Title:

Using CNN recognize the objects in the given images For the same set of images train using the other networks and determine best network model

Abstract:

This research develops a convolutional neural network (CNN) model for the classification of chest X-ray images to detect COVID-19. With the ongoing need for rapid and accurate COVID-19 diagnostics, the proposed model aims to aid healthcare providers in identifying COVID-19 cases through chest X-ray imagery. By leveraging data augmentation techniques, the model addresses dataset imbalance and minimizes overfitting. Key metrics, such as accuracy and confusion matrices, assess model performance and its reliability in identifying COVID-19 in radiographic images.

Introduction:

COVID-19 is a respiratory illness that has posed significant diagnostic challenges globally. Chest X-rays are commonly used in diagnostic settings, but manually analyzing these images requires expertise and time. Leveraging machine learning, specifically CNNs, offers an efficient solution for automating the COVID-19 diagnosis process. This research applies a CNN model to classify chest X-ray images as either COVID-19 positive or normal, aiming to support healthcare facilities in accurate and timely diagnoses.

Literature review:

Several studies highlight CNNs' effectiveness in medical imaging, including detecting COVID-19 from chest X-rays. Previous research demonstrates that CNNs can extract relevant features from radiographic images, which assists in identifying patterns indicative of disease. Many studies incorporate techniques like transfer learning, data augmentation, and dropout to improve model performance. However, challenges such as dataset imbalance, image quality variability, and the need to balance false positives and false negatives remain key considerations in medical AI applications.

Methodology:

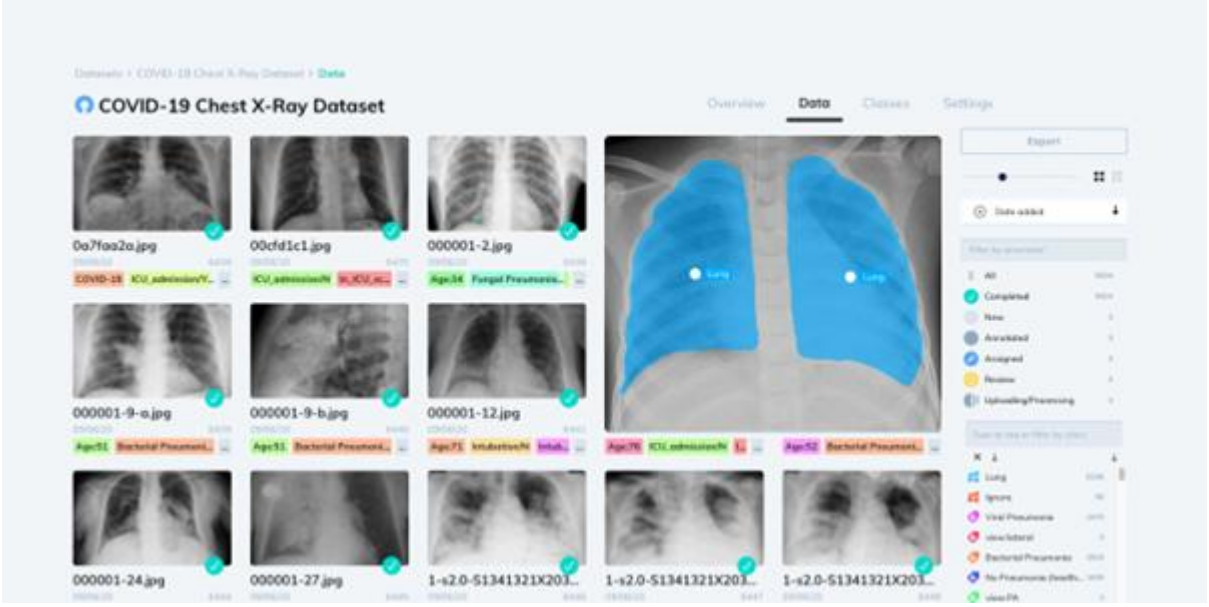
1. **Data Collection and Preprocessing:** The dataset includes labeled chest X-ray images, categorized as COVID-19 positive or normal. Images are resized to a uniform 150x150 pixels and converted to grayscale to reduce processing complexity while preserving essential features.
2. **Data Augmentation:** To address dataset imbalance and prevent overfitting, transformations including random rotations, zooms, flips, and shifts are applied during training. This augmentation increases training set diversity without requiring more labeled data.
3. **CNN Architecture:** The CNN model includes multiple convolutional layers with ReLU activations, batch normalization, and max pooling. Dropout layers are added to reduce overfitting, and the final sigmoid-activated layer outputs binary classification results for COVID-19 positive or normal.
4. **Training and Optimization:** The model is compiled with binary cross-entropy loss and trained using the RMSprop optimizer. Learning rate reduction is implemented to optimize training by adjusting the learning rate when validation accuracy plateaus.
5. **Evaluation Metrics:** Model performance is evaluated based on accuracy, confusion matrix, and classification reports. Training and validation metrics, including accuracy and loss trends, are visualized to assess progress over epochs.

Workflow:

1. **Data Loading:** X-ray images are loaded from specific directories, and labels are assigned (0 for COVID-19, 1 for normal).
2. **Data Augmentation:** The ImageDataGenerator library augments the training data through transformations.
3. **Model Training:** The CNN is trained using the augmented data, with validation accuracy monitored to detect overfitting.
4. **Model Evaluation:** The model's effectiveness on the test set is analyzed through accuracy, loss, and confusion matrix metrics.
5. **Visualization:** Examples of correct and incorrect predictions are visualized to provide insights into the model's strengths and areas for improvement.

Results and Decision:

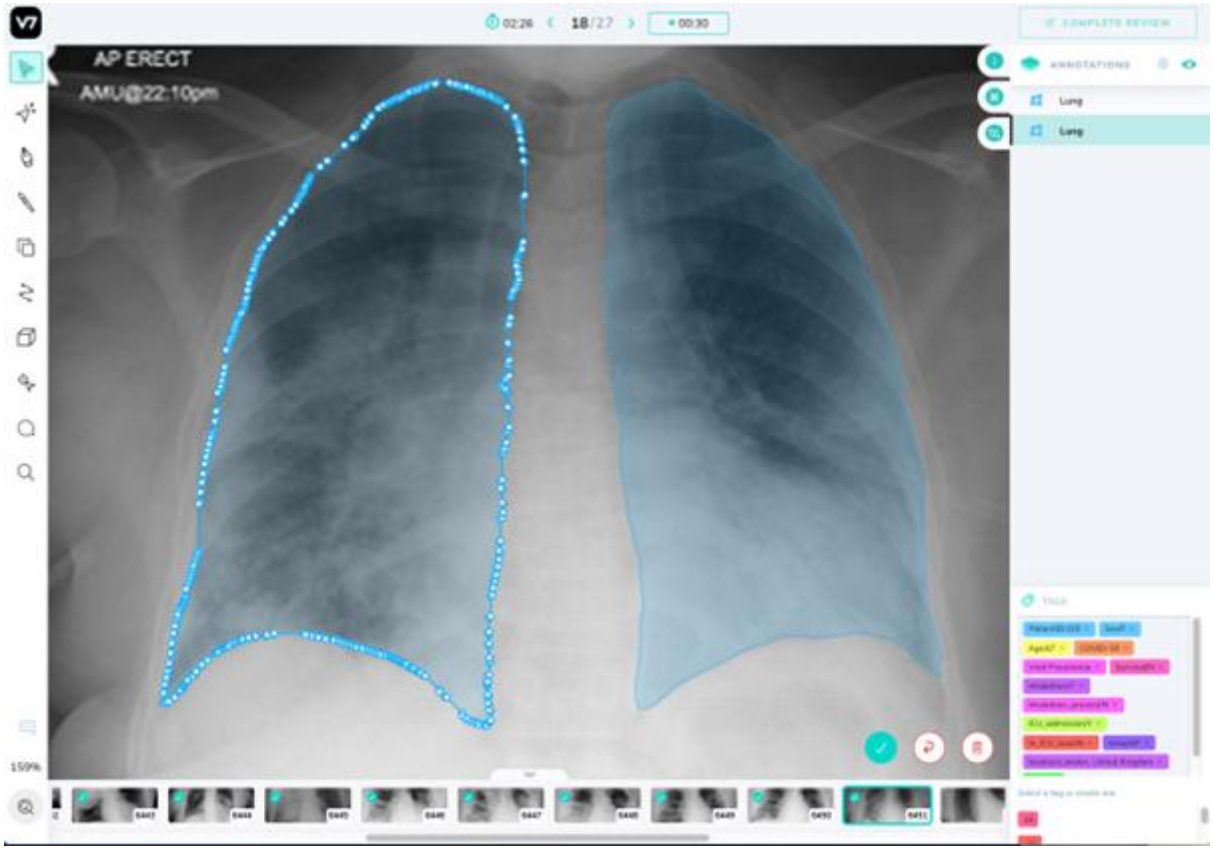
The CNN model was selected for this research due to its success in recognizing complex image patterns and its utility in medical imaging. The chosen CNN architecture involved several convolutional layers with batch normalization and dropout to manage overfitting and ensure the model's generalizability. Images were resized to 150x150 pixels and processed in grayscale, reducing computational load without compromising the model's accuracy in recognizing COVID-19.



Each image contains:

- Two "Lung" segmentation masks (rendered as polygons, including the posterior region behind the heart).
- A tag for the type of pneumonia (viral, bacterial, fungal, healthy/none)
- If the patient has COVID-19, additional tags stating age, sex, temperature, location, intubation status, ICU admission, and patient outcome.

Lung annotations are polygons following pixel-level boundaries. These can be exported as COCO, VOC, or Darwin JSON formats. Each annotation file contains a URL to the original full resolution image, as well as a reduced size thumbnail.



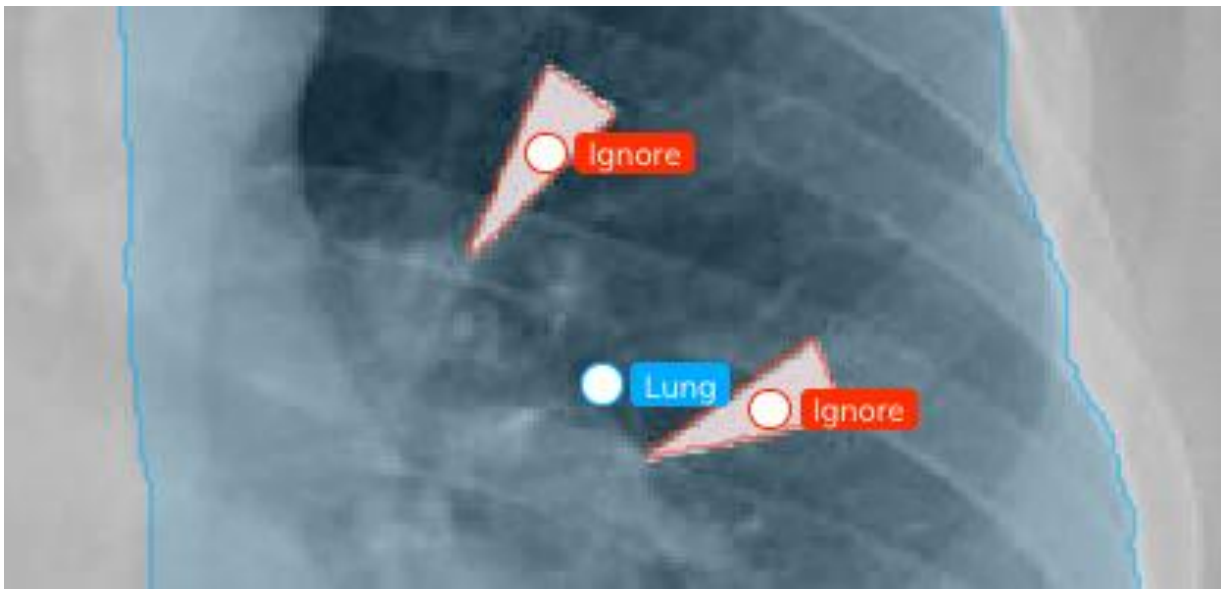
LUNG SEGMENTATION NOTES:

Lung segmentations in this dataset include most of the heart, revealing lung opacities behind the heart which may be relevant for assessing the severity of viral pneumonia. Uniformly shaped lungs also de-couples the shape and content within the left lung from the size of the heart.

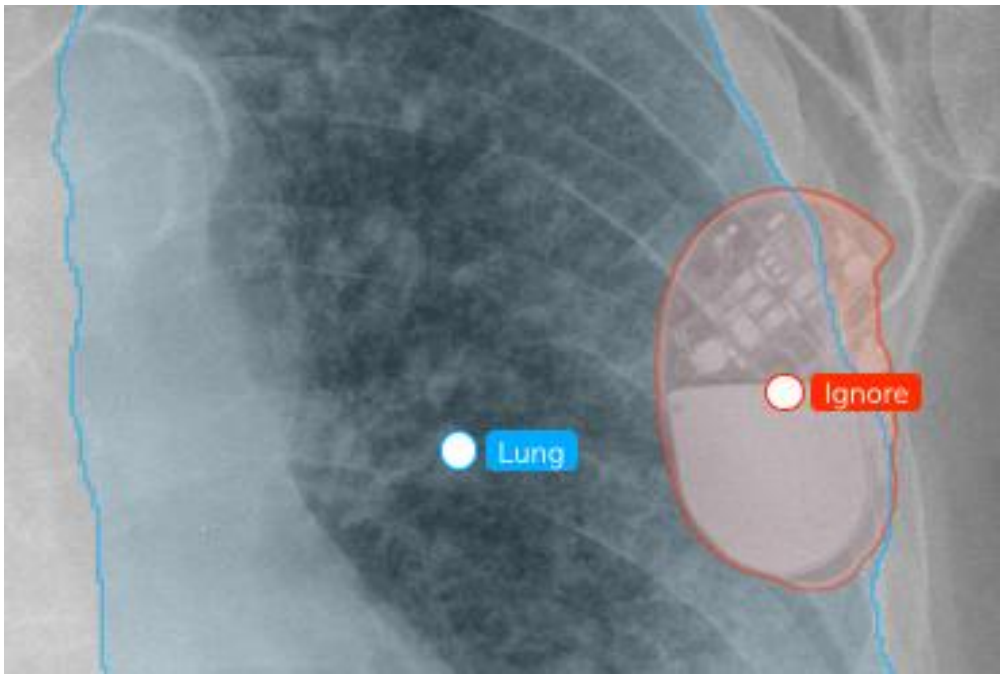
The lower-most part of the lungs is defined by the extent of the diaphragm, where visible. If the back of the lungs is clearly visible through the diaphragm it is also included up until the lower-most visible part of the lungs.

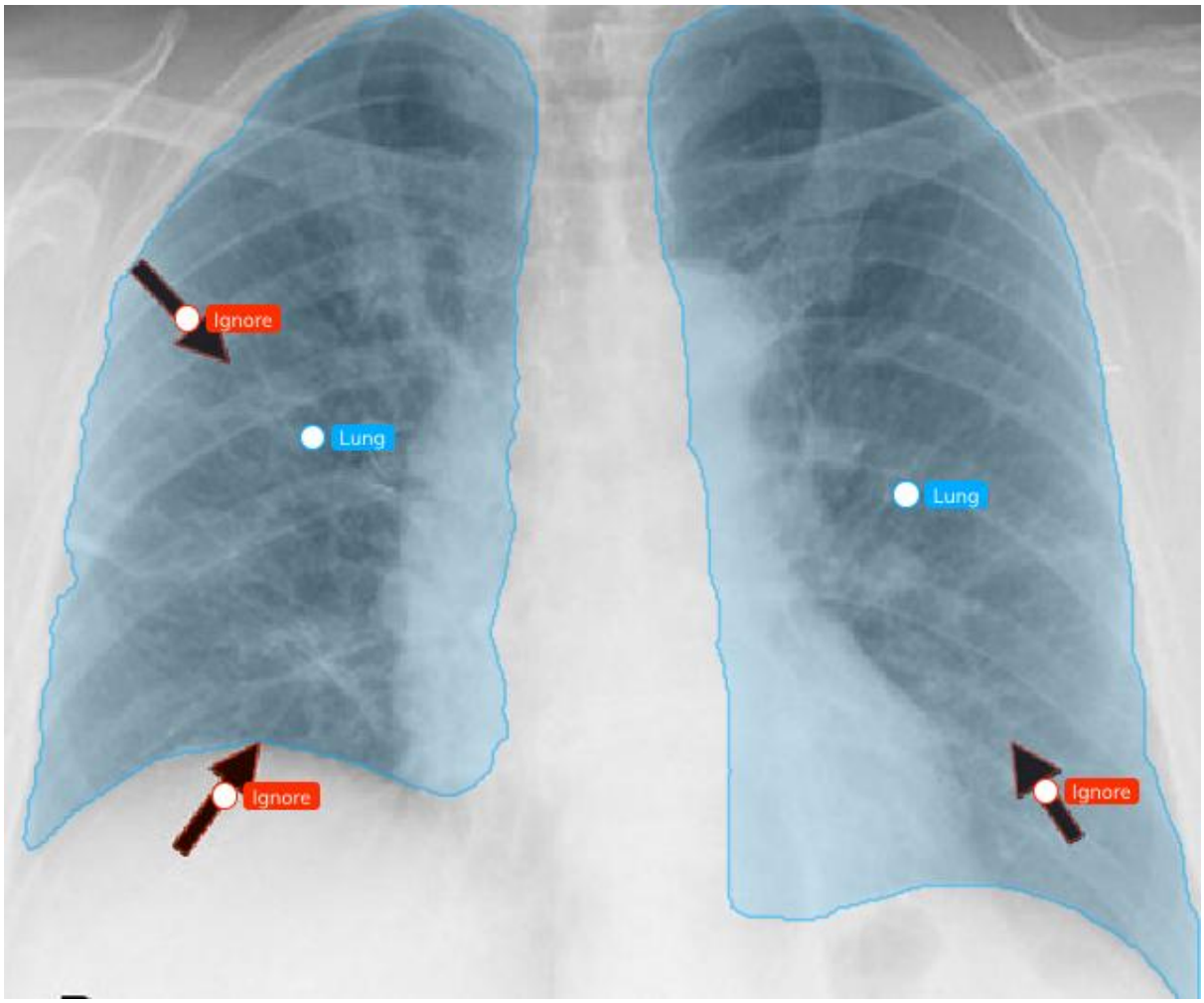
Lung segmentations were performed by human annotators using Auto-Annotate, adjusted, and reviewed by humans.

Other important notes:

- Image resolutions, sources, and orientations vary across the dataset, with the largest image being 5600x4700 and smallest being 156x156. You may sort images by dimensions on Darwin to exclude those below a threshold.
- Lateral x-rays do not contain lung segmentations. They have classification tags, but should be ignored if you are working with detection-based networks.
- There are 63 axial CT scan slices left un-labelled with masks (although they contain tags) as a way of maintaining integrity to one of the source datasets. We encourage discarding these when performing x-ray analysis.
- Portable x-ray images are of significant lower quality than others. Be aware that they correlate highly with severe conditions. Classification models will bias portable x-ray images with diseases like COVID-19.
- Medical instruments like pacemakers and markup that overlap the lungs are masked with an "Ignore" class. We encourage masking these out when performing lung analysis as they correlated strongly with sick patients. Intubation instruments are not removed if smaller/thinner than 1cm.
-


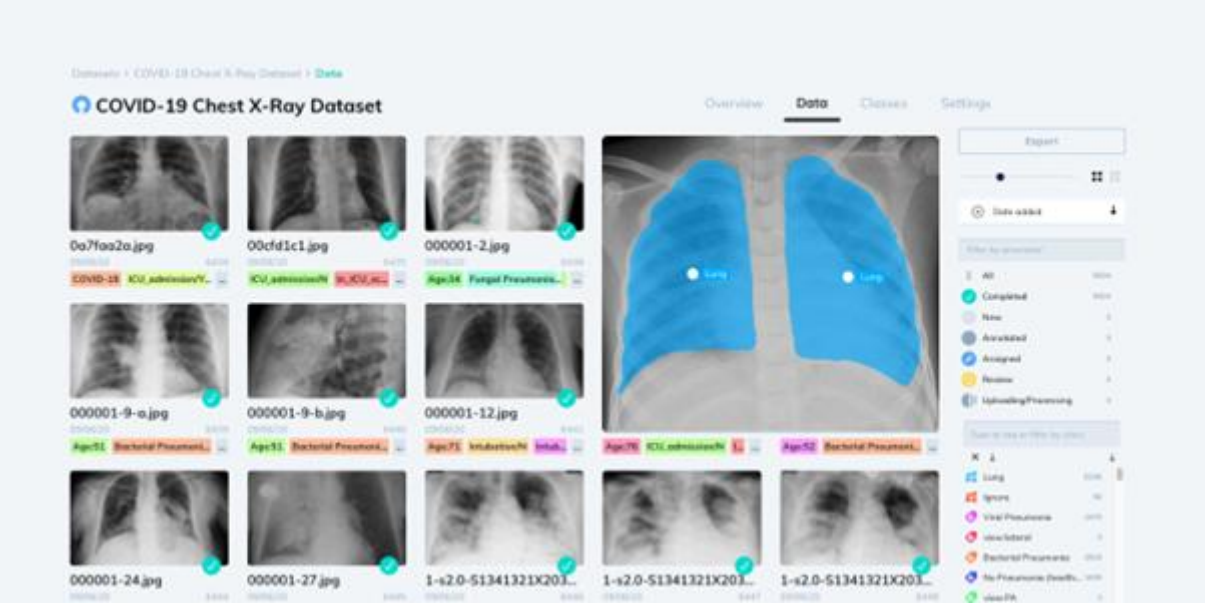
-





You may also use the Ignore class to filter out images with occluding markups or large medical instruments.

You can browse the available images and filter them by tag or annotation class using the right-sidebar as seen below.



Each image contains:

- Two "Lung" segmentation masks (rendered as polygons, including the posterior region behind the heart).
- A tag for the type of pneumonia (viral, bacterial, fungal, healthy/none)
- If the patient has COVID-19, additional tags stating age, sex, temperature, location, intubation status, ICU admission, and patient outcome.

Lung annotations are polygons following pixel-level boundaries. These can be exported as COCO, VOC, or Darwin JSON formats. Each annotation file contains a URL to the original full resolution image, as well as a reduced size thumbnail.

Code:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
for dirname, _, filenames in os.walk('C:\\Users\\davis\\Music\\chest_xray'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
import matplotlib.pyplot as plt
import seaborn as sns
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout ,
BatchNormalization
from tensorflow.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


def get_training_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img),
cv2.IMREAD_GRAYSCALE)
                if img_arr is not None:  correctly
                    resized_arr = cv2.resize(img_arr, (img_size, img_size))
                    data.append([resized_arr, class_num])
                else:
                    print(f"Image {img} could not be read.")
```

```
except Exception as e:
                print(f"Error processing image {img}: {e}")
    return np.array(data, dtype=object)

train = get_training_data('C:\\Users\\davis\\Music\\chest_xray\\train')
test = get_training_data('C:\\Users\\davis\\Music\\chest_xray\\test')
val = get_training_data('C:\\Users\\davis\\Music\\chest_xray\\val')

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
l = []
for i in train:
    if i[1] == 0:
        l.append("Covid")
    else:
        l.append("Normal")

df = pd.DataFrame(l, columns=["Label"])

sns.set_style('darkgrid')
plt.figure(figsize=(8, 5))
sns.countplot(x="Label", data=df)
plt.title("Count of COVID and Normal Cases")
plt.xlabel("Condition")
plt.ylabel("Count")
plt.show()

plt.figure(figsize = (5,5))
plt.imshow(train[0][0], cmap='gist_heat')
plt.title(labels[train[0][1]])

plt.figure(figsize = (5,5))
plt.imshow(train[-1][0], cmap='gist_heat')
plt.title(labels[train[-1][1]])
```

```
x_train = []
y_train = []
x_val = []
y_val = []
x_test = []
y_test = []

for feature, label in train:
    x_train.append(feature)
    y_train.append(label)

for feature, label in test:
    x_test.append(feature)
    y_test.append(label)

for feature, label in val:
    x_val.append(feature)
    y_val.append(label)

# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test = np.array(x_test) / 255

# resize data for deep learning
x_train = x_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

x_val = x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)

x_test = x_test.reshape(-1, img_size, img_size, 1)
y_test = np.array(y_test)
```

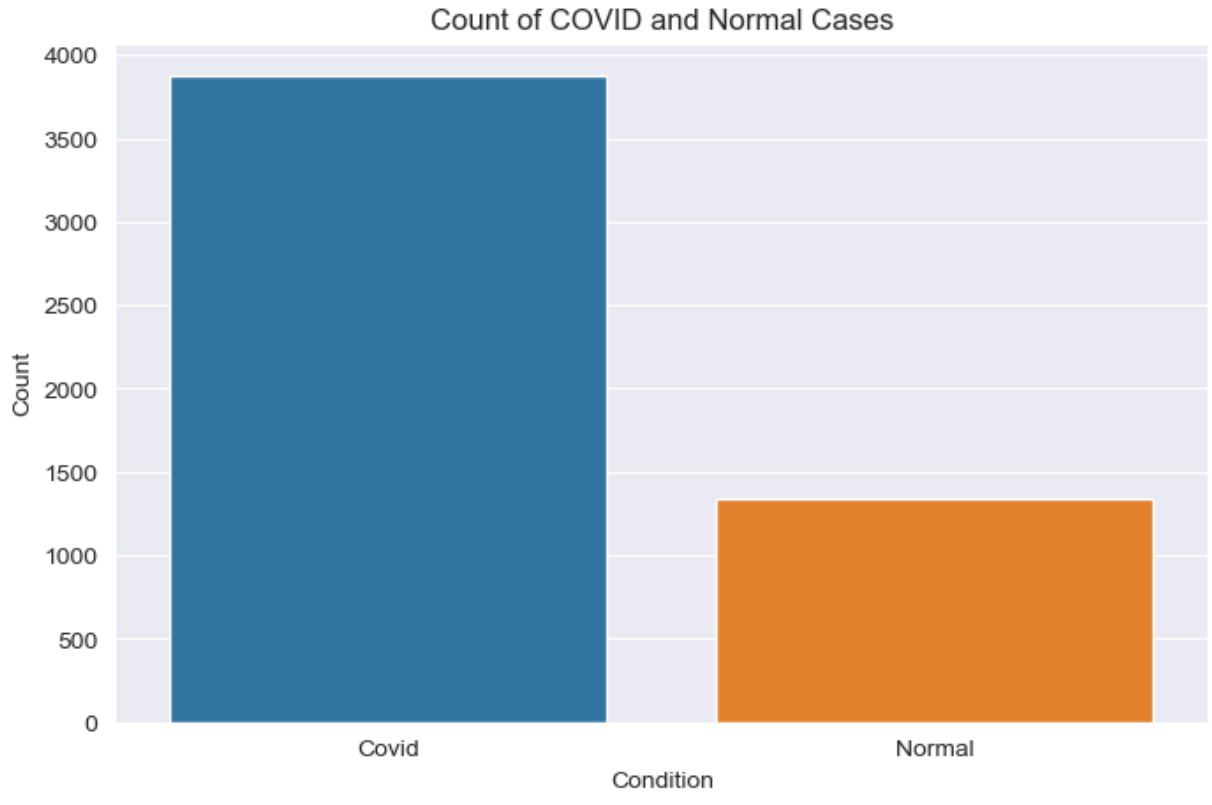
```
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the
dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range = 30,  # randomly rotate images in the range
(degrees, 0 to 180)
        zoom_range = 0.2, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally
(fraction of total width)
        height_shift_range=0.1,  # randomly shift images vertically
(fraction of total height)
        horizontal_flip = True,  # randomly flip images
        vertical_flip=False)  # randomly flip images
datagen.fit(x_train)


model = Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation =
'relu' , input_shape = (150,150,1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation =
'relu'))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation =
'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation =
'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
```

```
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Flatten())

model.add(Dense(units = 128 , activation = 'relu'))

model.add(Dropout(0.2))

model.add(Dense(units = 1 , activation = 'sigmoid'))

model.compile(optimizer = "rmsprop" , loss = 'binary_crossentropy' , metrics
= ['accuracy'])

model.summary()


learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience
= 2, verbose=1,factor=0.3, min_lr=0.000001)


history = model.fit(datagen.flow(x_train,y_train, batch_size = 32) ,epochs =
12,

                    validation_data = datagen.flow(x_val, y_val) ,callbacks
= [learning_rate_reduction])


print("Loss of the model is - " , model.evaluate(x_test,y_test)[0])

print("Accuracy of the model is - " , model.evaluate(x_test,y_test)[1]*100 ,
"%")


epochs = [i for i in range(12)]

fig , ax = plt.subplots(1,2)

train_acc = history.history['accuracy']

train_loss = history.history['loss']

val_acc = history.history['val_accuracy']

val_loss = history.history['val_loss']

fig.set_size_inches(20,10)


ax[0].plot(epochs , train_acc , 'go-' , label = 'Training Accuracy')

ax[0].plot(epochs , val_acc , 'ro-' , label = 'Validation Accuracy')

ax[0].set_title('Training & Validation Accuracy')

ax[0].legend()

ax[0].set_xlabel("Epochs")

ax[0].set_ylabel("Accuracy")
```

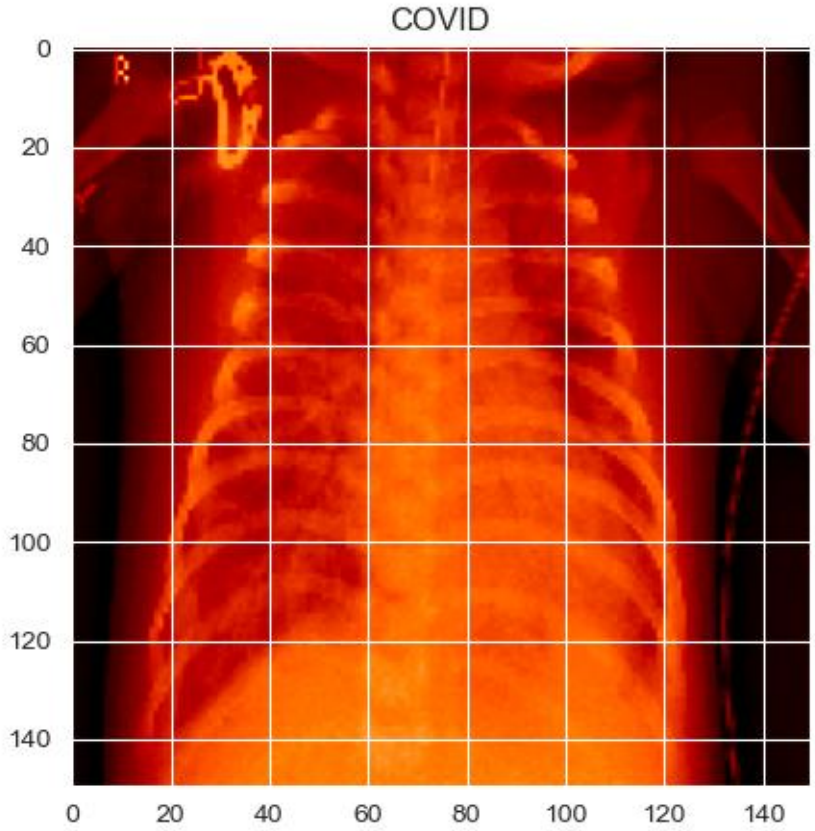
```
ax[1].plot(epochs , train_loss , 'g-o' , label = 'Training Loss')
ax[1].plot(epochs , val_loss , 'r-o' , label = 'Validation Loss')
ax[1].set_title('Testing Accuracy & Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Training & Validation Loss")
plt.show()


predictions_prob = model.predict(x_test)
predictions = (predictions_prob > 0.5).astype(int)  # Set a threshold of 0.5
print(predictions[:15])


print(classification_report(y_test, predictions, target_names = ['Covid
(Class 0)','Normal (Class 1)']))


cm = confusion_matrix(y_test,predictions)
cm
cm = pd.DataFrame(cm , index = ['0','1'] , columns = ['0','1'])
plt.figure(figsize = (10,10))
sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth = 1 , annot =
True, fmt='',xticklabels = labels,yticklabels = labels)


correct = np.nonzero(predictions == y_test)[0]
incorrect = np.nonzero(predictions != y_test)[0]


i = 0
for c in correct[:6]:
    plt.subplot(3,2,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[c].reshape(150,150), cmap="gray",
interpolation='none')
    plt.title("Predicted Class {},Actual Class {}".format(predictions[c],
y_test[c]))
    plt.tight_layout()
    i += 1
```

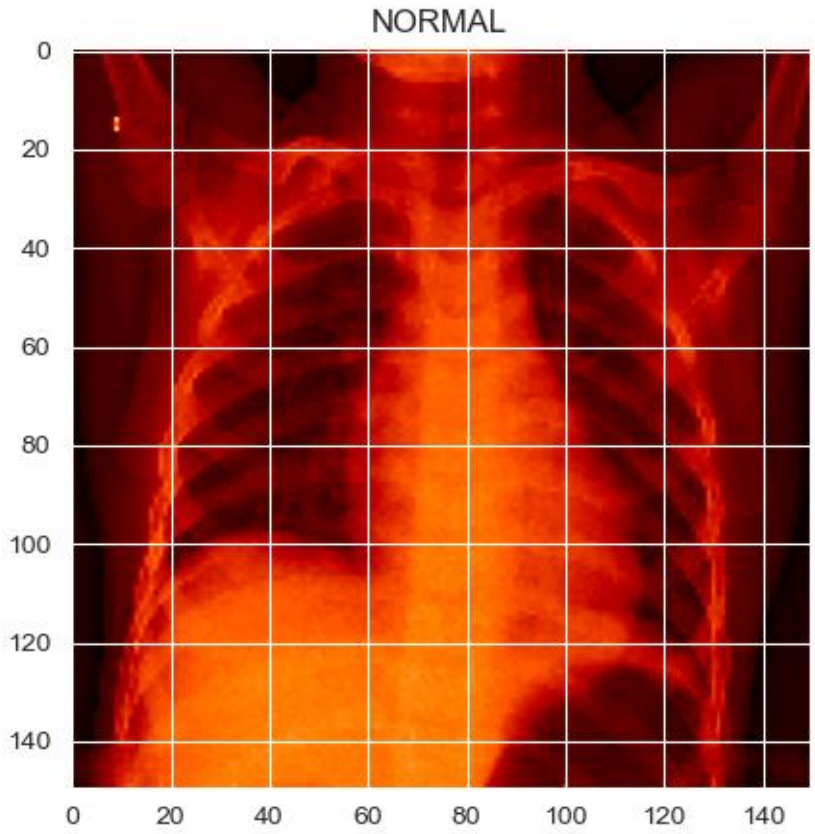
Output:

```
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person984_bacteria_2911.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person985_bacteria_2912.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person986_bacteria_2913.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person987_bacteria_2914.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person988_bacteria_2915.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person988_virus_1666.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person989_virus_1667.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person98_virus_182.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person990_bacteria_2917.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person991_bacteria_2918.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person991_virus_1669.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person992_bacteria_2919.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person992_bacteria_2920.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person992_virus_1670.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person993_bacteria_2921.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person993_virus_1671.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person994_bacteria_2922.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person994_virus_1672.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person995_bacteria_2923.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person995_virus_1676.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person996_bacteria_2924.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person996_virus_1677.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person997_bacteria_2926.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person997_virus_1678.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person998_bacteria_2927.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person998_bacteria_2928.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person99_virus_183.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person9_bacteria_38.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person9_bacteria_39.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person9_bacteria_40.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\train\PNEUMONIA\._person9_bacteria_41.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\._.DS_Store
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\NORMAL\._.DS_Store
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\NORMAL\._NORMAL2-IM-1427-0001.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\NORMAL\._NORMAL2-IM-1430-0001.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\NORMAL\._NORMAL2-IM-1431-0001.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\NORMAL\._NORMAL2-IM-1436-0001.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\NORMAL\._NORMAL2-IM-1437-0001.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\NORMAL\._NORMAL2-IM-1438-0001.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\NORMAL\._NORMAL2-IM-1440-0001.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\NORMAL\._NORMAL2-IM-1442-0001.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\PNEUMONIA\._.DS_Store
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\PNEUMONIA\._person1946_bacteria_4874.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\PNEUMONIA\._person1946_bacteria_4875.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\PNEUMONIA\._person1947_bacteria_4876.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\PNEUMONIA\._person1949_bacteria_4880.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\PNEUMONIA\._person1950_bacteria_4881.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\PNEUMONIA\._person1951_bacteria_4882.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\PNEUMONIA\._person1952_bacteria_4883.jpeg
C:\Users\davis\Music\chest_xray\__MACOSX\chest_xray\val\PNEUMONIA\._person1954_bacteria_4886.jpeg
```



```
Text(0.5, 1.0, 'NORMAL')
```





Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 32)	320
batch_normalization (BatchNormalization)	(None, 150, 150, 32)	128
max_pooling2d (MaxPooling2D)	(None, 75, 75, 32)	0
conv2d_1 (Conv2D)	(None, 75, 75, 64)	18,496
dropout (Dropout)	(None, 75, 75, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 75, 75, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 38, 38, 64)	0
conv2d_2 (Conv2D)	(None, 38, 38, 64)	36,928
batch_normalization_2 (BatchNormalization)	(None, 38, 38, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 19, 19, 64)	0
conv2d_3 (Conv2D)	(None, 19, 19, 128)	73,856
dropout_1 (Dropout)	(None, 19, 19, 128)	0
batch_normalization_3 (BatchNormalization)	(None, 19, 19, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 10, 10, 128)	0
conv2d_4 (Conv2D)	(None, 10, 10, 256)	295,168

Total params: 1,246,401 (4.75 MB)

Trainable params: 1,245,313 (4.75 MB)

Non-trainable params: 1,088 (4.25 KB)

Epoch 1/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **102s** 588ms/step - accuracy: 0.8085 - loss: 1.3302 - val_accuracy: 0.5000 - val_loss: 34.2184 - learning_rate: 0.0010

Epoch 2/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **98s** 599ms/step - accuracy: 0.9008 - loss: 0.2662 - val_accuracy: 0.5000 - val_loss: 48.9144 - learning_rate: 0.0010

Epoch 3/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **0s** 545ms/step - accuracy: 0.9098 - loss: 0.2478

Epoch 3: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **89s** 546ms/step - accuracy: 0.9098 - loss: 0.2476 - val_accuracy: 0.5000 - val_loss: 64.1739 - learning_rate: 0.0010

Epoch 4/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **91s** 558ms/step - accuracy: 0.9360 - loss: 0.1625 - val_accuracy: 0.5000 - val_loss: 44.5910 - learning_rate: 3.0000e-04

Epoch 5/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **91s** 555ms/step - accuracy: 0.9533 - loss: 0.1326 - val_accuracy: 0.6250 - val_loss: 1.0031 - learning_rate: 3.0000e-04

Epoch 6/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **91s** 556ms/step - accuracy: 0.9612 - loss: 0.1236 - val_accuracy: 0.5625 - val_loss: 1.1806 - learning_rate: 3.0000e-04

Epoch 7/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **0s** 621ms/step - accuracy: 0.9538 - loss: 0.1196

Epoch 7: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **101s** 622ms/step - accuracy: 0.9538 - loss: 0.1196 - val_accuracy: 0.5000 - val_loss: 5.7666 - learning_rate: 3.0000e-04

Epoch 8/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **98s** 602ms/step - accuracy: 0.9638 - loss: 0.1081 - val_accuracy: 0.7500 - val_loss: 0.5983 - learning_rate: 9.0000e-05

Epoch 9/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **92s** 563ms/step - accuracy: 0.9565 - loss: 0.1195 - val_accuracy: 0.6250 - val_loss: 0.6194 - learning_rate: 9.0000e-05

Epoch 10/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **0s** 559ms/step - accuracy: 0.9631 - loss: 0.1038

Epoch 11/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **94s** 579ms/step - accuracy: 0.9645 - loss: 0.1070 - val_accuracy: 0.6875 - val_loss: 1.8387 - learning_rate: 2.7000e-05

Epoch 12/12

163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **0s** 563ms/step - accuracy: 0.9686 - loss: 0.0895

Epoch 12: ReduceLROnPlateau reducing learning rate to 8.100000013655517e-06.

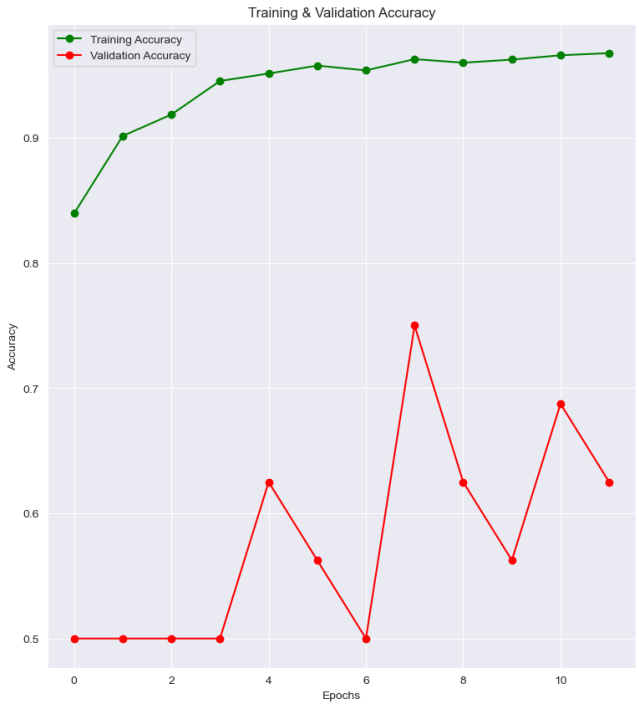
163/163 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **92s** 564ms/step - accuracy: 0.9685 - loss: 0.0895 - val_accuracy: 0.6250 - val_loss: 0.7614 - learning_rate: 2.7000e-05

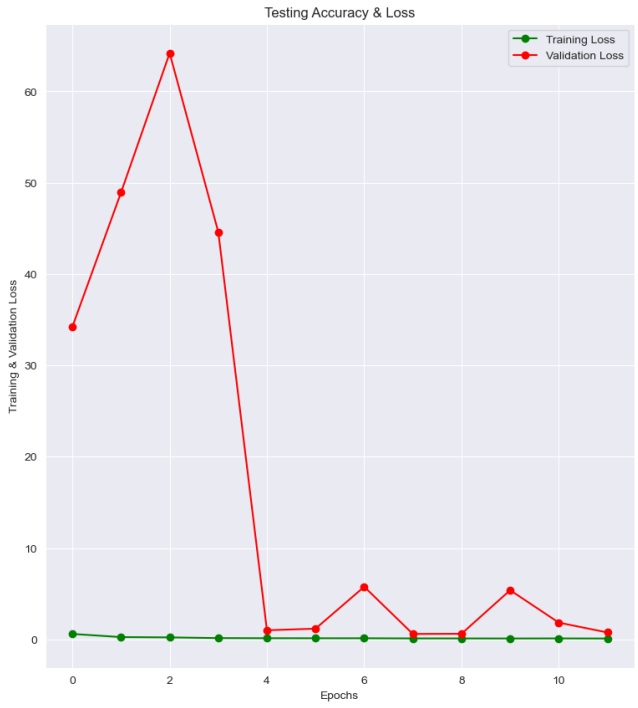
20/20 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **2s** 91ms/step - accuracy: 0.9106 - loss: 0.2547

Loss of the model is - 0.27128636837005615

20/20 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **2s** 89ms/step - accuracy: 0.9106 - loss: 0.2547

Accuracy of the model is - 89.90384340286255 %



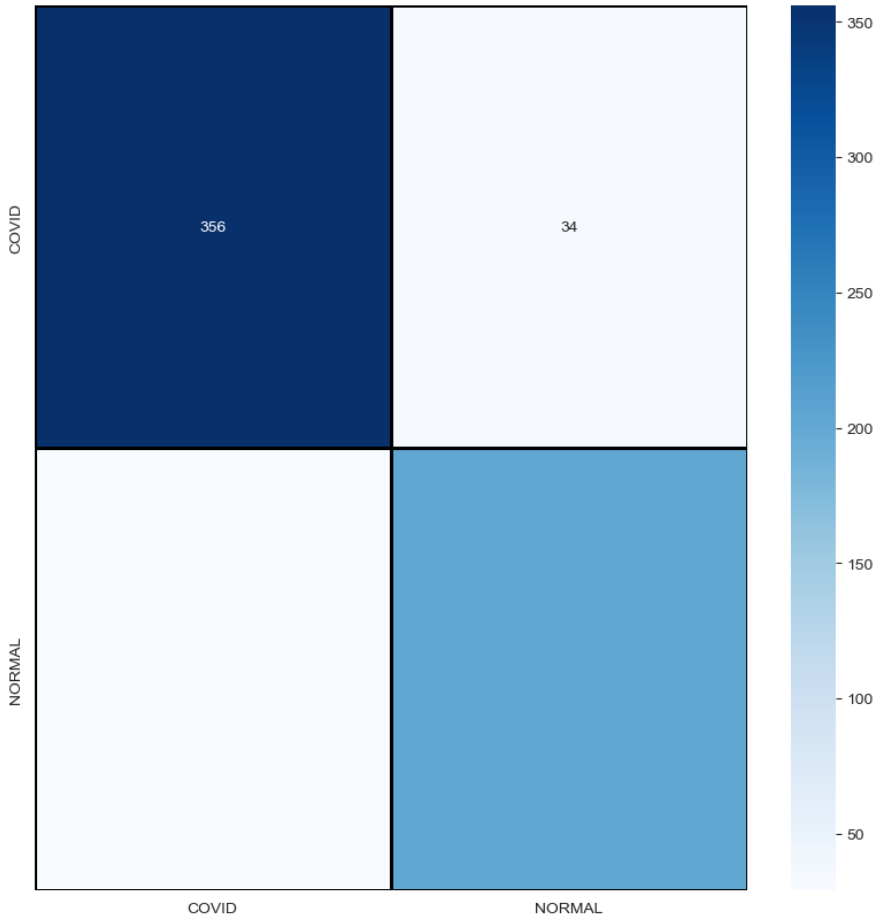


20/20 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ **2s** 103ms/step

```
[[0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [1]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]]
```

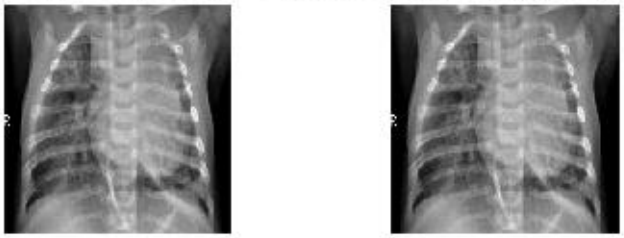
	precision	recall	f1-score	support
Covid (Class 0)	0.92	0.91	0.92	390
Normal (Class 1)	0.86	0.88	0.87	234
accuracy			0.90	624
macro avg	0.89	0.89	0.89	624
weighted avg	0.90	0.90	0.90	624

```
array([[356,  34],
       [ 29, 205]], dtype=int64)
```



Predicted Class [0],Actual Class 0 Predicted Class [0],Actual Class 0

Predicted Class [0],Actual Class 0 Predicted Class [0],Actual Class 0



Predicted Class [0],Actual Class 0 Predicted Class [0],Actual Class 0