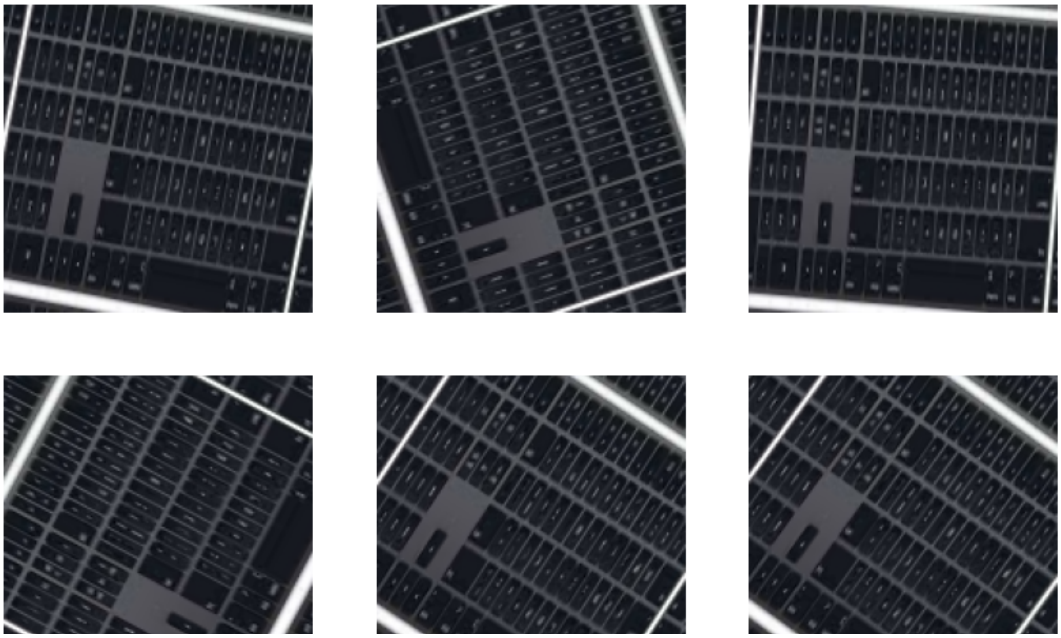
```
Numero de batches para validation_dataset = 22
Numero de batches para test_dataset = 27
```

```
AUTOTUNE = tf.data.AUTOTUNE

test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
```

```
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip('horizontal'),
    tf.keras.layers.RandomRotation(0.2)
])
```

```
for image, _ in train_dataset.take(1):
  plt.figure(figsize=(10,10))
  first_image = image[0]
  for i in range(9):
    ax=plt.subplot(3,3,i+1)
    augmented_image = data_augmentation(tf.expand_dims(first_image,0))
    plt.imshow(augmented_image[0] / 255)
    plt.axis("off")
```



```
rescale = tf.keras.layers.Rescaling(1./127.5, offset=-1)
preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
```

Cargar la red pre-entrenada DenseNet121

```
IMG_SHAPE = IMG_SIZE + (3,)
print(IMG_SHAPE)
base_model = tf.keras.applications.DenseNet121(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

```
(160, 160, 3)
```

```
image_batch, label_batch = next(iter(train_dataset))
feature_batch = base_model(image_batch)

print(feature_batch.shape)
```

```
(32, 5, 5, 1024)
```

```
base_model.trainable = False
base_model.summary()
```

```
 atenate)                                                        conv5_block15_2_conv[0][0] ]

 conv5_block16_0_bn (BatchN  (None, 5, 5, 992)           3968      ['conv5_block15_concat[0][0]']
 ormalization)

 conv5_block16_0_relu (Acti  (None, 5, 5, 992)           0         ['conv5_block16_0_bn[0][0]']
 vation)

 conv5_block16_1_conv (Conv  (None, 5, 5, 128)           126976    ['conv5_block16_0_relu[0][0]']
 2D)

 conv5_block16_1_bn (BatchN  (None, 5, 5, 128)           512       ['conv5_block16_1_conv[0][0]']
 ormalization)

 conv5_block16_1_relu (Acti  (None, 5, 5, 128)           0         ['conv5_block16_1_bn[0][0]']
 vation)

 conv5_block16_2_conv (Conv  (None, 5, 5, 32)            36864     ['conv5_block16_1_relu[0][0]']
 2D)

 conv5_block16_concat (Conc  (None, 5, 5, 1024)          0         ['conv5_block15_concat[0][0]',
 atenate)                                                         'conv5_block16_2_conv[0][0]']

 bn (BatchNormalization)     (None, 5, 5, 1024)          4096      ['conv5_block16_concat[0][0]']

 relu (Activation)           (None, 5, 5, 1024)          0         ['bn[0][0]']

==================================================================================================
Total params: 7037504 (26.85 MB)
Trainable params: 0 (0.00 Byte)
Non-trainable params: 7037504 (26.85 MB)
__________________________________________________________________________________________________
```

```
1 global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
2 feature_batch_average = global_average_layer(feature_batch)
3 print(feature_batch_average.shape)
```

```
(32, 1024)
```

```
1 prediction_layer = tf.keras.layers.Dense(3)
2 prediction_batch = prediction_layer(feature_batch_average)
3 print(prediction_batch.shape)
```

```
(32, 3)
```

Unir modelo

```
1 inputs = tf.keras.Input(shape=(160,160,3))
2 x = data_augmentation(inputs)
3 x = preprocess_input(x)
4 x = base_model(x, training=False)
5 x = global_average_layer(x)
6 x = tf.keras.layers.Dropout(0.2)(x)
7 outputs = prediction_layer(x)
8 model = tf.keras.Model(inputs, outputs)
9
```

```
1 base_learning_rate = 0.0001
2 model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
3               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
4               metrics=['accuracy'])
5 model.summary()
```

```
Model: "model_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_16 (InputLayer)       [(None, 160, 160, 3)]     0

 sequential_7 (Sequential)   (None, 160, 160, 3)       0

 tf.math.truediv_7 (TFOpLam  (None, 160, 160, 3)       0
 bda)

 tf.math.subtract_7 (TFOpLa  (None, 160, 160, 3)       0
 mbda)
```

```
densenet121 (Functional)    (None, 5, 5, 1024)        7037504

global_average_pooling2d_7  (None, 1024)              0
 (GlobalAveragePooling2D)

dropout_7 (Dropout)         (None, 1024)              0

dense_7 (Dense)             (None, 3)                 3075

=================================================================
Total params: 7040579 (26.86 MB)
Trainable params: 3075 (12.01 KB)
Non-trainable params: 7037504 (26.85 MB)
_________________________________________________________________
```

```
initial_epochs = 10

loss0, accuracy0 = model.evaluate(validation_dataset)

print(loss0)
print(accuracy0)
```

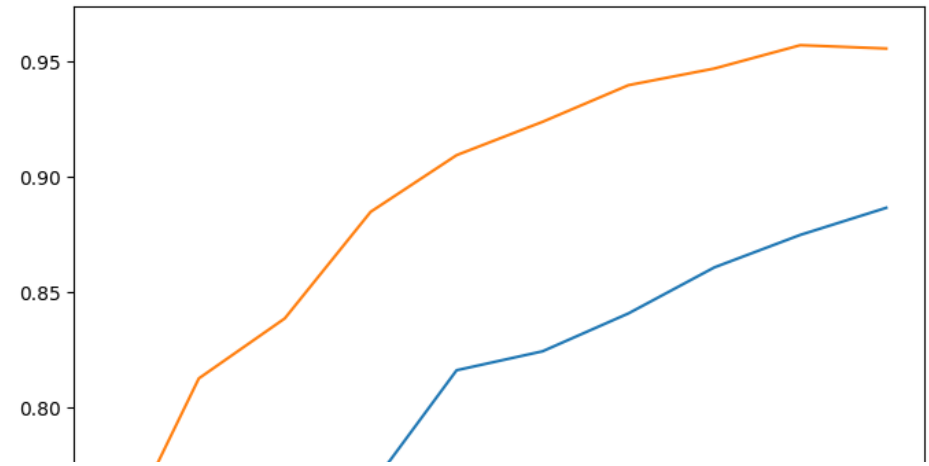
```
22/22 [==============================] - 11s 187ms/step - loss: 0.9614 - accuracy: 0.6537
0.9613553881645203
0.6536796689033508
```

```
history = model.fit(train_dataset,
                    epochs=initial_epochs,
                    validation_data = validation_dataset)
```

```
Epoch 1/10
27/27 [==============================] - 17s 291ms/step - loss: 0.9987 - accuracy: 0.6237 - val_loss: 0.6455 - val_accuracy: 0.7359
Epoch 2/10
27/27 [==============================] - 10s 318ms/step - loss: 0.8429 - accuracy: 0.6542 - val_loss: 0.5013 - val_accuracy: 0.8124
Epoch 3/10
27/27 [==============================] - 10s 331ms/step - loss: 0.6840 - accuracy: 0.7140 - val_loss: 0.4281 - val_accuracy: 0.8384
Epoch 4/10
27/27 [==============================] - 9s 296ms/step - loss: 0.5972 - accuracy: 0.7655 - val_loss: 0.3529 - val_accuracy: 0.8846
Epoch 5/10
27/27 [==============================] - 10s 302ms/step - loss: 0.4846 - accuracy: 0.8159 - val_loss: 0.3126 - val_accuracy: 0.9091
Epoch 6/10
27/27 [==============================] - 10s 332ms/step - loss: 0.4618 - accuracy: 0.8242 - val_loss: 0.2804 - val_accuracy: 0.9235
Epoch 7/10
27/27 [==============================] - 10s 320ms/step - loss: 0.4391 - accuracy: 0.8406 - val_loss: 0.2438 - val_accuracy: 0.9394
Epoch 8/10
27/27 [==============================] - 9s 301ms/step - loss: 0.4014 - accuracy: 0.8605 - val_loss: 0.2164 - val_accuracy: 0.9466
Epoch 9/10
27/27 [==============================] - 11s 344ms/step - loss: 0.3416 - accuracy: 0.8746 - val_loss: 0.1954 - val_accuracy: 0.9567
Epoch 10/10
27/27 [==============================] - 10s 348ms/step - loss: 0.3214 - accuracy: 0.8863 - val_loss: 0.1787 - val_accuracy: 0.9553
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.figure(figsize=(8,8))
plt.plot(acc, label = 'Training acc')
plt.plot(val_acc, label = 'Validation acc')
```

```
[<matplotlib.lines.Line2D at 0x7df24e152e60>]
```



Fine tunning

```
1 base_model.trainable = True
2
3 print('Numero de capas ', len(base_model.layers))
```

```
Numero de capas  427
```

```
1 fine_tune_at=100
2 for layer in base_model.layers[:fine_tune_at]:
3   layer.trainable = False
```

```
1 model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=base_learning_rate),
2               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
3               metrics=['accuracy'])
4 model.summary()
```

```
Model: "model_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_16 (InputLayer)       [(None, 160, 160, 3)]     0

 sequential_7 (Sequential)   (None, 160, 160, 3)       0

 tf.math.truediv_7 (TFOpLam  (None, 160, 160, 3)       0
 bda)

 tf.math.subtract_7 (TFOpLa  (None, 160, 160, 3)       0
 mbda)

 densenet121 (Functional)    (None, 5, 5, 1024)        7037504

 global_average_pooling2d_7  (None, 1024)              0
  (GlobalAveragePooling2D)

 dropout_7 (Dropout)         (None, 1024)              0

 dense_7 (Dense)             (None, 3)                 3075

=================================================================
Total params: 7040579 (26.86 MB)
Trainable params: 6152259 (23.47 MB)
Non-trainable params: 888320 (3.39 MB)
_________________________________________________________________
```

```
1 fine_tune_epochs = 10
2 total_epochs = initial_epochs + fine_tune_epochs
3 history_fine = model.fit (train_dataset,
4                           epochs = total_epochs,
5                           initial_epoch = history.epoch[-1],
6                           validation_data=validation_dataset)
```

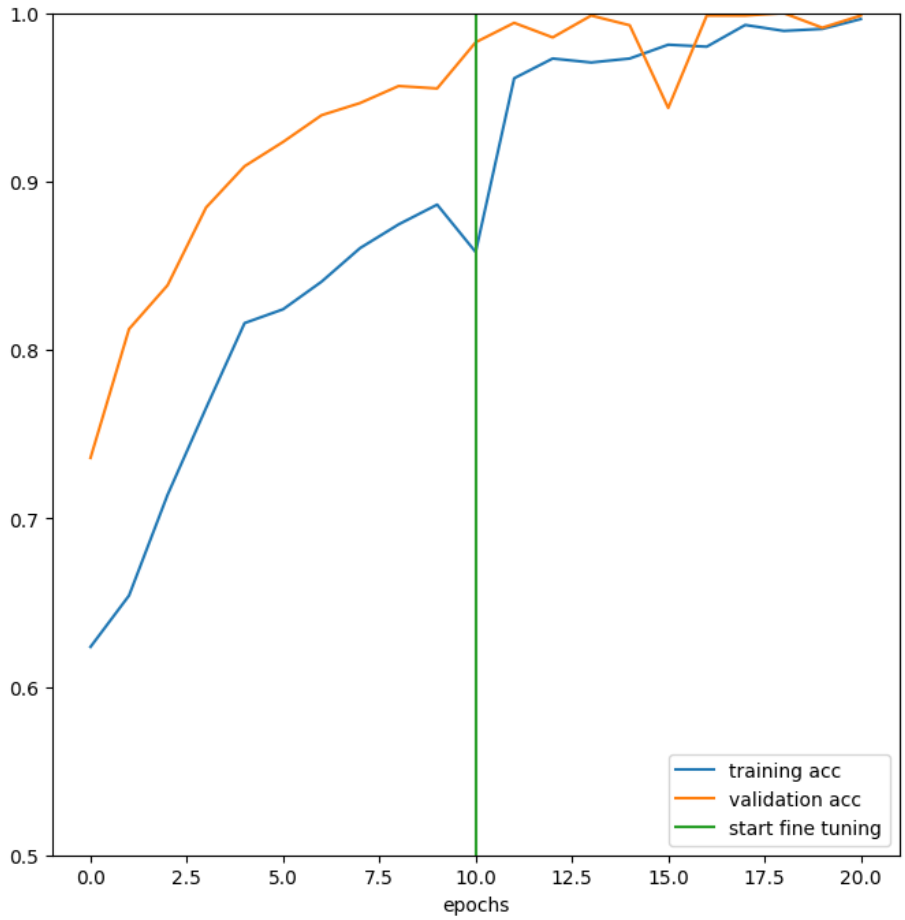
```
Epoch 10/20
27/27 [==============================] - 55s 511ms/step - loss: 0.5351 - accuracy: 0.8581 - val_loss: 0.0598 - val_accuracy: 0.9827
```

```
Epoch 11/20
27/27 [==============================] - 11s 374ms/step - loss: 0.1131 - accuracy: 0.9613 - val_loss: 0.0222 - val_accuracy: 0.9942
Epoch 12/20
27/27 [==============================] - 10s 347ms/step - loss: 0.0662 - accuracy: 0.9730 - val_loss: 0.0385 - val_accuracy: 0.9856
Epoch 13/20
27/27 [==============================] - 11s 367ms/step - loss: 0.0916 - accuracy: 0.9707 - val_loss: 0.0054 - val_accuracy: 0.9986
Epoch 14/20
27/27 [==============================] - 11s 375ms/step - loss: 0.1060 - accuracy: 0.9730 - val_loss: 0.0230 - val_accuracy: 0.9928
Epoch 15/20
27/27 [==============================] - 11s 348ms/step - loss: 0.0595 - accuracy: 0.9812 - val_loss: 0.1460 - val_accuracy: 0.9437
Epoch 16/20
27/27 [==============================] - 11s 361ms/step - loss: 0.0889 - accuracy: 0.9801 - val_loss: 0.0089 - val_accuracy: 0.9986
Epoch 17/20
27/27 [==============================] - 11s 368ms/step - loss: 0.0225 - accuracy: 0.9930 - val_loss: 0.0045 - val_accuracy: 0.9986
Epoch 18/20
27/27 [==============================] - 11s 359ms/step - loss: 0.0264 - accuracy: 0.9894 - val_loss: 9.5069e-04 - val_accuracy: 1.
Epoch 19/20
27/27 [==============================] - 10s 324ms/step - loss: 0.0315 - accuracy: 0.9906 - val_loss: 0.0295 - val_accuracy: 0.9913
Epoch 20/20
27/27 [==============================] - 11s 374ms/step - loss: 0.0111 - accuracy: 0.9965 - val_loss: 0.0045 - val_accuracy: 0.9986
```

```
acc += history_fine.history['accuracy']
val_acc += history_fine.history['val_accuracy']

plt.figure(figsize=(8,8))
plt.plot(acc, label = 'training acc')
plt.plot(val_acc, label = 'validation acc')
plt.ylim([0.5, 1])
plt.plot([initial_epochs, initial_epochs],
         plt.ylim(), label='start fine tuning')

plt.legend(loc='lower right')
plt.xlabel('epochs')
plt.show()
```




```
model.save('model.h5')
```

```
predictions = model.predict(test_dataset)

# class_name = train_dataset.class_names

plt.figure(figsize = (10,10))
for image, label in train_dataset.take(1):
  for i in range(9):
    ax=plt.subplot(3,3,i+1)
    plt.imshow(image[i].numpy().astype("uint8"))
    plt.title(class_name[np.argmax(predictions[i])])
    plt.axis("off")
```

```
27/27 [==============================] - 6s 176ms/step
```


```
1
```