



Lung Disease.

Notes: All the code that we will use in this thesis is python.

probably we will use flask to create the web app.

also we will use tailwind for the css and of course html to create the structure and to add the tailwind css

Basic description of the dataset:

Dataset Contents:

Total Number of Images: The dataset comprises a total of 3,475 X-ray images.

Classes within the Dataset:

Normal (1250 Images): These images represent healthy lung conditions, serving as a reference for comparison in diagnostic procedures.

Lung Opacity (1125 Images): This class includes X-ray images depicting various degrees of lung abnormalities, providing a diverse set of cases for analysis.

Viral Pneumonia (1100 Images): Images in this category are associated with viral pneumonia cases, contributing to the understanding and identification of this

specific lung infection.

In conclusion, the "Lung X-Ray Image Dataset" plays a crucial role in the healthcare sector by providing a diverse and well-documented collection of X-ray images that support the detection, classification, and understanding of lung diseases. This resource is instrumental in advancing the field of respiratory medicine and improving patient outcomes.

Dataset Overview:

- **Total images:** 3,475
- **Classes:**
 - **Normal (1250 images)** – Healthy lungs
 - **Lung Opacity (1125 images)** – Abnormal lung conditions
 - **Viral Pneumonia (1100 images)** – X-rays showing viral infection

This high-quality dataset, collected from various healthcare institutions, is valuable for training diagnostic models and improving outcomes in **respiratory medicine**.

Dividing the code step by step:

1-) Import the libraries.

2-) Process all the data that we have, basically we created a dataframe where we will store all the data of each path and label.

3-) divide the dataset into a train dataset and a testing dataset.

```
print(train_df.shape)
print(test_df.shape) #dividimos nuestros datos,
print(val_df.shape)

(2215, 2)
(869, 2)
(391, 2)
```

In that image we can see how we have 2215 image to train the convolutional neural network (**CNN**)

869 are the image that we will use to test the model.

4-) Then we create a function that improve each image constrast and the borders of each image, that function is define as : **enhance_image**


5-) Process each image, first we process the Train, test and then the validation. we convert the values of the scale from 0-255 to 0-1 to make the training easier, it is just a normalization.

.flow_from_dataframe(...)

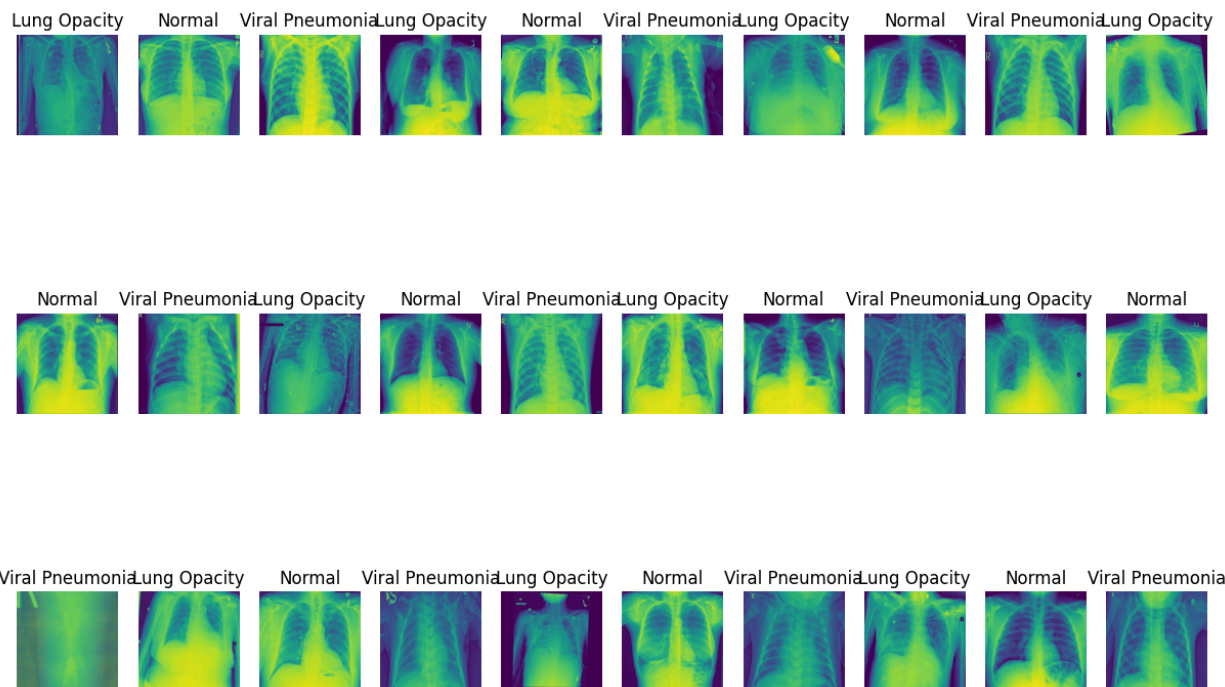
Este método **carga imágenes desde un DataFrame**, donde se indican las rutas de las imágenes (**x_col**) y sus etiquetas (**y_col**). Se crean tres generadores diferentes:

```
train = image_gen.flow_from_dataframe(dataframe= train_df,x_col="filepaths",y_col="labels",
                                     target_size=(256,256),
                                     color_mode='rgb',
                                     class_mode="categorical",
                                     batch_size=32,
                                     shuffle=False
                                     )
```

Recapitulación Visual:

Dataset	¿Cuándo se usa?	¿Qué hace?	Aprende de él 
train	Durante entrenamiento	Modelo aprende los patrones	✓ Sí
val	Durante entrenamiento	Verifica si el modelo está mejorando sin memorizar	✗ No
test	Después del entrenamiento	Mide el rendimiento real	✗ No

6-) we plot some of the image and their actual label, using matplotlib.



notas: seria interesante poder predecir mas enfermedades, como podrían ser Covid-19, tuberculosis entre otras enfermedades.

7-) Train the CNN MobileNet with the data that we have in our dataset.

```
#aca entrenamos el modelo, con nuestros datos, hay que tener en cuenta que lo
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Dropout
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adamax

# Load pre-trained MobileNet model without the top (fully connected) layers
base_model = MobileNet(weights=None, include_top=False, input_shape=(256, 256, 3))
base_model.load_weights('./models/mobilenet_weights.h5')

# Add custom top layers for your specific task
x = base_model.output
x = BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.45)(x)
predictions = Dense(3, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

optimizer = Adamax(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(train, epochs=20, validation_data=val)
```

- **MobileNet** es una CNN ligera y eficiente, preentrenada sobre ImageNet.
- Con `include_top=False` quitamos las capas finales de clasificación genéricas de ImageNet para poder poner nuestras propias capas.

- `pooling='max'` aplica un Max-Pooling global al final de la red, reduciendo cada mapa de características a un valor máximo, lo que simplifica la entrada a las capas densas posteriores.
- Cargamos manualmente los pesos descargados (`mobilenet_weights.h5`), que contienen los filtros entrenados para reconocer patrones visuales generales (bordes, texturas, formas).

básicamente lo que hacemos en ese ultimo código es:

- en el **feature-extractor** (MobileNet) ya trae filtros útiles para imagen (bordes, texturas).
- añadimos capas densas y regularización para adaptarlo a tu problema de 3 clases pulmonares (normal, neumonía y opacidad pulmonar).
- Con `fit()` , optimizamos los pesos de esas capas nuevas (y posiblemente reajustaste ligeramente las capas base, dejando el learning rate lo suficientemente alto).