BIG DATA ANALYTICS FOR PNEUMONIA, COVID-19 DETECTION USING CNN

Team:
RAMYA SRI TELLAKULA
LAKSHMI DEEPIKA KARLAPUDI
MANASA JAGATI

Project Description

Description:

Our project focuses on the development and deployment of a Convolutional Neural Network (CNN) model tailored specifically for the detection of Pneumonia and COVID-19 from chest X-ray images.

Objective:

Our project aim to create an accurate and efficient model by leveraging deep learning techniques to enhance early diagnosis, patient management, and aid in disease containment.

1. **Problem Statement:**

The pandemic has highlighted the critical importance of early and accurate diagnosis for effective patient care and disease control. Traditional diagnostic methods often require time-consuming laboratory tests, leading to delays in diagnosis and treatment.

2. **Approach:**

Using big data tech like Spark, we analyze extensive imaging data. By training a CNN model on varied chest X-ray images, including COVID-19 and Pneumonia cases, we aim for efficient detection.

3. **Impact:**

Our model aids healthcare by swiftly classifying new chest X-ray images, enabling informed decisions for early COVID-19 detection and management, contributing to pandemic control.

Data Set Information

COVID-19 Dataset:

- Source: Kaggle
- Title: "COVIDx CXR-4 - Chest X-ray images for the detection of COVID-19"
- Description: The dataset consists of chest X-ray images obtained from diverse medical institutions, containing both positive and negative COVID-19 cases.
- Format: All files are in PNG format.
- Size: 31 GB
- Link: https://www.kaggle.com/datasets/andyczhao/covidx-cxr2

Pneumonia (Chest X-Ray) Dataset:

- Source: Kaggle
- Title: "Viral Pneumonia Classification"
- Description: This dataset comprises chest X-ray images specifically focused on pneumonia cases.
- Format: PNG
- Size: 2 GB
- Link: https://www.kaggle.com/code/chaitanya99/viral-pneumonia-classification-googlenet

Key Implementations

1. **Setting up Spark Session:**

   Created a Spark Session to efficiently load data stored on the local disk, facilitating seamless data processing.

2. Data Preparation and Processing:

1. Defined data directories and read images using Spark.
2. Utilized Spark for categorizing, preparing, and processing both Pneumonia and COVID-19 data.
3. Leveraged RDD functions for efficient data manipulation.
4. Visually displayed images of COVID-19, Pneumonia, and normal cases.

3. Exploratory Data Analysis (EDA):

1. Analyzed class distribution to understand the dataset's composition.
2. Visualized the distribution of classes to gain insights.

Key Implementations

4. **Image Data Cleaning:**

   Removed unopened and inaccessible images from the dataset, ensuring data quality.

5. **Data Splitting, Analysis, and Model Building:**

   1. Split data and trained the model using Convolutional Neural Networks (CNN) on SET of images (2000, 5000 and 20000 images).
   2. Chunked the data into 2000 test images, built a model, and verified accuracy.
   3. Plotted accuracy and loss graphs to assess model performance.
   4. Evaluated the model for SET1 images, calculating accuracy, precision, recall, and F1 scores.
   5. Displayed the confusion matrix to visualize model performance.

6. **Model Evaluation on Unknown Images:**

   Ran unknown images through the trained model and predicted the results, enabling real-world application and validation of the model's effectiveness.

Pneumonia Data Preparation

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3 (ipykernel)

Run Code

In [8]:
```
#Paths to the data directories
train_image_dir = PneumoniaDir + '/train'
test_image_dir = PneumoniaDir + '/test'
val_image_dir = PneumoniaDir + '/val'

# Schema for the Spark DataFrame
schema = StructType([
    StructField("image_path", StringType(), True),
    StructField("label", StringType(), True)
])

# Function to prepare data from a directory and return a Spark DataFrame
def prepare_data_spark(image_dir, strat):

    # List of categories in the dataset
    categories = ["NORMAL", "PNEUMONIA"]

    # Initialize an empty list to store data
    data = []

    # Iterate through each category and process images
    for category in categories:
        category_path = os.path.join(image_dir, category)

        # Iterate through each image file in the category
        for img_file in tqdm(os.listdir(category_path)):
            img_path = os.path.join(category_path, img_file)

            # Append a tuple with the image path and label
            data.append((img_path, category))

    # Create a Spark DataFrame from the list of data tuples
    df = spark.createDataFrame(data, schema=schema)
    # Add a column for the strategy (e.g., train, test, val)
    df = df.withColumn("strategy", lit(strat))

    return df

# Prepare data for train, test, and validation sets using Spark
train_df = prepare_data_spark(train_image_dir, 'train')
test_df = prepare_data_spark(test_image_dir, 'test')
val_df = prepare_data_spark(val_image_dir, 'val')
```

```
100%|| 1341/1341 [00:00<00:00, 685337.11it/s]
100%|| 3875/3875 [00:00<00:00, 1135377.44it/s]
100%|| 234/234 [00:00<00:00, 2088227.95it/s]
100%|| 391/391 [00:00<00:00, 2477300.40it/s]
100%|| 8/8 [00:00<00:00, 316551.25it/s]
100%|| 8/8 [00:00<00:00, 360800.34it/s]
```

Covid -19 Data Preparation

File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Code

Covid-19 Data Preparation

In [11]:
```
# Schema for the Spark DataFrame
schema = StructType([
    StructField("image_id", StringType(), True),
    StructField("image_path", StringType(), True),
    StructField("label", StringType(), True)
])

# Image data directory and file paths

train_txt_path = os.path.join(DataDir, "train.txt")
test_txt_path = os.path.join(DataDir, "test.txt")
val_txt_path = os.path.join(DataDir, "val.txt")

# Function to classify label
def classify_label(label):
    if label == "negative":
        return "NORMAL"
    elif label == "positive":
        return "COVID"
    else:
        return None

# Function to process text file and create a Spark DataFrame
def process_txt_to_df(txt_path, strat):

    # Read the text file as an RDD
    rdd = spark.read.text(txt_path).rdd

    # Process each line in the RDD
    def process_line(line):
        querywords = line[0].split()  # Line is a tuple, extract the text
        image_id = querywords[0]
        img_formats = ['jpg', 'jpeg', 'png']

        # Determine the image path and label based on the line length
        if len(querywords) == 4:
            image_path = os.path.join(DataDir, strat, querywords[1])
            label = querywords[2]
        elif len(querywords) == 5:
            image_path = os.path.join(DataDir, strat, querywords[2])
            label = querywords[3]
        else:
            return None

        # Check the image format
        for img_format in img_formats:
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
            label = querywords[3]
        else:
            return None

        # Check the image format
        for img_format in img_formats:
            if img_format in image_path:
                # Classify label
                classified_label = classify_label(label)
                if classified_label:
                    return (image_id, image_path, classified_label)

        return None

    # Map and filter RDD
    processed_rdd = rdd.map(process_line).filter(lambda x: x is not None)

    # Convert RDD to DataFrame
    df = spark.createDataFrame(processed_rdd, schema=schema)

    # Add strategy column
    df = df.withColumn("strategy", lit(strat))

    return df

# Process train, test, and validation text files
train_df = process_txt_to_df(train_txt_path, "train")
test_df = process_txt_to_df(test_txt_path, "test")
val_df = process_txt_to_df(val_txt_path, "val")
```

In [12]:
```
# Combine the data frames
covid_data_df = train_df.unionByName(test_df).unionByName(val_df)

# Drop the image_id column - dropped image_id column to equate the no.of columns in both datasets
covid_data_df1 = covid_data_df.drop("image_id")
```

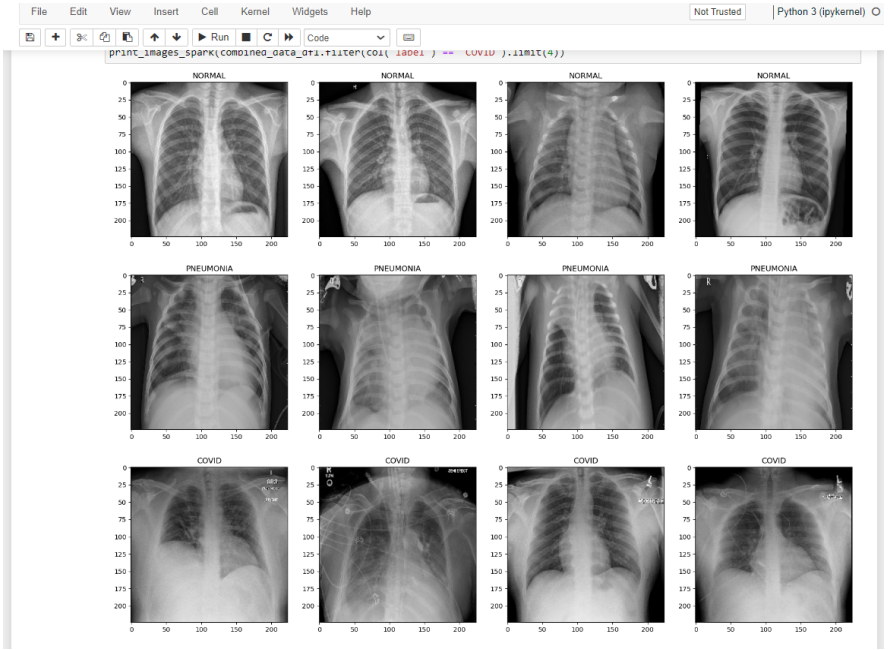
In [13]:
```
# Save the combined DataFrame to a CSV file
covid_csv_path = "/Users/ramyasritellakula/Documents/Covid_19_detection/result/covid_data.csv"

covid_data_df1.write.csv(covid_csv_path, header=True, mode="overwrite")
```

Visualization -Displaying Images Using Spark

Visualization -Displaying Images Using Spark DataFrame

In [17]:
```
# Function to display images using Spark DataFrame
def print_images_spark(samples_df):

    # Convert PySpark DataFrame to Pandas DataFrame
    samples_pd = samples_df.toPandas()

    # Get image paths and labels as NumPy arrays
    images = samples_pd["image_path"].to_numpy()
    labels = samples_pd["label"].to_numpy()

    # Set up the figure
    fig = plt.figure(figsize=(20, 8))
    columns = 4
    rows = 1

    # Display images
    for i, image_path in enumerate(images):
        # Read the image using OpenCV
        image = cv2.imread(image_path, cv2.IMREAD_COLOR)

        # Add a subplot for each image
        fig.add_subplot(rows, columns, i + 1)

        # Resize the image for better visualization
        resized_image = cv2.resize(image, (224, 224), interpolation=cv2.INTER_CUBIC)

        # Display the image
        plt.imshow(resized_image, cmap='gray')
        # Set the title to the corresponding label
        plt.title(labels[i])

    # Show the plot
    plt.show()
```

In [18]:
```
# Load the data from the CSV file
# Filter samples with different labels and display images

print_images_spark(combined_data_df1.filter(col("label") == "NORMAL").limit(4))
print_images_spark(combined_data_df1.filter(col("label") == "PNEUMONIA").limit(4))
print_images_spark(combined_data_df1.filter(col("label") == "COVID").limit(4))
```



Exploratory Data Analysis (EDA)

1. Data Overview - Prints the schema and shows a sample of the data.

In [19]:
```
# Print Data Schema
print("Data Schema:")
combined_data_df1.printSchema()
```

```
Data Schema:
root
 |-- image_path: string (nullable = true)
 |-- label: string (nullable = true)
 |-- strategy: string (nullable = true)
```

In [20]:
```
# Print Data Sample (Top 5 rows)
print("\nData Sample (Top 5 rows):")
combined_data_df1.show(5)
```

```
Data Sample (Top 5 rows):
+--------------------+------+--------+
|          image_path| label|strategy|
+--------------------+------+--------+
|/Users/ramyasrite...|NORMAL|    test|
|/Users/ramyasrite...|NORMAL|    test|
|/Users/ramyasrite...|NORMAL|    test|
|/Users/ramyasrite...|NORMAL|    test|
|/Users/ramyasrite...|NORMAL|    test|
+--------------------+------+--------+
only showing top 5 rows
```

```
Data Sample (Top 5 rows):
+--------------------+------+--------+
|          image_path| label|strategy|
+--------------------+------+--------+
|/Users/ramyasrite...|NORMAL|    test|
|/Users/ramyasrite...|NORMAL|    test|
|/Users/ramyasrite...|NORMAL|    test|
|/Users/ramyasrite...|NORMAL|    test|
|/Users/ramyasrite...|NORMAL|    test|
+--------------------+------+--------+
only showing top 5 rows
```

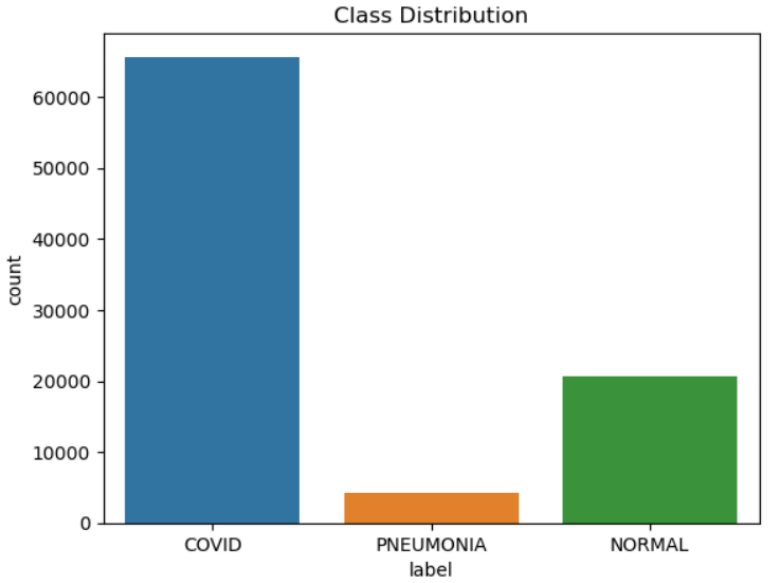
In [21]:
```
# Print Data Sample (Random 5 rows)
print("\nData Sample (Random 5 rows):")
combined_data_df1.sample(fraction=0.01, seed=42).show(5)
```

```
Data Sample (Random 5 rows):
+--------------------+---------+--------+
|          image_path|    label|strategy|
+--------------------+---------+--------+
|/Users/ramyasrite...|PNEUMONIA|    test|
|/Users/ramyasrite...|PNEUMONIA|    test|
|/Users/ramyasrite...|PNEUMONIA|    test|
|/Users/ramyasrite...|    COVID|    test|
|/Users/ramyasrite...|    COVID|    test|
+--------------------+---------+--------+
only showing top 5 rows
```

Class Distribution of Images

```
In [22]: # Class Distribution
         class_distribution = combined_data_df1.groupBy("label").count()
         print("\nClass Distribution:")
         class_distribution.show()
```

```
Class Distribution:
+---------+-----+
|    label|count|
+---------+-----+
|    COVID|65681|
|PNEUMONIA| 4274|
|   NORMAL|20718|
+---------+-----+
```



Data Cleaning

1. Removal of Rows with Missing Values
2. Removal of Duplicate Rows
3. Verification of Image Paths
4. Validation of Labels

In [24]:
```
# Load the combined data
combined_data_df1 = spark.read.csv("/Users/ramyasritellakula/Documents/Covid_19_detection/result/complete_data.csv",
                                   header=True, inferSchema=True)
```

In [25]:
```
# Step 1: Remove rows with missing values in important columns
cleaned_df = combined_data_df1.dropna(subset=['image_path', 'label'])

# Step 2: Remove duplicate rows based on image path
cleaned_df = cleaned_df.dropDuplicates(['image_path'])
```

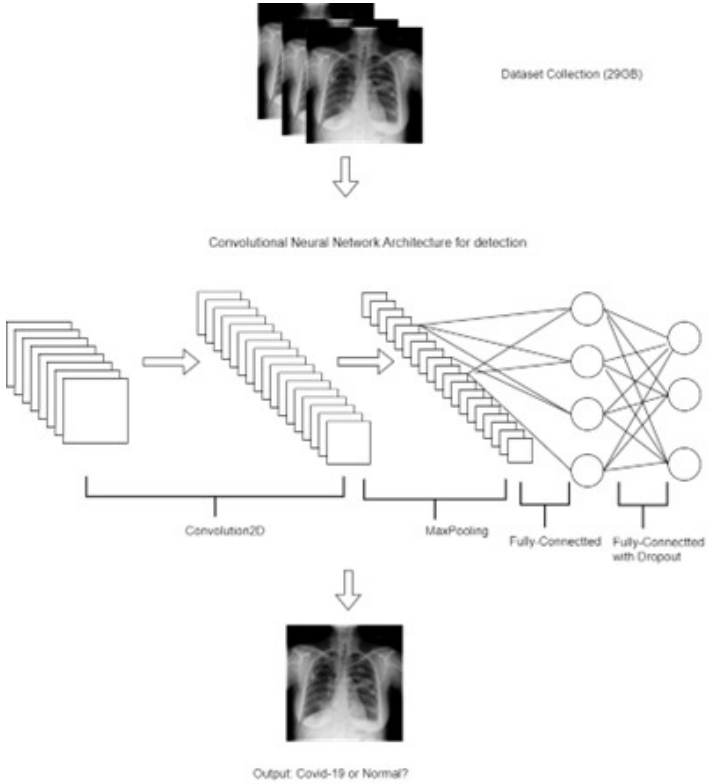
In [26]:
```
# Step 3: Verify image paths
# Convert the DataFrame to an RDD and filter out rows with invalid paths
def is_valid_image_path(row):
    image_path = row.image_path
    try:
        # Attempt to open the image
        with Image.open(image_path):
            return True
    except:
        # Image could not be opened or path is invalid
        return False

valid_images_rdd = cleaned_df.rdd.filter(is_valid_image_path)

# Convert the RDD back to a DataFrame
cleaned_df = spark.createDataFrame(valid_images_rdd, schema=combined_data_df1.schema)
```

```
[Stage 52:>                                                 (0 + 3) / 3]
```

In [27]:
```
# Step 4: Verify labels
# Ensure that the labels are within the expected range ('NORMAL', 'PNEUMONIA', 'COVID')
expected_labels = ['NORMAL', 'PNEUMONIA', 'COVID']
cleaned_df = cleaned_df.filter(cleaned_df['label'].isin(expected_labels))
```

CNN Model Building

1. Setting up Environment
2. Data Loading and Conversion
3. Data Splitting
4. Data Preprocessing
5. Model Building
6. Model Training
7. Model Evaluation



File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
# Initialize Spark session
spark = SparkSession.builder.appName("Covid19Detection").getOrCreate()
# Load the data into a Spark DataFrame
combined_df = spark.read.csv("/Users/ramyasritellakula/Documents/Covid_19_detection/result/cleaned_data.csv", header=True, infers

# Convert Spark DataFrame to Pandas DataFrame
combined_df_pandas = combined_df.toPandas()

# Split data into features (image paths) and labels
X = combined_df_pandas['image_path']
y = combined_df_pandas['label']

# Sample train data to get exactly 2000 samples
train_sample_size = 2000
train_data = combined_df_pandas.sample(n=train_sample_size, random_state=42)
X_train = train_data['image_path']
y_train = train_data['label']

# Get remaining data for test and validation
remaining_data = combined_df_pandas.drop(train_data.index)

# Split remaining data into test and validation sets (80% test, 20% validation)
X_remaining = remaining_data['image_path']
y_remaining = remaining_data['label']
X_test, X_val, y_test, y_val = train_test_split(X_remaining, y_remaining, test_size=0.2, random_state=42)

# Print the sizes and label counts
print(f"Size of training set: {len(X_train)} samples")
print(f"Size of validation set: {len(X_val)} samples")
print(f"Size of test set: {len(X_test)} samples")

print("\nLabel counts in training set:")
print(y_train.value_counts())

print("\nLabel counts in validation set:")
print(y_val.value_counts())

print("\nLabel counts in test set:")
print(y_test.value_counts())


# Define image dimensions and batch size
img_height, img_width = 224, 224
batch_size = 32

# Create image data generators
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

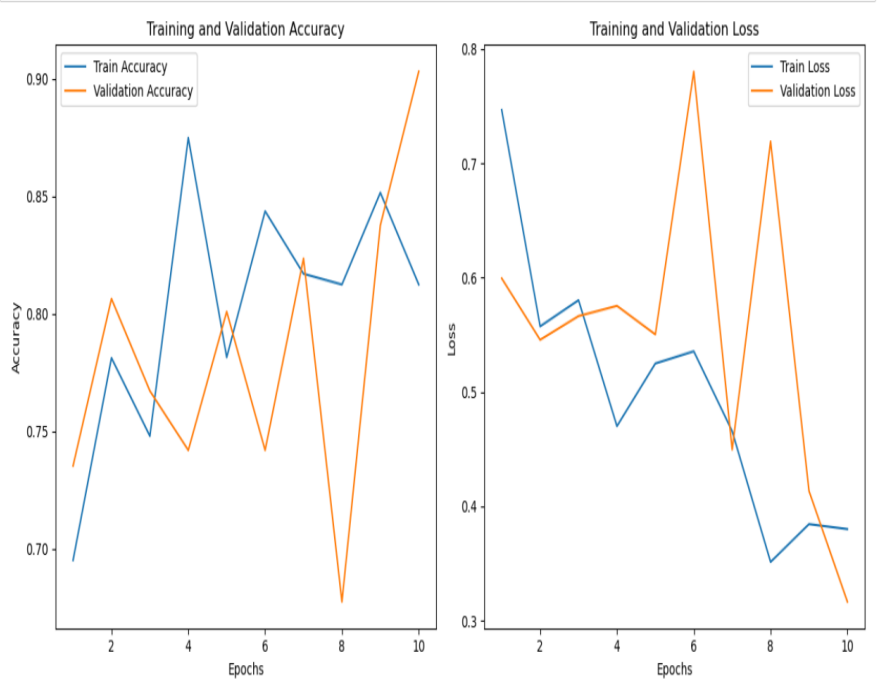
```
    x_col="image_path",
    y_col="label",
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_dataframe(
    dataframe=pd.DataFrame({'image_path': X_test, 'label': y_test}),
    x_col="image_path",
    y_col="label",
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

# Build the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Define model checkpoint to save best model during training
checkpoint = ModelCheckpoint("best_model.keras", monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    validation_data=val_generator,
    validation_steps=val_generator.samples // batch_size,
    epochs=10,
    callbacks=[checkpoint]
)
```

Accuracy and Loss Graphs

- *For 2000 Images*

```
# Evaluate the model
test_loss, test_accuracy = model.evaluate(test_generator)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

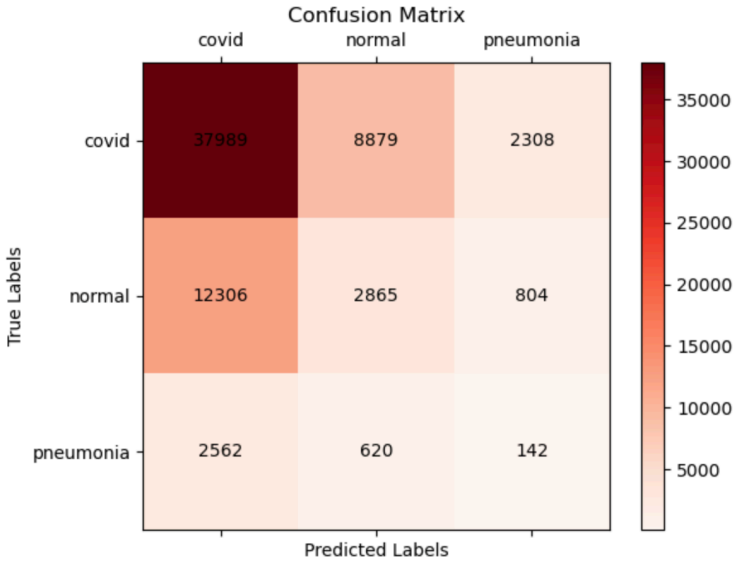
```
        [[{{node IteratorGetNext}}]]

62/62 ━━━━━━━━━━━━━━━━━━━━ 0s 433ms/step - accuracy: 0.8516 - loss: 0.4009
Epoch 9: val_accuracy improved from 0.82362 to 0.83772, saving model to best_model.keras
62/62 ━━━━━━━━━━━━━━━━━━━━ 4484s 73s/step - accuracy: 0.8516 - loss: 0.4006 - val_accuracy: 0.8377 - val_loss: 0.4135
Epoch 10/10
 1/62 ━━━━━━━━━━━━━━━━━━━━ 24s 402ms/step - accuracy: 0.8125 - loss: 0.3802
Epoch 10: val_accuracy improved from 0.83772 to 0.90323, saving model to best_model.keras
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8125 - loss: 0.3802 - val_accuracy: 0.9032 - val_loss: 0.3167

2024-05-05 07:46:57.157449: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
        [[{{node IteratorGetNext}}]]
2024-05-05 07:46:57.271002: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
        [[{{node IteratorGetNext}}]]

2140/2140 ━━━━━━━━━━━━━━━━━━━━ 3196s 1s/step - accuracy: 0.8322 - loss: 0.4226
Test Loss: 0.4207819998264313
Test Accuracy: 0.833734929561615
```



Confusion Matrix

Classification Report:

	precision	recall	f1-score	support
COVID	0.72	0.77	0.74	49176
NORMAL	0.23	0.18	0.20	15975
PNEUMONIA	0.04	0.04	0.04	3324
accuracy			0.60	68475
macro avg	0.33	0.33	0.33	68475
weighted avg	0.57	0.60	0.58	68475

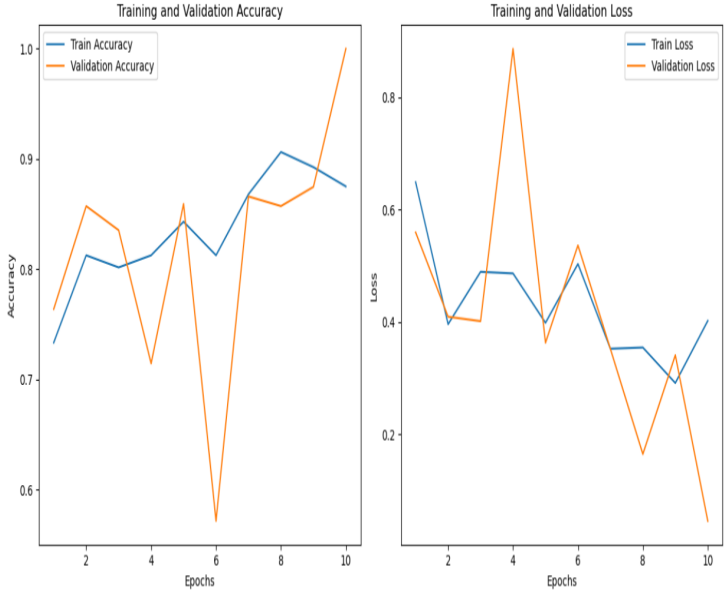


Accuracy and Loss Graphs

- *For 5000 Images*

```
# Evaluate the model
test_loss, test_accuracy = model.evaluate(test_generator)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

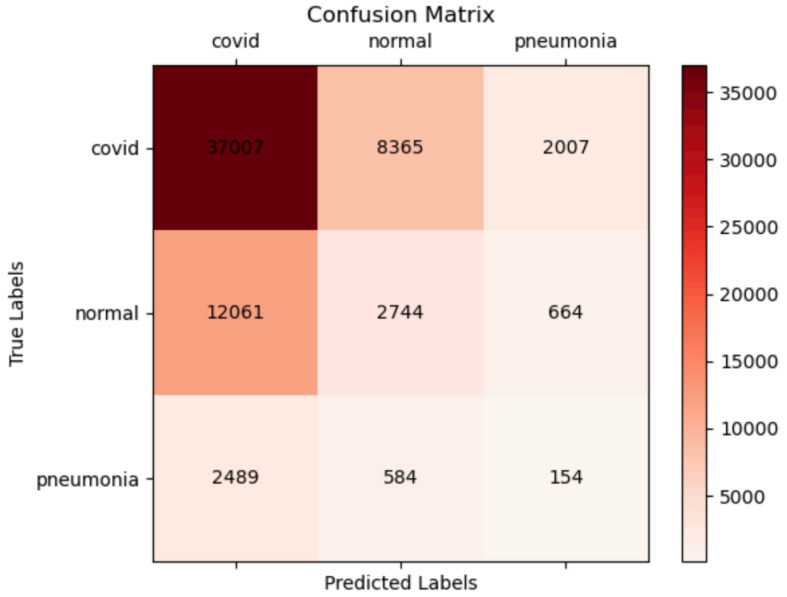
```
        [[{{node IteratorGetNext}}]]

156/156 ━━━━━━━━━━━━━━━━━━━━ 0s 505ms/step - accuracy: 0.8932 - loss: 0.2935
Epoch 9: val_accuracy improved from 0.86592 to 0.87458, saving model to best_model.keras
156/156 ━━━━━━━━━━━━━━━━━━━━ 198s 1s/step - accuracy: 0.8932 - loss: 0.2935 - val_accuracy: 0.8746 - val_loss: 0.3413
Epoch 10/10
  1/156 ━━━━━━━━━━━━━━━━━━━━ 1:17 499ms/step - accuracy: 0.8750 - loss: 0.4025
Epoch 10: val_accuracy improved from 0.87458 to 1.00000, saving model to best_model.keras
156/156 ━━━━━━━━━━━━━━━━━━━━ 1s 763us/step - accuracy: 0.8750 - loss: 0.4025 - val_accuracy: 1.0000 - val_loss: 0.0461

2024-05-05 09:23:00.972529: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
        [[{{node IteratorGetNext}}]]
2024-05-05 09:23:01.011492: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
        [[{{node IteratorGetNext}}]]

2065/2065 ━━━━━━━━━━━━━━━━━━━━ 479s 232ms/step - accuracy: 0.8768 - loss: 0.3492
Test Loss: 0.3532032370567322
Test Accuracy: 0.875838041305542
```



Confusion Matrix

Classification Report:

	precision	recall	f1-score	support
COVID	0.72	0.78	0.75	47379
NORMAL	0.24	0.18	0.20	15469
PNEUMONIA	0.05	0.04	0.04	3227
accuracy			0.60	66075
macro avg	0.33	0.33	0.33	66075
weighted avg	0.57	0.60	0.59	66075



Accuracy and Loss Graphs

- *For 20000 Images*

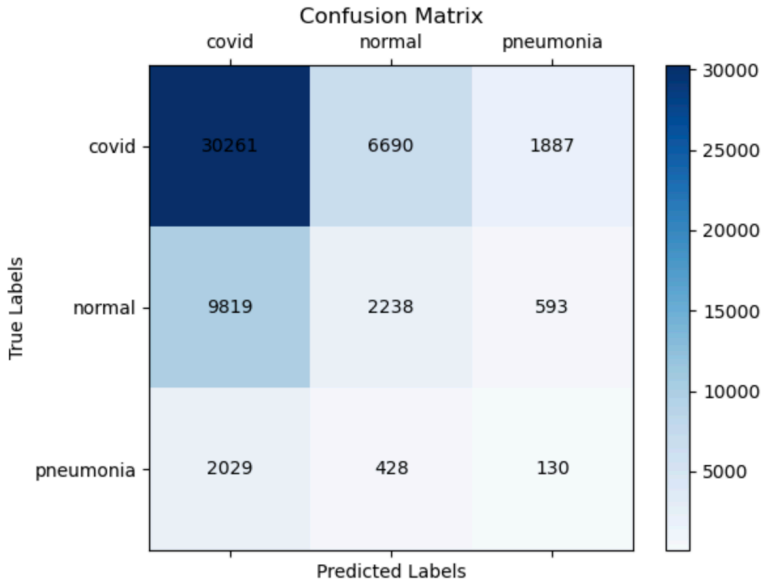
```
# Evaluate the model
test_loss, test_accuracy = model.evaluate(test_generator)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

```
625/625 ━━━━━━━━━━━━━━━━━━━━ 0s 609ms/step - accuracy: 0.9183 - loss: 0.2231
Epoch 9: val_accuracy did not improve from 1.00000
625/625 ━━━━━━━━━━━━━━━━━━━━ 491s 784ms/step - accuracy: 0.9183 - loss: 0.2231 - val_accuracy: 0.9037 - val_loss: 0.2780
Epoch 10/10

Epoch 10: val_accuracy did not improve from 1.00000
625/625 ━━━━━━━━━━━━━━━━━━━━ 0s 141us/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 - val_accuracy: 0.8000 - val_loss: 0.571
8

2024-05-05 10:35:24.589261: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OU
T_OF_RANGE: End of sequence
        [[{{node IteratorGetNext}}]]
2024-05-05 10:35:24.669502: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OU
T_OF_RANGE: End of sequence
        [[{{node IteratorGetNext}}]]

1690/1690 ━━━━━━━━━━━━━━━━━━━━ 435s 257ms/step - accuracy: 0.9019 - loss: 0.2853
Test Loss: 0.2807624936103821
Test Accuracy: 0.9038557410240173
```

Confusion Matrix

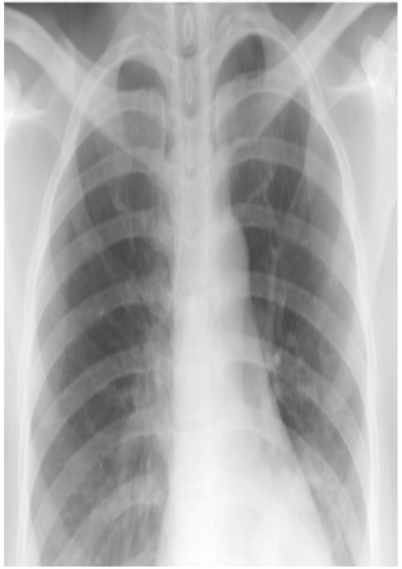
Classification Report:

	precision	recall	f1-score	support
COVID	0.72	0.78	0.75	38838
NORMAL	0.24	0.18	0.21	12650
PNEUMONIA	0.04	0.04	0.04	2587
accuracy			0.60	54075
macro avg	0.34	0.33	0.33	54075
weighted avg	0.58	0.60	0.59	54075

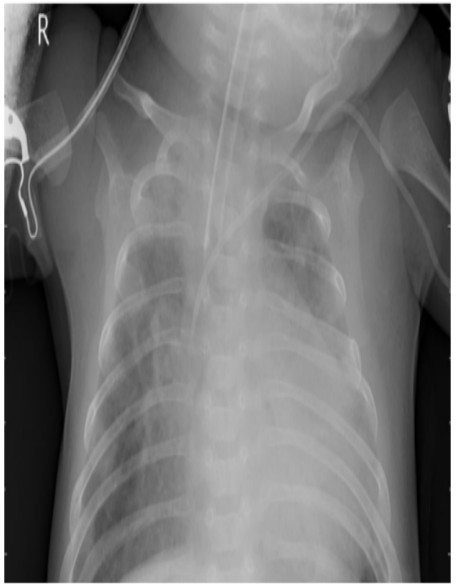


Predicting Unlabelled Images

Predicted: normal
Probabilities: [[7.9842085e-01 2.0121607e-01 3.6314529e-04]]



Predicted: covid
Probabilities: [[3.6892667e-04 7.8891553e-03 9.9174190e-01]]



Predicted: pneumonia
Probabilities: [[0.0086776 0.9832706 0.00805176]]

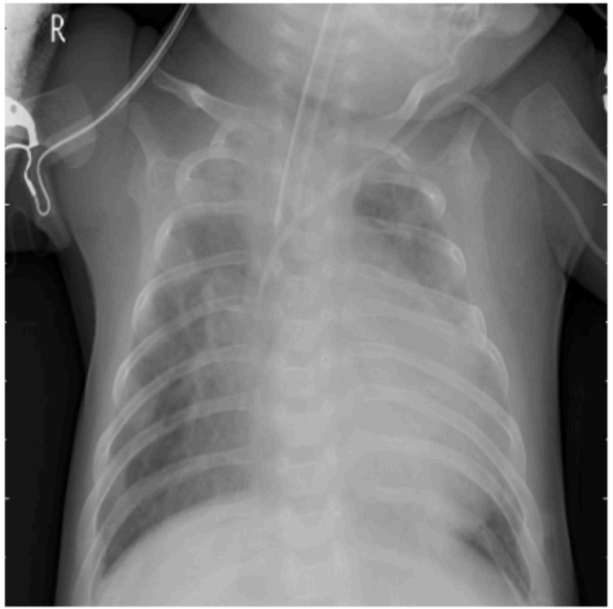
Predicting Covid Status of Sample Images

```
# Ask the user for an image path and make a prediction
image_path = input("Enter path for chest X-ray image: ")
predict_from_image(image_path)
```

```
Enter path for chest X-ray image:  /Users/ramyasritellakula/Documents/Covid_19_detection/sample/person1_bacteria_1.jpeg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
Prediction of our model: Negative for Covid-19
```

Predicting Covid Status of Sample Images

```
Enter path for chest X-ray image:  /Users/ramyasritellakula/Documents/Covid_19_detection/sample/person1_virus_6.jpeg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
Prediction of our model: Positive for Covid-19
```



Observations

- SET1 (2000 images): Achieved a test loss of 0.4208 and test accuracy of 83.37%, indicating decent model performance.
- SET2 (5000 images): Improved results with a test loss of 0.3532 and test accuracy of 87.58%, suggesting the model benefits from a larger dataset.
- SET3 (20,000 images): Best performance with the largest dataset, achieving a test loss of 0.2808 and test accuracy of 90.39%. This showcases the model's ability to generalize effectively with more data.
- Larger training datasets enhance model accuracy and reduce loss in Covid-19 detection using CNNs, with a substantial increase in predictive precision as the dataset size increases.

Conclusion and Lessons Learnt from the Project

- The model trained with SET3 (20,000 images) exhibited the best performance(Test Accuracy: 90.39% Test Loss: 0.2808)
- As the size of the training set increased, there was a clear improvement in the model's performance.This improvement was reflected in lower test loss and higher test accuracy.
- If we have disturbed networks, modeling accuracy would have been more for the dataset.
- We tried mongoldb and amazon web services as an alternative for data storage.
- First, reading data from Mongo db and cloning data of 31GB to amazon web services was challenging.
- Selecting local disk to store and reading out using spark configuration made our way easy, minimizing errors

OUR TEAM AND ROLES

Lakshmi Deepika Karlapudi

- Data Acquisition
- Preprocessing

Manasa Jagati

- Deployment
- Integration

Tellakula Ramya Sri

- Model Development
- Documentation

ANY QUESTIONS ?