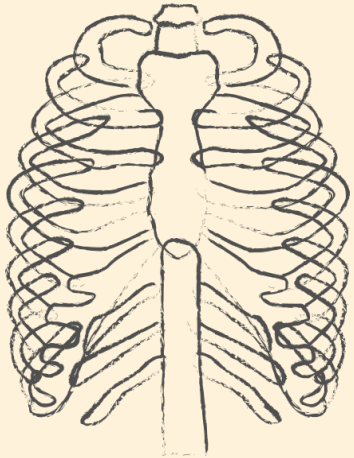
PROYECTO DEEP LEARNING

Ruben Tenreiro
Jesús Moncho
Marco Hernani
Helen Navarro

ÍNDICE

INTRODUCCIÓN

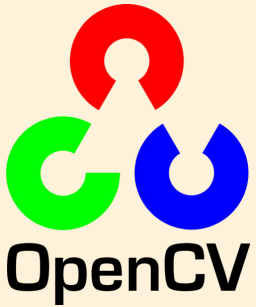


Squad ETL

Nos encargaron el proyecto de detectar mediante reconocimiento de imágenes si un paciente tiene neumonía o no, con un proceso de entrenamiento de DEEP LEARNING.

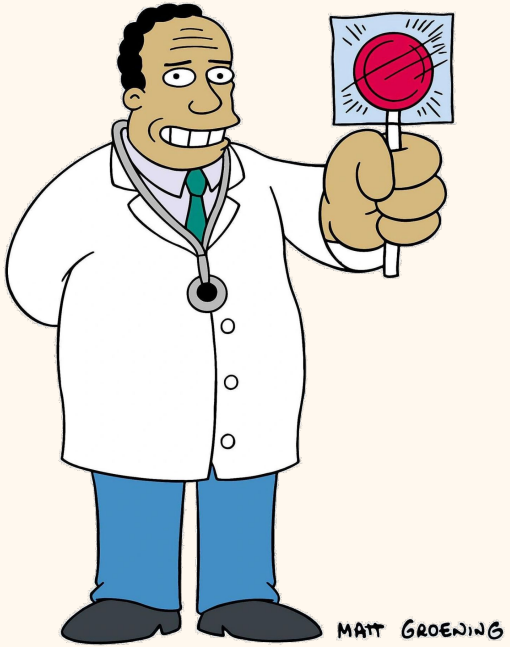
Importación de librerías

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout , BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from keras.callbacks import ReduceLROnPlateau
import cv2
import os
```



01

CARGA DE IMÁGENES



SIGUIENTE

POR FAVOR

OpenCV

Las imágenes se convierten en matrices de 300 x 300

```
image_size = (300, 300)
```

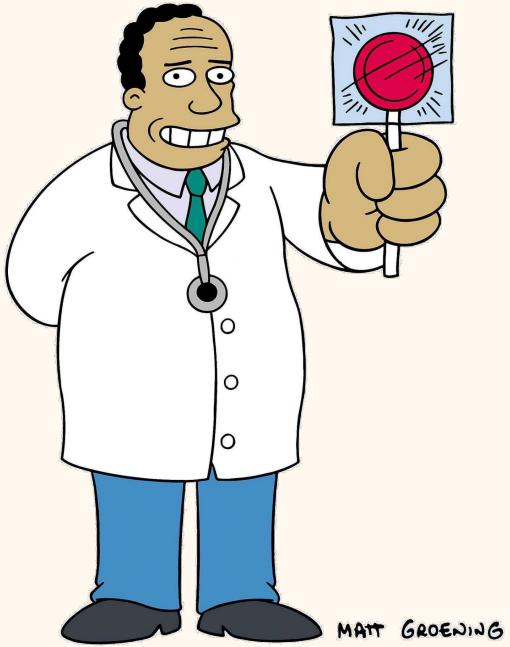
```
# What these images really are

train_data[0][0]
```

```
array([[18, 19, 24, ...,  5,  2,  0],
       [19, 21, 24, ...,  6,  3,  0],
       [18, 22, 24, ...,  7,  5,  0],
       ...,
       [10, 10, 10, ..., 15, 15, 15],
       [10, 10, 10, ..., 15, 15, 15],
       [10, 10, 10, ..., 15, 15, 15]], dtype=uint8)
```

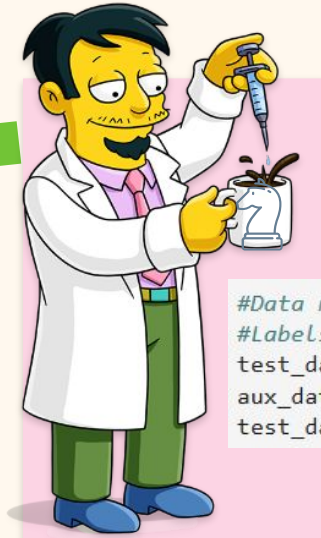
Cargamos las imágenes de radiografías con cv2

02

Etiquetado de imágenes



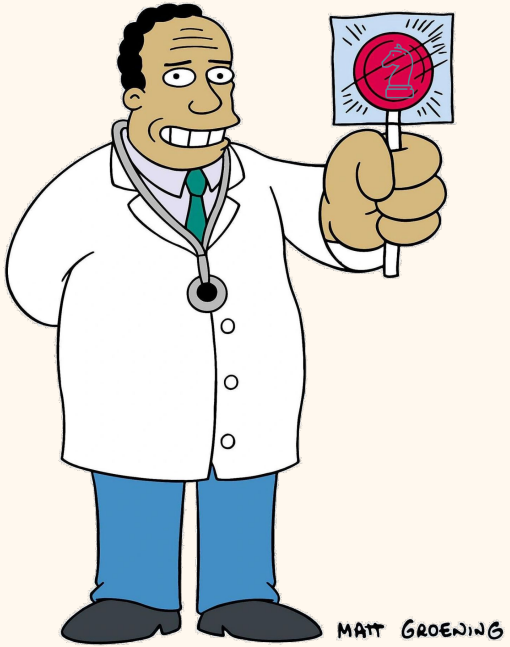
ETIQUETADO DE IMÁGENES



```
#Data needed to train our model
#Labels, 0:normal and 1:pneumonía
test_data = load_images("datasets/test/PNEUMONIA", image_size, 1)
aux_data = load_images("datasets/test/NORMAL", image_size, 0)
test_data.extend(aux_data)
```

03

Balanceo de clases

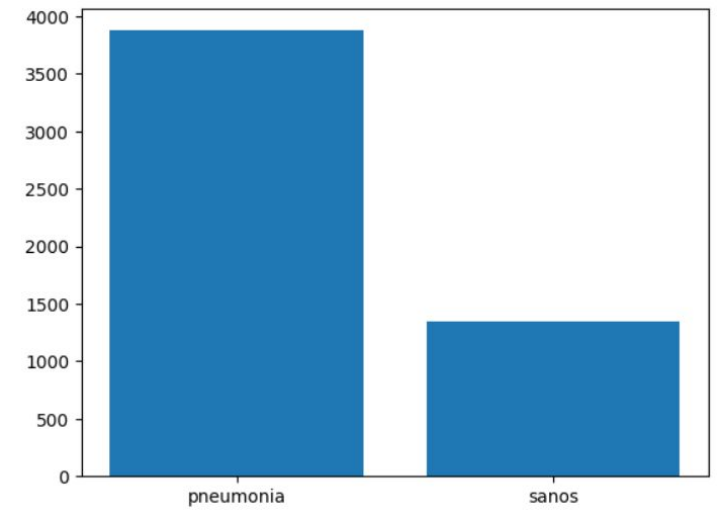


SIGUIENTE

POR FAVOR

Datos sin balancear

```
1 bar_data = [len(train_pneu), len(train_norm)]
2 plt.bar(["pneumonia", "sanos"], bar_data)
3 plt.show()
```

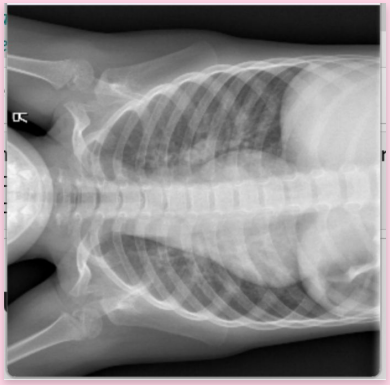


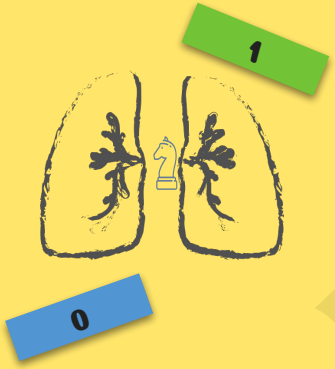

```
1 total = healthy + pneumonia
2 print(f"sanos: {round(healthy/total*100, 2)}%")
3 print(f"con pneumonia: {round(pneumonia/total*100, 2)}%")
```

```
sanos: 26.97%
con pneumonia: 73.03%
```

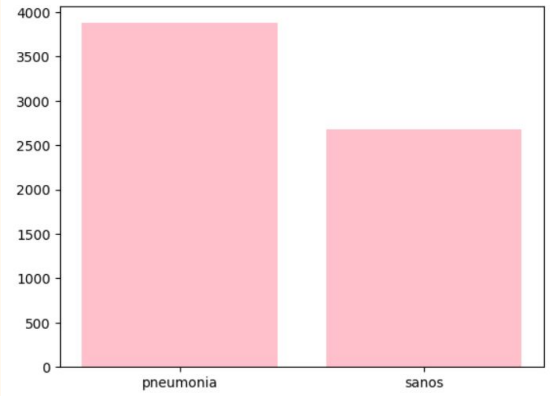
Balanceo de clases

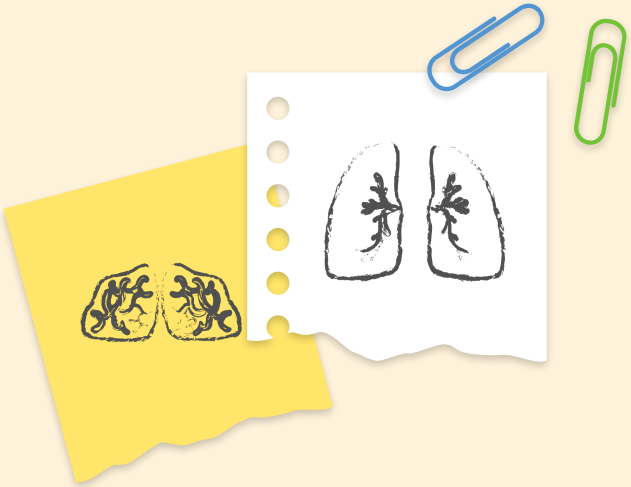
```
#We add normal data
for x, _ in train_normal[:]:
    train_data.append( (x.T, 0) )
```





```python
bar_data = [len(train_pneu), len(train_norm) + len(train_norm_mirror)]
plt.bar(["pneumonia", "sanos"], bar_data, color = "pink")
plt.show()
```

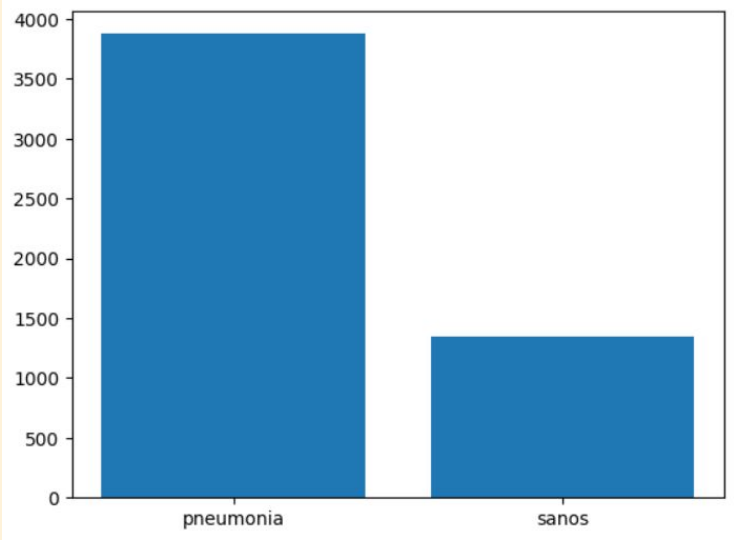


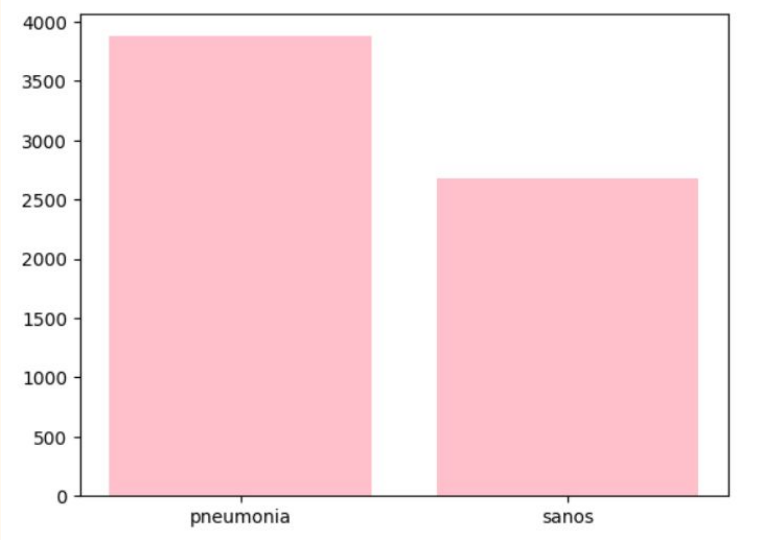


```python
train = train_pneu + train_norm + train_norm_mirror

print(f"sanos: {round(len(train_norm + train_norm_mirror)/len(train)*100, 2)}%")
print(f"con pneumonia: {round(len(train_pneu)/len(train)*100, 2)}%")
```

```
sanos: 40.9%
con pneumonia: 59.1%
```

Comparación Inicio-final





04

Entrenamiento

Normalización y creación de train y test sets

```python
#Outputs
train_y = []
#Inputs
train_x = []

for x,y in train_data:
    train_y.append(y)
    train_x.append(x)

train_y = np.array(train_y, dtype='float32')

train_x = np.array(train_x, dtype='float32')/255
```

```python
#Outputs
test_y = []
#Inputs
test_x = []

for x,y in test_data:
    test_y.append(y)
    test_x.append(x)

test_y = np.array(test_y, dtype='float32')

test_x = np.array(test_x, dtype='float32')/255
```

Creación del modelo

```
model = Sequential([
  Flatten(input_shape=image_size),
  Dense(32, activation='relu'),
  Dropout(0.2),
  Dense(16, activation='relu'),
  Dropout(0.2),
  Dense(8, activation='relu'),
  Dropout(0.2),
  Dense(1, activation="sigmoid") #'softmax')
])


model.compile(optimizer='adam',
              loss='binary_crossentropy',
              #loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Entrenamiento

```
model.fit(train_x, train_y, epochs=5)
```

```
Epoch 1/5
205/205 [==============================] - 6s 23ms/step - loss: 1.0538 - accuracy: 0.7423
Epoch 2/5
205/205 [==============================] - 5s 22ms/step - loss: 0.5409 - accuracy: 0.7837
Epoch 3/5
205/205 [==============================] - 4s 19ms/step - loss: 0.5064 - accuracy: 0.7927
Epoch 4/5
205/205 [==============================] - 4s 21ms/step - loss: 0.4822 - accuracy: 0.8062
Epoch 5/5
205/205 [==============================] - 4s 20ms/step - loss: 0.4466 - accuracy: 0.8280
<keras.callbacks.History at 0x1c4f97a5b10>
```

Resultados

```
model.evaluate(train_x,  train_y, verbose=2)
```

```
205/205 - 1s - loss: 0.3243 - accuracy: 0.9395 - 1s/epoch - 6ms/step
[0.3242920935153961, 0.9394540190696716]
```

```
model.evaluate(test_x,  test_y, verbose=2)
```

```
20/20 - 0s - loss: 0.4868 - accuracy: 0.8205 - 150ms/epoch - 7ms/step
[0.4867691695690155, 0.8205128312110901]
```

05

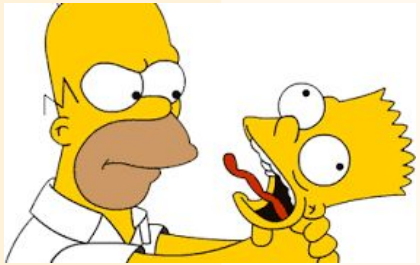
Incidencias

Una vez obtenido el modelo este hacía overfitting.

- Posible causa: falta de datos de personas sanas (1/4 respecto al total)
- Posible solución: crear nuevas imagenes rotando las anteriores

> 1/4 sanas -> aprox 2/4 2000 -> 2/4 4000
> 3/4 pneu -> aprox 2/4 2000-> 2/4 4000

accuracy: **~70 test, ~90 train**



Gracias por vuestra atención.
¿Dudas?