UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
SECCIONAL TUNJA
VIGILADA MINEDUCACIÓN - SNIES 1732

REDES NEURONALES

REDES NEURONALES

Ejercicio:

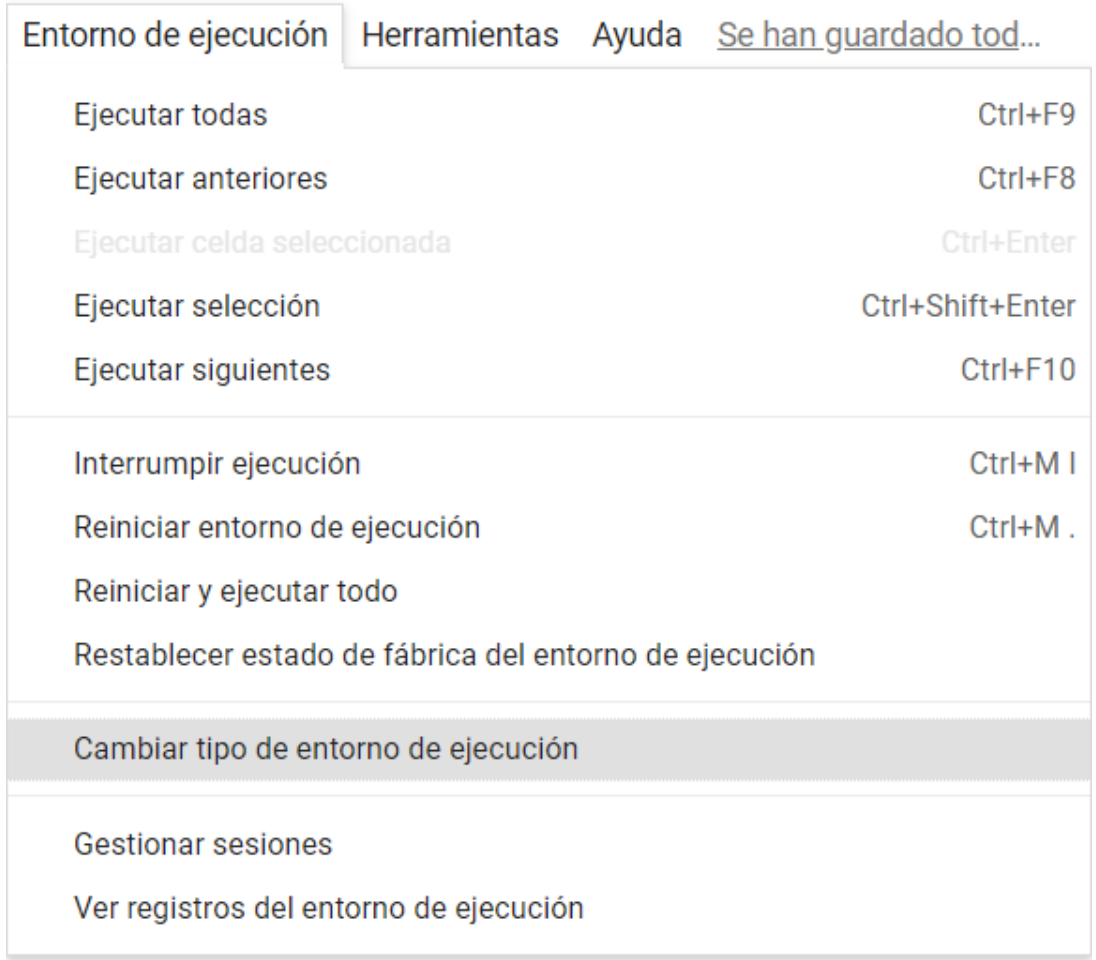
Realizar un programa que utilice una red neuronal para entrenar con el siguiente conjunto de datos

X	Y
1	-1,65
2	1,2
3	4,05
4	6,9
5	9,75
6	12,6
7	15,45
8	18,3
9	21,15
10	24

Utilice la red para comprobar que la predicción de una entrada de x=20 es aproximadamente igual a 52,5

Clasificador de perros y gatos

REDES NEURONALES



Para que los modelos se entrenen más rápido

¡Siempre hacia lo alto!

REDES NEURONALES

1. Importar las librerías y cargar un conjunto de datos de imágenes preexistente

```
import tensorflow as tf
import tensorflow_datasets as tfds

#descargar el dataset
datos,metadatos = tfds.load('cats_vs_dogs', as_supervised=True, with_info=True)
```

Esto tardará un tiempo, debido a que tendrá que leer muchas imágenes. La aplicación mostrará una barra de progreso.

REDES NEURONALES

```
Downloading and preparing dataset cats_vs_dogs/4.0.0 (download: 786.68 MiB, generated: Unknown size, total: 786.68 MiB) to /ro
Dl Completed...: 100% 1/1 [00:16<00:00, 16.72s/ url]
Dl Size...: 100% 786/786 [00:16<00:00, 45.60 MiB/s]

WARNING:absl:1738 images were corrupted and were skipped
Shuffling and writing examples to /root/tensorflow_datasets/cats_vs_dogs/4.0.0.incompleteJD5MC3/cats_vs_dogs-train.tfrecord
100% 23261/23262 [00:02<00:00, 7973.05 examples/s]
Dataset cats_vs_dogs downloaded and prepared to /root/tensorflow_datasets/cats_vs_dogs/4.0.0. Subsequent calls will reuse this
```

REDES NEURONALES

Imprimir metadatos para conocer el dataset

```
metadatos
```

```
tfds.core.DatasetInfo(
    name='cats_vs_dogs',
    version=4.0.0,
    description='A large set of images of cats and dogs.There are 1738 corrupted images that are dropped.',
    homepage='https://www.microsoft.com/en-us/download/details.aspx?id=54765',
    features=FeaturesDict({
        'image': Image(shape=(None, None, 3), dtype=tf.uint8),
        'image/filename': Text(shape=(), dtype=tf.string),
        'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=2),
    }),
    total_num_examples=23262,
    splits={
        'train': 23262,
    },
    supervised_keys=('image', 'label'),
    citation="""@Inproceedings (Conference){asirra-a-captcha-that-exploits-interest-aligned-manual-image-categorization,
    author = {Elson, Jeremy and Douceur, John (JD) and Howell, Jon and Saul, Jared},
    title = {Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization},
```

REDES NEURONALES

Mostrar algunos datos

```
#Muestra 5 datos con su respectiva etiqueta
tfds.as_dataframe(datos['train'].take(5), metadatos)
```

	image	label
0		1 (dog)
1		1 (dog)
2		1 (dog)
3		0 (cat)
4		1 (dog)

¡Siempre hacia lo alto!

REDES NEURONALES

Mostrar los datos en forma de matriz o tabla

```
import matplotlib.pyplot as plt
plt.figure(figsize=(15,15))
for i, (imagen, etiqueta) in enumerate(datos['train'].take(25)):
  plt.subplot(5,5,i+1) #5 filas y 5 columnas
  plt.xticks([])
  plt.yticks([]) #para que no salgan las reglas en los ejes
  plt.imshow(imagen)
```

Se observa que algunas imágenes están de forma vertical y otras horizontal, además tienen diferentes tamaños.

REDES NEURONALES

Se igualan los tamaños de las imágenes para mayor control y efectividad

```
import matplotlib.pyplot as plt
import cv2 #para igualar tamaño de las imágenes
plt.figure(figsize=(15,15))
tamano=200  #se define el tamaño de la imagen
for i, (imagen, etiqueta) in enumerate(datos['train'].take(25)):
  #aplica el cambio de tamaño
  imagen=cv2.resize(imagen.numpy(), (tamano, tamano))
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(imagen)
```





REDES NEURONALES

Se cambian las imágenes a blanco y negro para minimizar tiempo de procesamiento

```
tamano=200  #se define el tamaño de la imagen
for i, (imagen, etiqueta) in enumerate(datos['train'].take(25)):
  #aplica el cambio de tamaño
  imagen=cv2.resize(imagen.numpy(), (tamano, tamano))
  imagen=cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(imagen,cmap='gray')
```





REDES NEURONALES

Entre más grande la imagen también más tiempo de procesamiento. Se debe buscar un tamaño pequeño pero que no pierda tanto detalle. Pruebe cambiando el tamaño y escoja uno

REDES NEURONALES

Entre más grande la imagen también más tiempo de procesamiento. Se debe buscar un tamaño pequeño pero que no pierda tanto detalle. Pruebe cambiando el tamaño y escoja uno

```
tamano=100  #se define el tamaño de la imagen
```





REDES NEURONALES

Se crea el conjunto de datos de entrenamiento con los ajustes de las imágenes (tamaño y color)

```
traindata=[]
for i, (imagen, etiqueta) in enumerate(datos['train']):   #toma todos los datos
  imagen=cv2.resize(imagen.numpy(), (tamano, tamano))
  imagen=cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
  imagen=imagen.reshape(tamano, tamano,1)  #para que sepa que solo toma el canal de color de grises
  traindata.append([imagen,etiqueta])
```

```
len(traindata)
```

```
23262
```

Verificamos que se hayan cargado la totalidad de las imágenes

REDES NEURONALES

Separamos los datos de entrenamiento diferenciando entradas y salidas

```
x=[] #lista que guardará los pixeles de las imágenes
y=[] #lista que guardará las etiquetas que indican si es perro o gato

for imagen,etiqueta in traindata:
  x.append(imagen)
  y.append(etiqueta)
```

REDES NEURONALES

Como los colores representados en los pixeles de las imágenes están en un rango de 0 a 255 es necesario normalizarlos para que se encuentren entre 0 y 1. Para ello utilizamos la librería numpy, convertimos los datos a float y dividimos entre 255 cada valor

```
import numpy as np
x=np.array(x).astype(float)/255
```

REDES NEURONALES

También convertimos las salidas en un arreglo con numpy, así los datos estarán en un formato simple que indicarán solo 1 o 0, según si es perro o gato

```
y=np.array(y)
y
```

```
array([1, 1, 1, ..., 0, 1, 0])
```

REDES NEURONALES

CREAR EL MODELO

Una red neuronal con una capa de entrada, dos capas ocultas de 150 neuronas cada una y una capa de salida

```
modeloDenso = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(100, 100, 1)),
  tf.keras.layers.Dense(150, activation='relu'),
  tf.keras.layers.Dense(150, activation='relu'),
  tf.keras.layers.Dense(1, activation='sigmoid')
])
```

REDES NEURONALES

COMPILAR EL MODELO

```
modeloDenso.compile(tf.keras.optimizers.Adam(0.1), loss='binary_crossentropy',
                    metrics=['accuracy'])
```

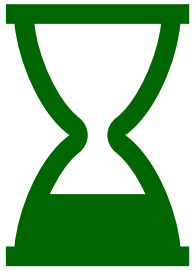
REDES NEURONALES

ENTRENAR EL MODELO

```
#Entrenar el modelo
from tensorflow.keras.callbacks import TensorBoard
tensorboardDenso = TensorBoard(log_dir='logs/denso')
entrenado = modeloDenso.fit(x, y,
                validation_split=0.15,
                epochs=100, callbacks=[tensorboardDenso])
```

El parámetro de validation_split, indica el porcentaje de datos que se tomará para las pruebas.

El entrenamiento puede tomar unos minutos.

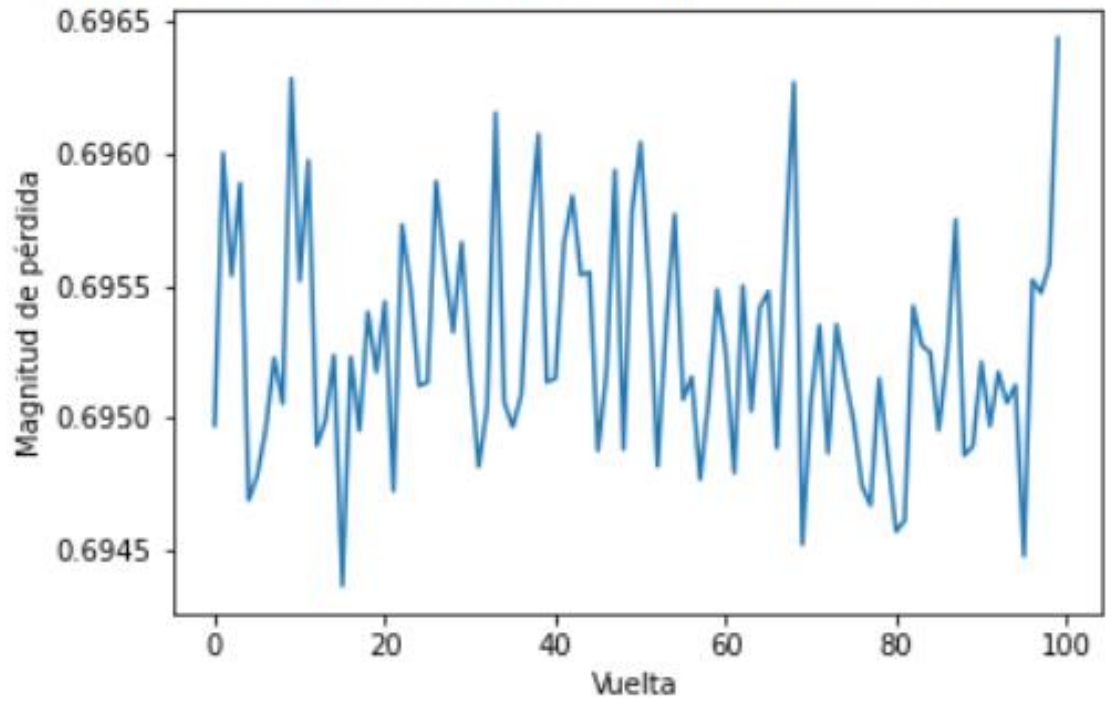


REDES NEURONALES

GRAFICAR LA PÉRDIDA

```
import matplotlib.pyplot as plt
plt.xlabel("Vuelta")
plt.ylabel("Magnitud de pérdida")
plt.plot(entrenado.history["loss"])
```

```
[<matplotlib.lines.Line2D at 0x7f769f341b50>]
```

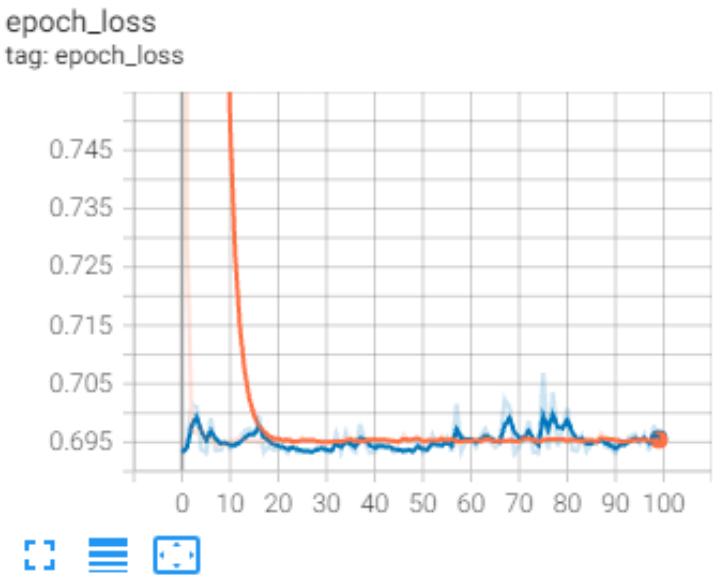


REDES NEURONALES

UTILIZAR UNA GRAFICADORA DE COLAB

```
#Cargar la extension de tensorboard de colab
%load_ext tensorboard
```

```
#Ejecutar tensorboard e indicarle que lea la carpeta "logs"
%tensorboard --logdir logs
```



REFERENCIAS

- https://www.iartificial.net/redes-neuronales-desde-cero-i-introduccion/#Redes_neuronales
- https://www.youtube.com/watch?v=DbwKbsCWPSg