26 DE NOVIEMBRE DE 2023

DATA SCIENCE INTERSHIP AT DATA GLACIER

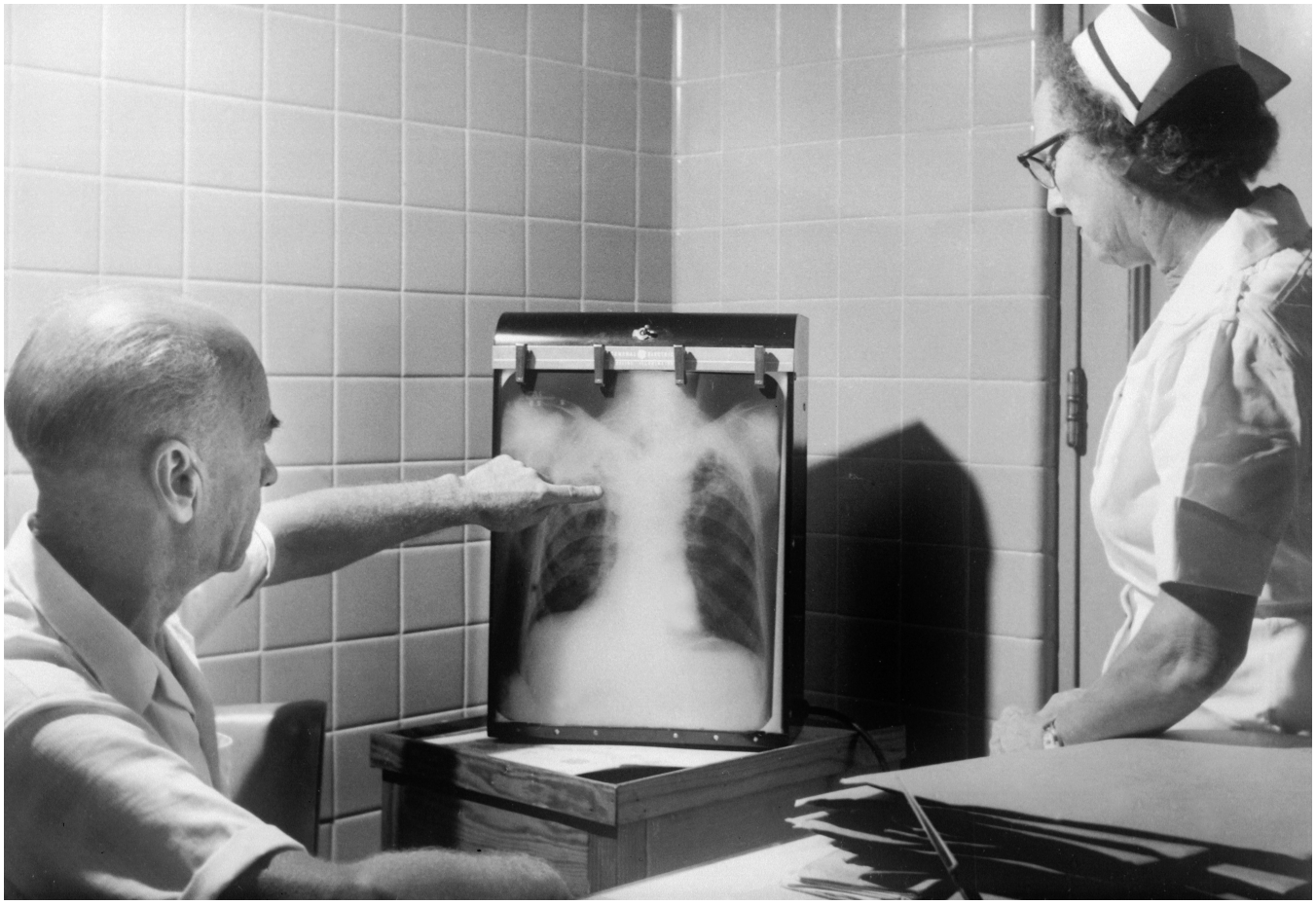
DEPLOYMENT FLASK

Jorge Alberto Jaramillo Bermúdez

DATA SCIENTIST

COVID-19 CLASSIFICATION IN CHEST X-RAYS

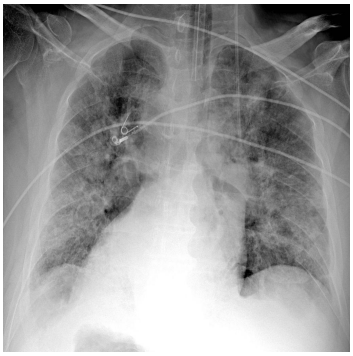
More than three years have passed since the first case of infection with a new coronavirus (CoV) (SARS-CoV-2) in Wuhan city (Hubei, China). In late November 2019, near Huanan market, Wuhan, Hubei province (China) described the first case of pneumonia due to a new CoV, also of the genus beta, which was initially designated 2019-nCoV by researchers in China. On February 11, 2020, it was renamed SARS-CoV-2 and the disease was named COVID-19. The SARS-CoV-2 pandemic has triggered an unprecedented economic and health crisis. Although the diagnosis is microbiological, imaging techniques play an important role in supporting the diagnosis, grading the severity of the disease, guiding treatment, detecting possible complications and assessing therapeutic response. The involvement is mainly pulmonary. Chest radiography in a conventional or portable room is the first imaging method due to its wide availability and low cost, and the most frequent radiological findings are airspace

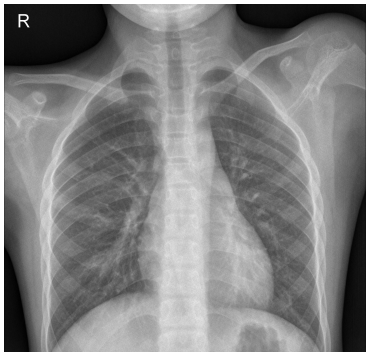


opacities in the form of consolidations and/or ground glass opacities, typically bilateral, peripheral and predominantly in the lower fields.

INTRODUCTION

The introduction highlights the timeline from the first case of infection with the new coronavirus (SARS-CoV-2) in Wuhan, China, to the declaration of the COVID-19 pandemic. The importance of imaging techniques, such as chest X-rays, in the diagnosis, monitoring, and treatment of the disease is emphasized. It is mentioned that the project aims to implement a deep neural network to classify patients with suspected COVID-19 infection from these images.





DATA INFORMATION:

A link to Google Drive containing the project's data is provided. Pathways to the training and testing folders are specified, and the first 10 images from the COVID and NORMAL classes in both sets are displayed. This gives an overview of the data used in the project.

```python
import os


# Path to the 'Datos' folder in Google Drive
datos_folder_path = '/content/drive/My Drive/Datos'

# Paths to the 'train' and 'test' folders inside 'Datos'
train_folder_path = os.path.join(datos_folder_path, 'train')
test_folder_path = os.path.join(datos_folder_path, 'test')

# List files in the 'train' folder
files_in_train = os.listdir(train_folder_path)
print("Files in 'train' folder:", files_in_train)

# List files in the 'test' folder
files_in_test = os.listdir(test_folder_path)
print("Files in 'test' folder:", files_in_test)
```

```
Files in 'train' folder: ['NORMAL', 'COVID']
Files in 'test' folder: ['NORMAL', 'COVID']
```

```
# Training Set
train_covid_names = sorted(os.listdir(train_covid_path))
print("\n First 10 images of Training Set (COVID):", train_covid_names[0:10] )
train_normal_names = sorted(os.listdir(train_normal_path))
print("\n First 10 images of Training Set (NORMAL):", train_normal_names[0:10])
```

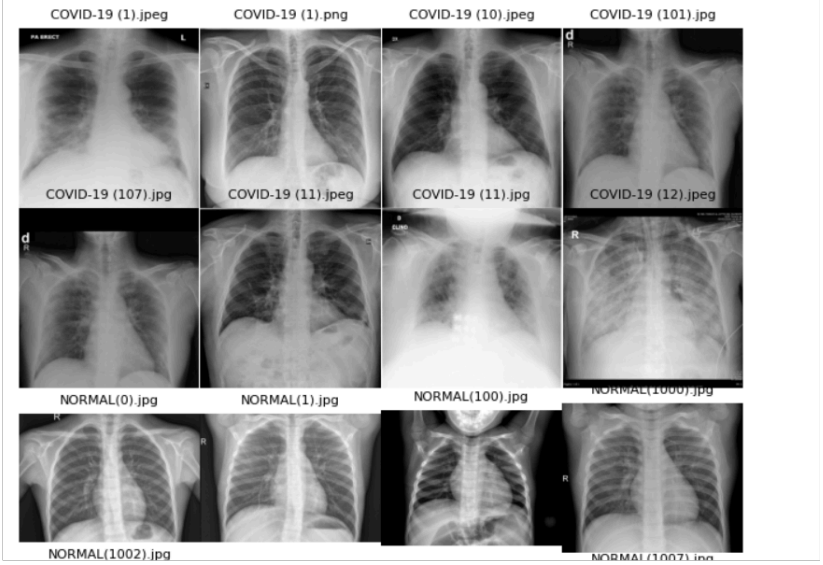
```
 First 10 images of Training Set (COVID): ['COVID-19 (1).jpeg', 'COVID-19 (1).png', 'COVID-19 (10).jpeg', 'COVID-19 (101).

 First 10 images of Training Set (NORMAL): ['NORMAL(0).jpg', 'NORMAL(1).jpg', 'NORMAL(100).jpg', 'NORMAL(1000).jpg', 'NORM
```

```
[6] # Test Set
test_covid_names = sorted(os.listdir(test_covid_path))
print("\n First 10 images of Test Set (COVID):", test_covid_names[0:10])
test_normal_names = sorted(os.listdir(test_normal_path))
print("\n First 10 images of Test Set (COVID):", test_normal_names[0:10])
```

```
 First 10 images of Test Set (COVID): ['COVID-19 (313).jpg', 'COVID-19 (353).jpg', 'COVID-19 (371).jpg', 'COVID-19 (425).j

 First 10 images of Test Set (COVID): ['NORMAL(10).jpg', 'NORMAL(1001).jpg', 'NORMAL(1003).jpg', 'NORMAL(1004).jpg', 'NORM
```

ATTRIBUTE INFORMATION:

Specific information about the attributes of the data used is not provided. However, it is revealed that the classes are "COVID" and "NORMAL," indicating the label for each chest X-ray image.



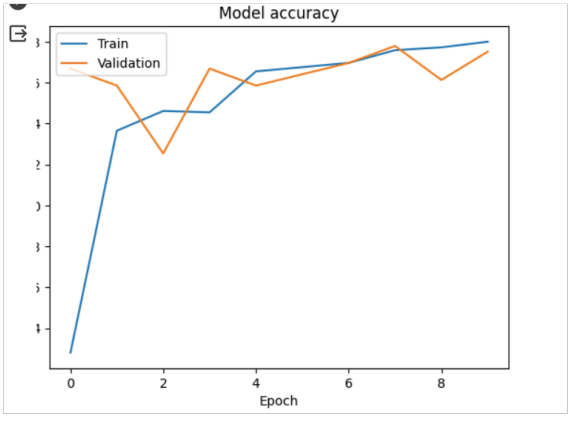
BUILD OF ML OR DL MODEL: TURNING MODEL INTO FLASK FRAMEWORK:

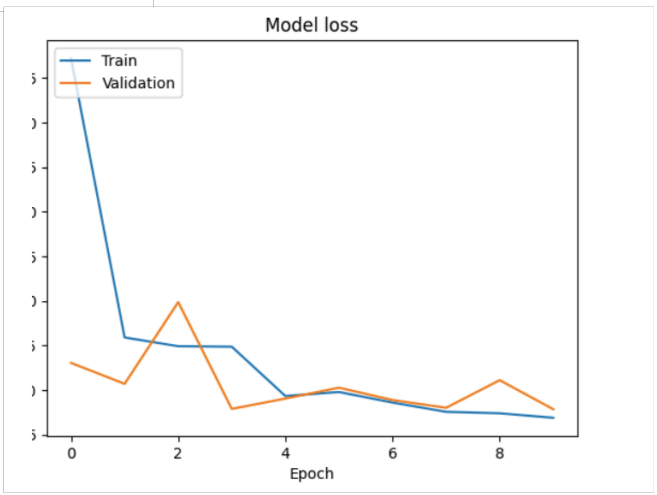
A convolutional neural network (CNN) model is built using TensorFlow and Keras. The network architecture includes convolutional layers, max-pooling, flattening, fully connected layers, and dropout to prevent overfitting. The model is compiled and trained using the Adam optimizer and binary cross-entropy loss.

Additionally, it is mentioned that this model will be translated into a web framework called Flask. Necessary configurations, such as modifying the runtime environment in Google Colab to support GPU hardware, are detailed. Code for importing libraries, setting up data batch generators, and visualizing some dataset images is provided.

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2  (None, 74, 74, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 36, 36, 64)        0
 g2D)

 flatten (Flatten)           (None, 82944)             0

 dense (Dense)               (None, 128)               10616960

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 10636481 (40.57 MB)
Trainable params: 10636481 (40.57 MB)
Non-trainable params: 0 (0.00 Byte)
```

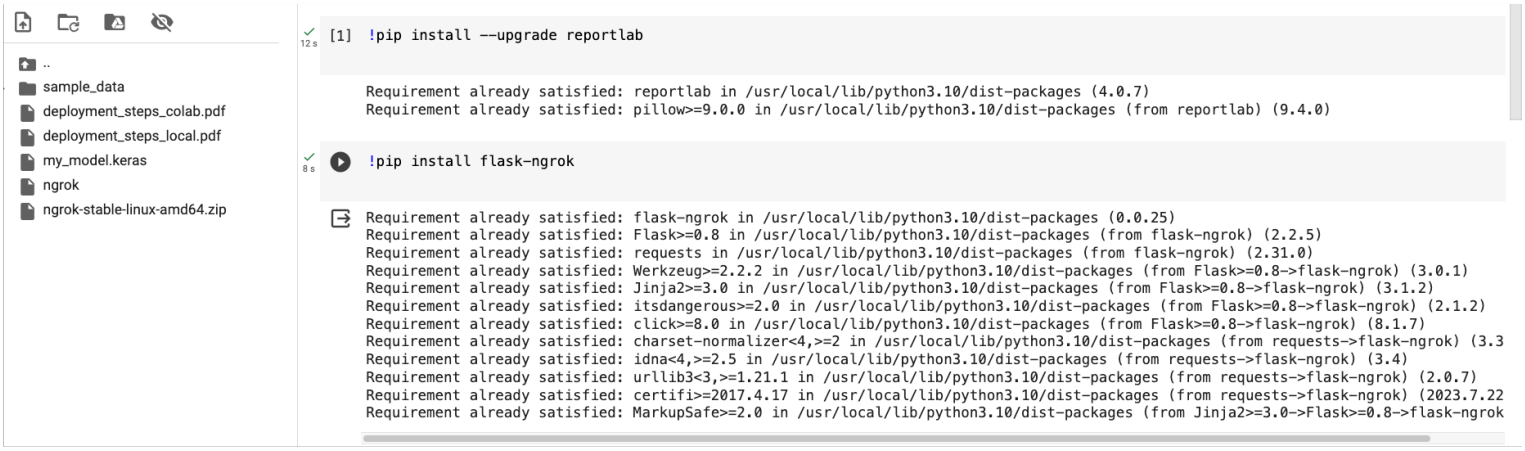




```
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix

# Calculate precision, recall, and F1-score
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)

print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print('Results on test set')
print(" - Loss: {:.2f}, Accuracy: {:.2f} ".format(test_loss, test_accuracy))
```

```
Precision: 0.6206896551724138
Recall: 0.6246056782334385
F1-score: 0.6226415094339622
Results on test set
 - Loss: 0.06, Accuracy: 0.98
```

1.1 TURNING MODEL INTO FLASK FRAMEWORK:



WEEK 4

```python
from flask import Flask, render_template, request, jsonify
from tensorflow.keras.models import load_model
from reportlab.pdfgen import canvas
import datetime
from google.colab import files
from flask_ngrok import run_with_ngrok  # Import run_with_ngrok

# Loading the model
model = load_model('my_model.keras')

# Creating the Flask application
app = Flask(__name__)
run_with_ngrok(app)  # Initialize ngrok when the app is run

# Defining the route to handle GET requests (home page)
@app.route('/')
def home():
    return render_template('index.html')  # Adjust the name according to your initial template

# Define the route to handle POST requests (prediction)
@app.route('/predict', methods=['POST'])
def predict():
    # Get data from the request
    data = request.form['input_text']  # Adjust based on how you get data from the form

    # Make a prediction
    prediction = model.predict([data])[0]

    # Render the template with the prediction
    return render_template('result.html', prediction=prediction)

# Replacing with your own values
my_name = 'Jorge Alberto Jaramillo Bermudez'
batch_code = 'LISUM27'
submitted_to = 'Data Glacier'

# Creating a PDF document for Colab
pdf_path_colab = '/content/deployment_steps_colab.pdf'
pdf_colab = canvas.Canvas(pdf_path_colab)
pdf_colab.drawString(10, 800, f'Name: {my_name}')
pdf_colab.drawString(10, 780, f'Batch code: {batch_code}')
pdf_colab.drawString(10, 760, f'Submission date: {datetime.datetime.now()}')
pdf_colab.drawString(10, 740, f'Submitted to: {submitted_to}')
pdf_colab.save()
```

```python
# Create another PDF document for local download
pdf_local_path = '/content/deployment_steps_local.pdf'
pdf_local = canvas.Canvas(pdf_local_path)
pdf_local.drawString(10, 800, f'Name: {my_name}')
pdf_local.drawString(10, 780, f'Batch code: {batch_code}')
pdf_local.drawString(10, 760, f'Submission date: {datetime.datetime.now()}')
pdf_local.drawString(10, 740, f'Submitted to: {submitted_to}')
pdf_local.save()

# Start the Flask application
if __name__ == '__main__':
    app.run()  # No use_reloader or debug when using ngrok
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
 * Running on http://4fe6-34-125-208-36.ngrok-free.app
 * Traffic stats available on http://127.0.0.1:4040
ERROR:__main__:Exception on / [GET]
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/flask/app.py", line 2529, in wsgi_app
    response = self.full_dispatch_request()
  File "/usr/local/lib/python3.10/dist-packages/flask/app.py", line 1825, in full_dispatch_request
    rv = self.handle_user_exception(e)
  File "/usr/local/lib/python3.10/dist-packages/flask/app.py", line 1823, in full_dispatch_request
    rv = self.dispatch_request()
  File "/usr/local/lib/python3.10/dist-packages/flask/app.py", line 1799, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
  File "<ipython-input-9-b46b47da351a>", line 18, in home
    return render_template('index.html')  # Adjust the name according to your initial template
  File "/usr/local/lib/python3.10/dist-packages/flask/templating.py", line 146, in render_template
    template = app.jinja_env.get_or_select_template(template_name_or_list)
  File "/usr/local/lib/python3.10/dist-packages/jinja2/environment.py", line 1081, in get_or_select_template
    return self.get_template(template_name_or_list, parent, globals)
  File "/usr/local/lib/python3.10/dist-packages/jinja2/environment.py", line 1010, in get_template
    return self._load_template(name, globals)
  File "/usr/local/lib/python3.10/dist-packages/jinja2/environment.py", line 969, in _load_template
```