
# TensorFlow

TensorFlow es un software de computación numérica, creado por Google, orientado a problemas de Deep Learning. Deep Learning es un área específica de Machine Learning que está tomando gran relevancia dentro del mundo de la Inteligencia Artificial y que está detrás de algunos de las novedades tecnológicas más sorprendentes de los últimos años.

El origen de TensorFlow está en años de experiencia de Google en el campo de la Inteligencia Artificial. TensorFlow nace del trabajo de Google Brain, un grupo de investigadores e ingenieros de Google dedicado a investigar en el área de la IA, que desarrollaron en 2011 DistBelief, el predecesor cerrado de TensorFlow.

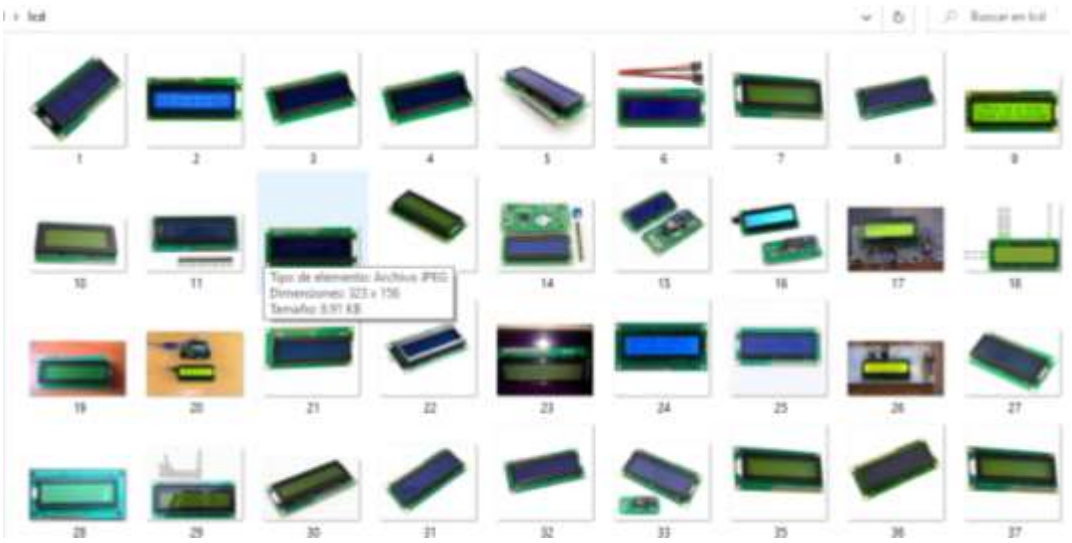
TensorFlow está implementado en C++ y Python, y la forma más conveniente y sencilla de utilizarlo es a través del API

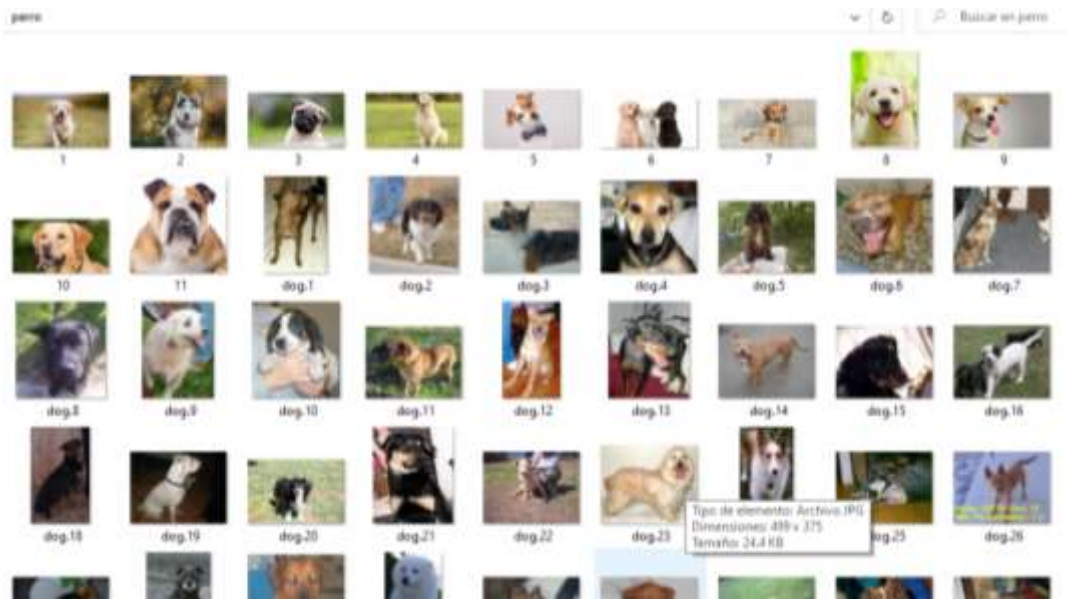
# Clasificador de imágenes

Creamos una carpeta con el nombre lcd y otra con el nombre perro.

| | | |
|---|---|---|
| lcd | 03/02/2021 11:45 a. m. | Carpeta de archivos |
| perro | 07/02/2021 03:27 p. m. | Carpeta de archivos |

Dentro de dichas carpetas estarán las imágenes que se utilizarán para el entrenamiento.





TensorFlow

## Código

prueba2.py - C:/Users/kv460/Documents/lcd/pr

File Edit Format Run Options Window

```
###importamos las  librerias
import cv2
import numpy as np
import os
import matplotlib.pyplot as plt
```

Reconocimiento de Imágenes

Para poder usar directorios

```
####cargamos las fotos  de las carpetas especificas###
lcd_folder_path="C:/Users/kv460/Documents/lcd/lcd"
lcd=[]
img_size=150
for img in os.listdir(lcd_folder_path):
    img = cv2.imread(os.path.join(lcd_folder_path,img))
    #img_gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img_resize= cv2.resize(img,(img_size,img_size))
    lcd.append(img_resize)
```

Carpeta lcd

Lista vacía, definimos el tamaño de la imagen

Leemos imágenes

Leemos las imágenes, las volvemos del mismo tamaño.

```
#vizualizamos tamaño de la imagen
lcd = np.array(lcd)
print(lcd.shape)
```

Se muestran 41 imágenes de 150*150 con 3 canales es decir RGB

Python 3.8.5 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/kv460/Documents/lcd/prueba2.py ==========
(41, 150, 150, 3)
```

Se realiza exactamente lo mismo, pero ahora para la carpeta de perro

```
####cargamos las fotos  de las carpetas especificas###
perro_folder_path="C:/Users/kv460/Documents/lcd/perro"
perro=[]
img_size=150
for img in os.listdir(perro_folder_path):
    img = cv2.imread(os.path.join(perro_folder_path,img))
    #img_gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img_resize= cv2.resize(img,(img_size,img_size))
    perro.append(img_resize)

perro = np.array(perro)
print(perro.shape)
```

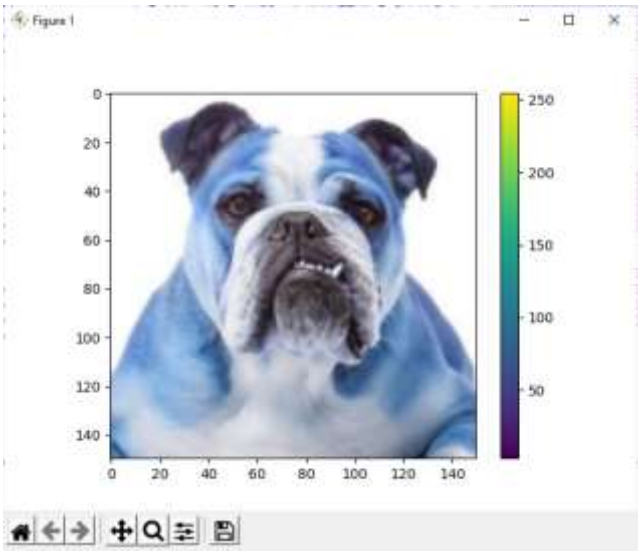
Se muestran 81 imágenes de 150*150 con 3 canales es decir RGB

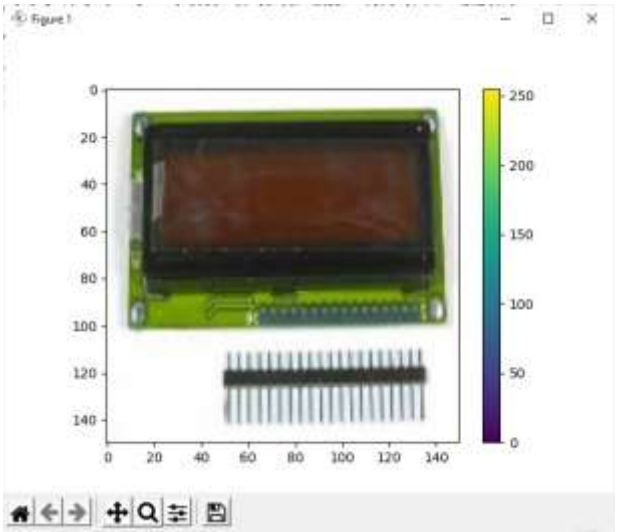
```
(81, 150, 150, 3)
```

```
#checamos el tamaño de las imagenes
print(lcd[4].shape)
plt.figure()
plt.imshow(np.squeeze(lcd[2]))
plt.colorbar()
plt.grid(False)
plt.show()
#############checamos el tamaño de las imagenes############
print(perro[2].shape)
plt.figure()
plt.imshow(np.squeeze(perro[2]))
plt.colorbar()
plt.grid(False)
plt.show()
```

Se grafica la imagen #4 para revisar las imágenes.

Se grafica la imagen #2 para revisar las imágenes.




Se juntarán las imágenes con una concatenación

```
##concatenamos imagenes#########
images=np.concatenate([lcd,perro])
#convertir a un arreglo
Images=np.array(images)
print(len(images)) #imprimecuantas imagenes hay en total
```

```
122
```

En este caso hay 122 imágenes

En este momento etiquetas todas nuestras imágenes teniendo en cuenta cuantas imágenes hay en cada carpeta en este caso en lcd hay 41 y en la carpeta perro hay

```
############################etiquetamos#################
etiquetas_lcd=np.repeat(0,41)#lcd se reconoce con elnumero 0
print(len(etiquetas_lcd))
print(etiquetas_lcd)
################etiquetamos##############
etiquetas_perro=np.repeat(1,81)#perro se reconoce con elnumero 1
print(len(etiquetas_perro))
print(etiquetas_perro)
```

Aquí podemos ver las etiquetas representadas en las siguientes listas

```
41
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
81
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1]
```

Ahora damos el nombre a nuestras clases teniendo en cuenta que lcd fue nuestra primera etiqueta y perro la segunda, por lo cual deben ser colocadas en ese orden.

Posterior a esto concatenamos las etiquetas

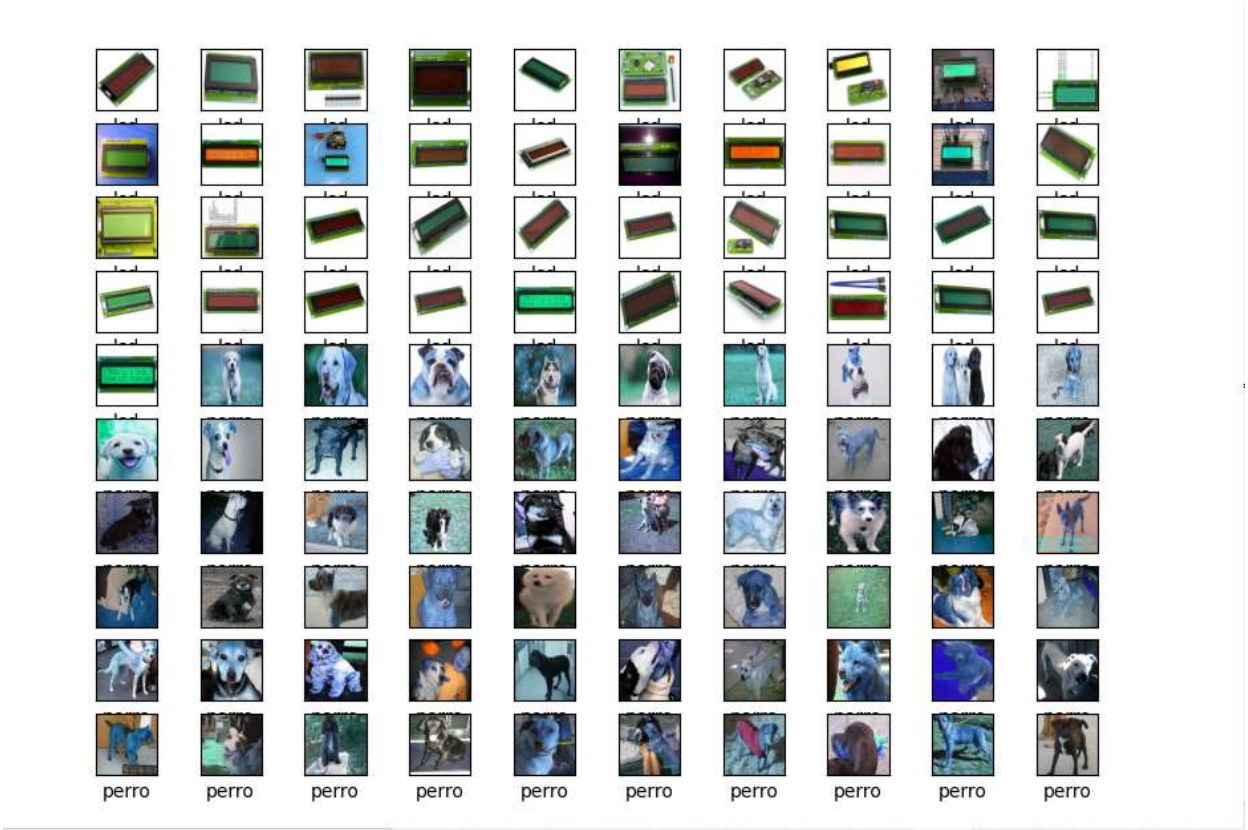
```
#nombre de las clases empezando con cero
class_names=['lcd','perro']
#concatenamos clases
labels=np.concatenate([etiquetas_lcd,etiquetas_perro])
print(len(labels))
print(labels)
```

Notamos que tenemos 122 etiquetas cada una con la clase correspondiente

```
122
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1]
(122,)
```

Graficamos las primeras 100 imágenes

```
print(Labels.shape)
plt.figure(figsize=(10,10))
for i in range(100):
    plt.subplot(10,10,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(Images[i])
    #, cmap=plt.cm.binary
    plt.xlabel(class_names[Labels[i]])
plt.show()
```



Importamos librerías de Tensorflow

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models
import tensorflow.keras.optimizers as Optimizer
```

Se realiza el entrenamiento con 30 iteraciones

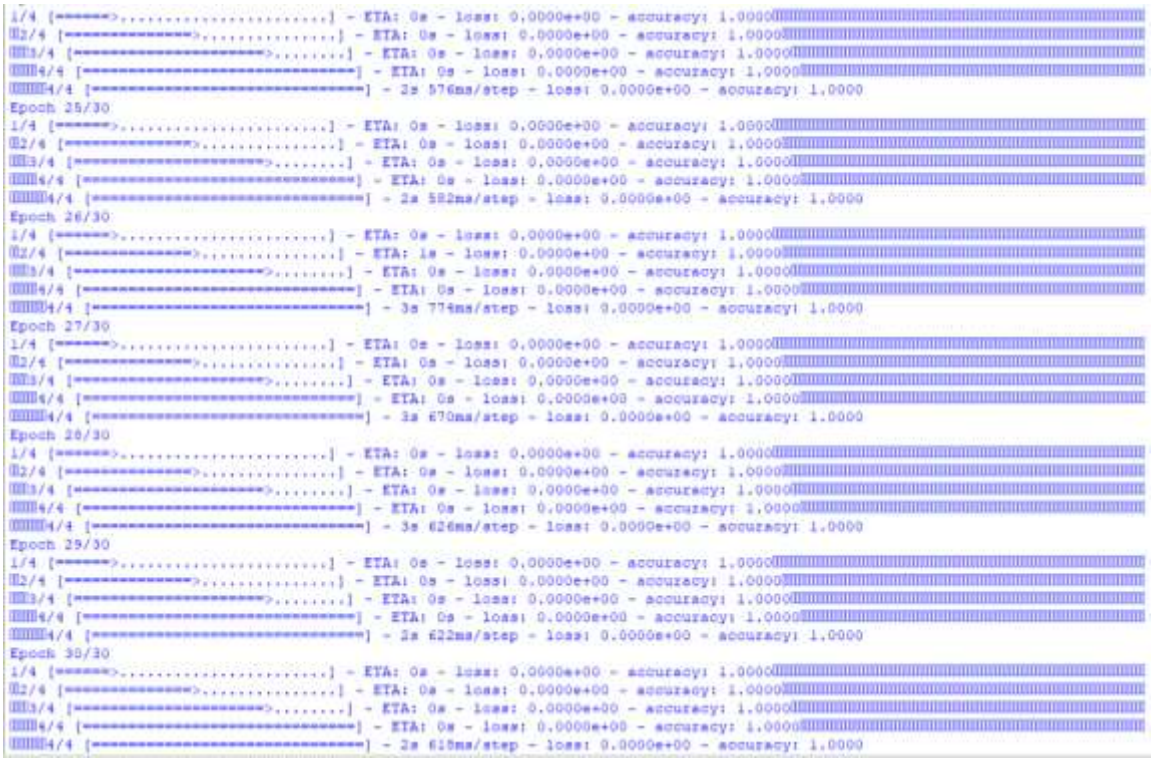
```
#REALIZAMOS ELENTRENAMIENTO
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(150, 150,3)),
    keras.layers.Dense(128, activation='relu'),

    keras.layers.Dense(2, activation='softmax'),

])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(Images, Labels, epochs=3)
trained=model.fit(Images, Labels, epochs=30)
```

Se mete una imagen de 150*150 y 3 canales.

Forma de medir.



Para comprobar que quedo adecuadamente cargamos una imagen y la visualizamos.

```
#imagen de interner se debe cargar
img=cv2.imread('h.jpeg')
img_cvt=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.show
```


Se hace un cambio de tamaño en la imagen para que quede de 150*150 con 3 canales

```
img2=img_cvt
img2=cv2.resize(img2,(img_size,img_size))
print(img2.shape)
img2=(np.expand_dims(img2,0))
print(img2.shape)
```

Se realiza la predicción para saber cómo la clasifica si como una lcd o un perro

```
predictions_single=model.predict(img2)
print(predictions_single)
print(np.sum(predictions_single))
print(np.argmax(predictions_single))
print(class_names[np.argmax(predictions_single)])
```

Hace una sola predicción en base a img2 que es la imagen cargada.

Se toma el número máximo donde se ubique la imagen ya que esto relaciona la clase.

Aquí predice si es una lcd o un tigre en base al índice más grande.

```
(150, 150, 3)
(1, 150, 150, 3)
[[1. 0.]]
1.0
0
lcd
```

Menciona si fue uno o cero

Marca que el valor es cero es decir una lcd

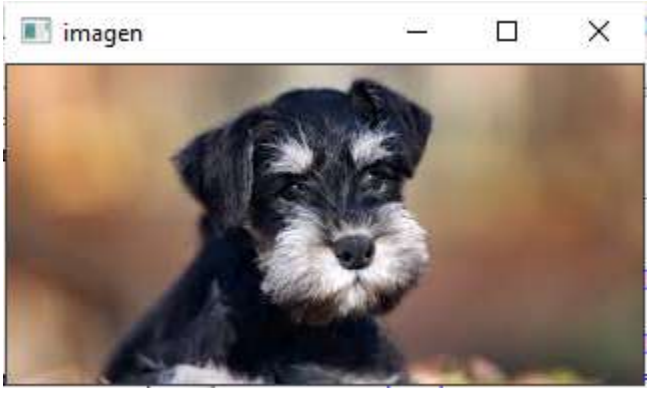
Para comprobar que quedo adecuadamente cargamos una imagen y la visualizamos, pero en esta ocasión será de un perro para ver que realmente funciona.

```
#imagen de interner se debe cargar
img=cv2.imread('j.jpg')
img_cvt=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
cv2.imshow('imagen',img)

img2=img_cvt
img2=cv2.resize(img2,(img_size,img_size))
print(img2.shape)
img2=(np.expand_dims(img2,0))
print(img2.shape)

predictions_single=model.predict(img2)
print(predictions_single)
print(np.sum(predictions_single))
print(np.argmax(predictions_single))
print(class_names[np.argmax(predictions_single)])
```

Podemos notar que pudo clasificar la imagen como un perro.




```
(150, 150, 3)
(1, 150, 150, 3)
[[0. 1.]]
1.0
1
perro
```