Pneumonia Detection VGG

May 23, 2024

```
[ ]: import numpy as np
     import tensorflow as tf
     from tensorflow import keras
     from keras import layers
     import tensorflow_addons as tfa
     import pandas as pd
     import json
     import zipfile
     import os
     import seaborn as sns
     import random
     import shutil
     import time
     from PIL import Image
     from matplotlib import pyplot as plt
     from keras.models import Sequential, Model
     from keras.applications import InceptionV3, Xception, InceptionResNetV2
     from keras.applications.resnet import preprocess_input
     from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D,␣
       ↪Dropout,GlobalAveragePooling2D
     from keras.preprocessing.image import ImageDataGenerator
     from keras.callbacks import ModelCheckpoint, TensorBoard
     import wandb
     !mkdir output
     !mkdir output/tmp-augmented-images/
     random.seed(123)
```

```
/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/tensorflow_addons/utils/tfa_eol_msg.py:23: UserWarning:

TensorFlow Addons (TFA) has ended development and introduction of new features.
TFA has entered a minimal maintenance and release mode until a planned end of
life in May 2024.
Please modify downstream libraries to take dependencies from other repositories
in our TensorFlow community (e.g. Keras, Keras-CV, and Keras-NLP).

For more information see: https://github.com/tensorflow/addons/issues/2807
```

```
  warnings.warn(

mkdir: output: File exists
mkdir: output/tmp-augmented-images/: File exists
```

[ ]:

```
def resample_data(move_from, move_to, cl, images_to_move=100):
  path = "./" + 'DATASET/data/pneumonia_data'

  classes = os.listdir(path + move_from)

  cl += '/'
  curr_path = path + move_from + cl
  for _, _, files in os.walk(curr_path):
    random.shuffle(files)
    files_to_move = files[:images_to_move]
    for fn in files_to_move:
      shutil.move(curr_path + fn, path + move_to + cl + fn)
      #print('Moved ' + curr_path + fn)

  print('Resampled Images')


move_from, move_to = 'train/', 'test/'
#resample_data(move_from, move_to, 'NORMAL', 200)
# Training images
print('Number of COVID training images:')
!ls DATASET/data/pneumonia_data/train/COVID_19/ | wc -l
print('Number of NORMAL training images:')
!ls DATASET/data/pneumonia_data/train/Normal/ | wc -l
print('Number of PNEUMONIA training images:')
!ls DATASET/data/pneumonia_data/train/Pneumonia// | wc -l
print()

# Validation images
print('Number of COVID training images:')
!ls DATASET/data/pneumonia_data/val/COVID_19/ | wc -l
print('Number of NORMAL validation images:')
!ls DATASET/data/pneumonia_data/val/Normal/ | wc -l
print('Number of PNEUMONIA validation images:')
!ls DATASET/data/pneumonia_data/val/Pneumonia/ | wc -l
print()


# Test images
#resample_data('test/', 'val/', 'PNEUMONIA', 2690)
print('Number of COVID training images:')
!ls DATASET/data/pneumonia_data/test/COVID_19/ | wc -l
print('Number of NORMAL test images:')
```

```
!ls DATASET/data/pneumonia_data/test/Normal/ | wc -l
print('Number of PNEUMONIA test images:')
!ls DATASET/data/pneumonia_data/test/Pneumonia/ | wc -l
```

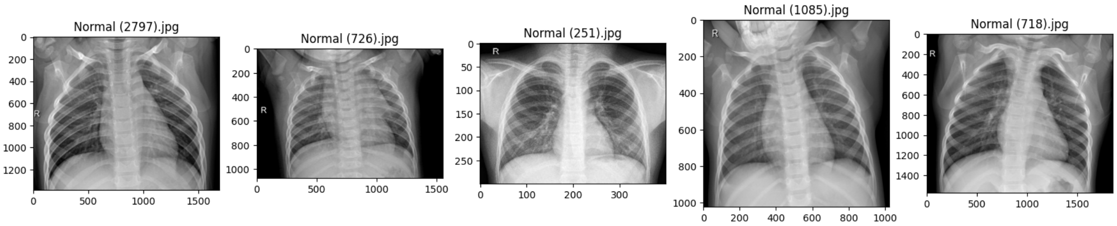
```
Number of COVID training images:
    1100
Number of NORMAL training images:
    3025
Number of PNEUMONIA training images:
    3872

Number of COVID training images:
     171
Number of NORMAL validation images:
     235
Number of PNEUMONIA validation images:
     765

Number of COVID training images:
      10
Number of NORMAL test images:
      10
Number of PNEUMONIA test images:
      20
```

[ ]:

```
def viewImagesFromDir(path, num=5):
  #Display num random images from dataset. Rerun cell for new random images. The images are only single-channel

  img_paths_visualise = sorted(
        os.path.join(path, fname)
        for fname in os.listdir(path)
        if fname.endswith(".jpg")
  )

  random.shuffle(img_paths_visualise)

  fig, ax = plt.subplots(1, num, figsize=(20, 10))
  print(num)
  for i in range(num):
    ax[i].imshow(Image.open(img_paths_visualise[i]))
    index = img_paths_visualise[i].rfind('/') + 1
    ax[i].title.set_text(img_paths_visualise[i][index:])

  fig.canvas.draw()
  time.sleep(1)
```

```python
viewImagesFromDir('DATASET/data/pneumonia_data/train/Normal/', num=5)
```

```
5
```



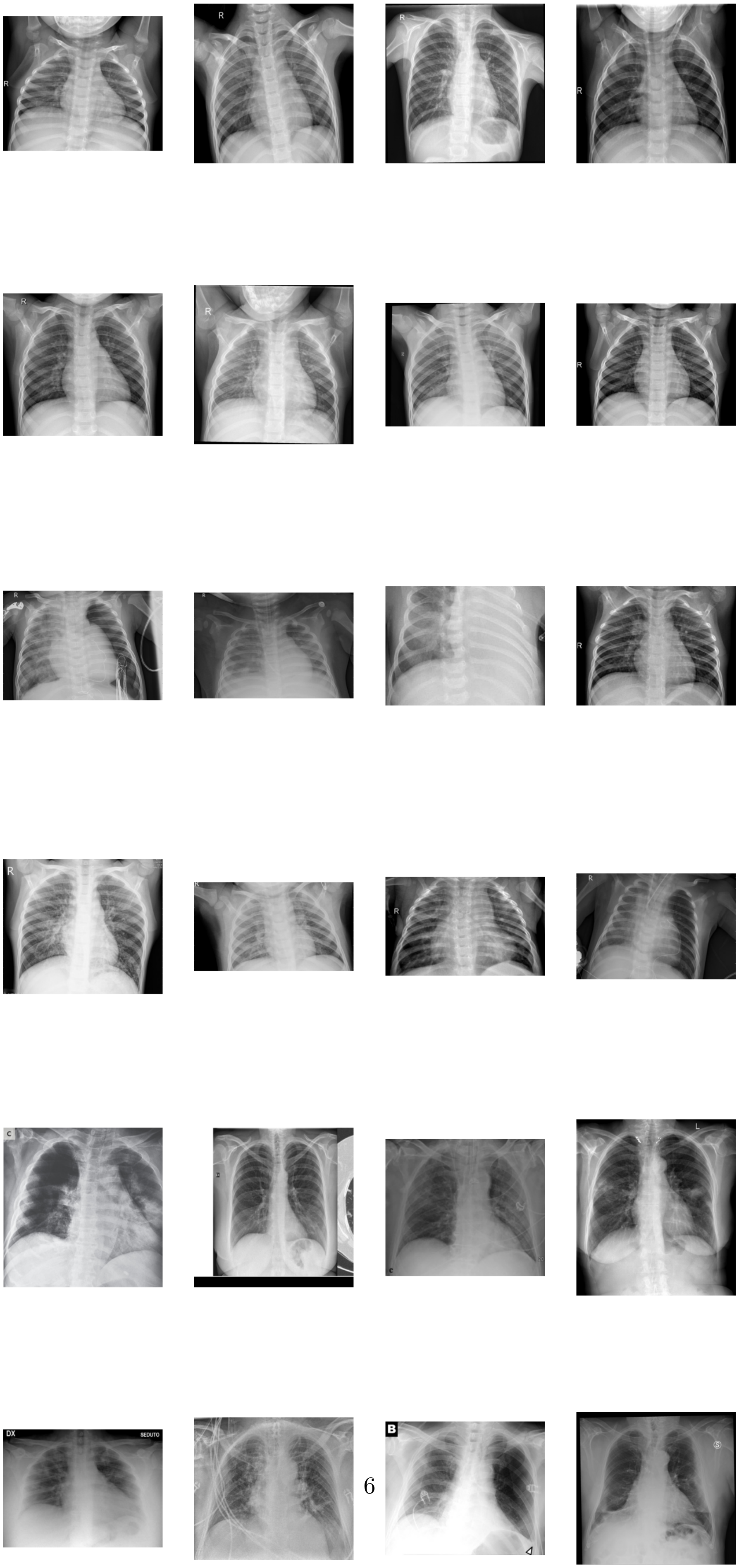
[ ]:

```python
base_dir = 'DATASET/data/pneumonia_data'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'val')

# Directory with our training covid 19 pictures
train_covid_dir = os.path.join(train_dir, 'COVID_19')

# Directory with our training normal pictures
train_normal_dir = os.path.join(train_dir, 'NORMAL')

# Directory with our training pneumonia pictures
train_pneumonia_dir = os.path.join(train_dir, 'PNEUMONIA')

# Directory with our validation covid 19 pictures
validation_covid_dir = os.path.join(validation_dir, 'COVID_19')

# Directory with our validation normal pictures
validation_normal_dir = os.path.join(validation_dir, 'NORMAL')

# Directory with our validation pneumonia pictures
validation_pneumonia_dir = os.path.join(validation_dir, 'PNEUMONIA')
```

[ ]:

```python
# Set up matplotlib fig, and size it to fit 4x4 pics
import matplotlib.image as mpimg
nrows = 6
ncols = 4

fig = plt.gcf()
fig.set_size_inches(ncols*4, nrows*6)
pic_index = 100
train_covid_fnames = os.listdir( train_covid_dir)
train_normal_fnames = os.listdir( train_normal_dir )
train_pneumonia_fnames = os.listdir( train_pneumonia_dir )
```

```python
next_covid_pix = [os.path.join(train_covid_dir, fname)
                for fname in train_covid_fnames[ pic_index-8:pic_index]
               ]
next_normal_pix = [os.path.join(train_normal_dir, fname)
                for fname in train_normal_fnames[ pic_index-8:pic_index]
               ]

next_pneumonia_pix = [os.path.join(train_pneumonia_dir, fname)
                for fname in train_pneumonia_fnames[ pic_index-8:pic_index]
               ]

for i, img_path in enumerate(next_normal_pix+next_pneumonia_pix+next_covid_pix):
  # Set up subplot; subplot indices start at 1
  sp = plt.subplot(nrows, ncols, i + 1)
  sp.axis('Off') # Don't show axes (or gridlines)

  img = mpimg.imread(img_path)
  plt.imshow(img)

plt.show()
```


R
C
L
DX
SEDUTO
B

[ ]:
```
# Add our data-augmentation parameters to ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255.,rotation_range = 40,␣
 ↪width_shift_range = 0.2, height_shift_range = 0.2, shear_range = 0.2,␣
 ↪zoom_range = 0.2, horizontal_flip = True)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator( rescale = 1.0/255. )
```

[ ]:
```
# Flow training images in batches of 20 using train_datagen generator
train_generator = train_datagen.flow_from_directory(train_dir, batch_size = 32,␣
 ↪class_mode = 'categorical', target_size = (224, 224))

# Flow validation images in batches of 20 using test_datagen generator
validation_generator = test_datagen.flow_from_directory( validation_dir, ␣
 ↪batch_size = 32, class_mode = 'categorical', target_size = (224, 224))
```

```
Found 7997 images belonging to 3 classes.
Found 1171 images belonging to 3 classes.
```

[ ]:
```
from keras.applications.vgg16 import VGG16

base_model = VGG16(input_shape = (224, 224, 3), # Shape of our images
include_top = False, # Leave out the last fully connected layer
weights = 'imagenet')
```

[ ]:
```
for layer in base_model.layers:
    layer.trainable = False
```

[ ]:
```
# Flatten the output layer to 1 dimension
x = layers.Flatten()(base_model.output)

# Add a fully connected layer with 512 hidden units and ReLU activation
x = layers.Dense(512, activation='relu')(x)

# Add a dropout rate of 0.5
x = layers.Dropout(0.5)(x)

# Add a final sigmoid layer with 1 node for classification output
x = layers.Dense(3, activation='softmax')(x)

model = tf.keras.models.Model(base_model.input, x)

# Use the legacy Keras optimizer
optimizer_legacy = tf.keras.optimizers.legacy.RMSprop(learning_rate=0.0001)
```

```
model.compile(optimizer = optimizer_legacy, loss = 'categorical_crossentropy',␣
 ↪metrics = ['acc'])
```

[ ]:

```
from keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_loss', patience=5,␣
 ↪restore_best_weights=True)
vgghist = model.fit(train_generator, validation_data=validation_generator,␣
 ↪steps_per_epoch=100, epochs=20, callbacks=[early_stopping])
```

```
Epoch 1/20
100/100 [==============================] - 225s 2s/step - loss: 0.7449 - acc:
0.7194 - val_loss: 0.3913 - val_acc: 0.8318
Epoch 2/20
100/100 [==============================] - 226s 2s/step - loss: 0.4097 - acc:
0.8394 - val_loss: 0.2760 - val_acc: 0.8813
Epoch 3/20
100/100 [==============================] - 225s 2s/step - loss: 0.3795 - acc:
0.8531 - val_loss: 0.3470 - val_acc: 0.8599
Epoch 4/20
100/100 [==============================] - 225s 2s/step - loss: 0.3463 - acc:
0.8671 - val_loss: 0.3558 - val_acc: 0.8668
Epoch 5/20
100/100 [==============================] - 226s 2s/step - loss: 0.3249 - acc:
0.8759 - val_loss: 0.3314 - val_acc: 0.8822
Epoch 6/20
100/100 [==============================] - 237s 2s/step - loss: 0.3050 - acc:
0.8856 - val_loss: 0.2279 - val_acc: 0.9095
Epoch 7/20
100/100 [==============================] - 225s 2s/step - loss: 0.2938 - acc:
0.8871 - val_loss: 0.2021 - val_acc: 0.9206
Epoch 8/20
100/100 [==============================] - 225s 2s/step - loss: 0.2759 - acc:
0.8947 - val_loss: 0.3040 - val_acc: 0.8864
Epoch 9/20
100/100 [==============================] - 225s 2s/step - loss: 0.2563 - acc:
0.9018 - val_loss: 0.2006 - val_acc: 0.9180
Epoch 10/20
100/100 [==============================] - 225s 2s/step - loss: 0.2649 - acc:
0.9059 - val_loss: 0.3110 - val_acc: 0.8822
Epoch 11/20
100/100 [==============================] - 228s 2s/step - loss: 0.2751 - acc:
0.8946 - val_loss: 0.2283 - val_acc: 0.9103
Epoch 12/20
100/100 [==============================] - 227s 2s/step - loss: 0.2564 - acc:
0.9024 - val_loss: 0.3573 - val_acc: 0.8599
Epoch 13/20
100/100 [==============================] - 227s 2s/step - loss: 0.2623 - acc:
```

```
0.8997 - val_loss: 0.1694 - val_acc: 0.9342
Epoch 14/20
100/100 [==============================] - 227s 2s/step - loss: 0.2634 - acc:
0.9056 - val_loss: 0.4541 - val_acc: 0.8113
Epoch 15/20
100/100 [==============================] - 228s 2s/step - loss: 0.2482 - acc:
0.9103 - val_loss: 0.3634 - val_acc: 0.8531
Epoch 16/20
100/100 [==============================] - 230s 2s/step - loss: 0.2498 - acc:
0.9100 - val_loss: 0.2334 - val_acc: 0.9069
Epoch 17/20
100/100 [==============================] - 228s 2s/step - loss: 0.2345 - acc:
0.9212 - val_loss: 0.2535 - val_acc: 0.9009
Epoch 18/20
100/100 [==============================] - 227s 2s/step - loss: 0.2375 - acc:
0.9087 - val_loss: 0.2143 - val_acc: 0.9146
```

[ ]:

```
history = vgghist.history
```

[ ]:

```
history.keys()
```

[ ]:

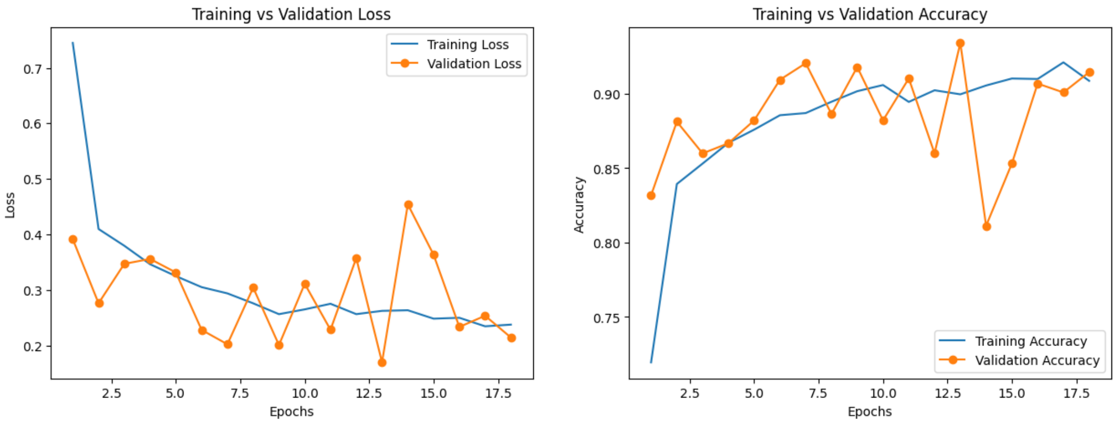
```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

[ ]:

```
train_loss, val_loss = history['loss'], history['val_loss']
train_acc, val_acc = history['acc'], history['val_acc']
```

[ ]:

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,5))
epoch_runs = [i+1 for i in range(18)]

ax1.plot(epoch_runs, train_loss, label='Training Loss')
ax1.plot(epoch_runs, val_loss, label='Validation Loss', marker='o')
ax1.set(title='Training vs Validation Loss', xlabel='Epochs',ylabel='Loss')
ax1.legend()

ax2.plot(epoch_runs, train_acc, label='Training Accuracy')
ax2.plot(epoch_runs, val_acc, label='Validation Accuracy', marker='o')
ax2.set(title='Training vs Validation Accuracy',␣
 ↪xlabel='Epochs',ylabel='Accuracy')
ax2.legend()

plt.show()
```



[ ]:

```
test_dir = 'DATASET/data/pneumonia_data/test'
# Assuming you have a separate test dataset stored in the variable test_dir
test_generator = test_datagen.flow_from_directory(test_dir, batch_size=32,␣
  ↪class_mode='categorical', target_size=(224,224))

# Evaluate the model on the test dataset
test_loss, test_acc = model.evaluate(test_generator)

print("Test Loss:", test_loss)
print("Test Accuracy:", test_acc)
```

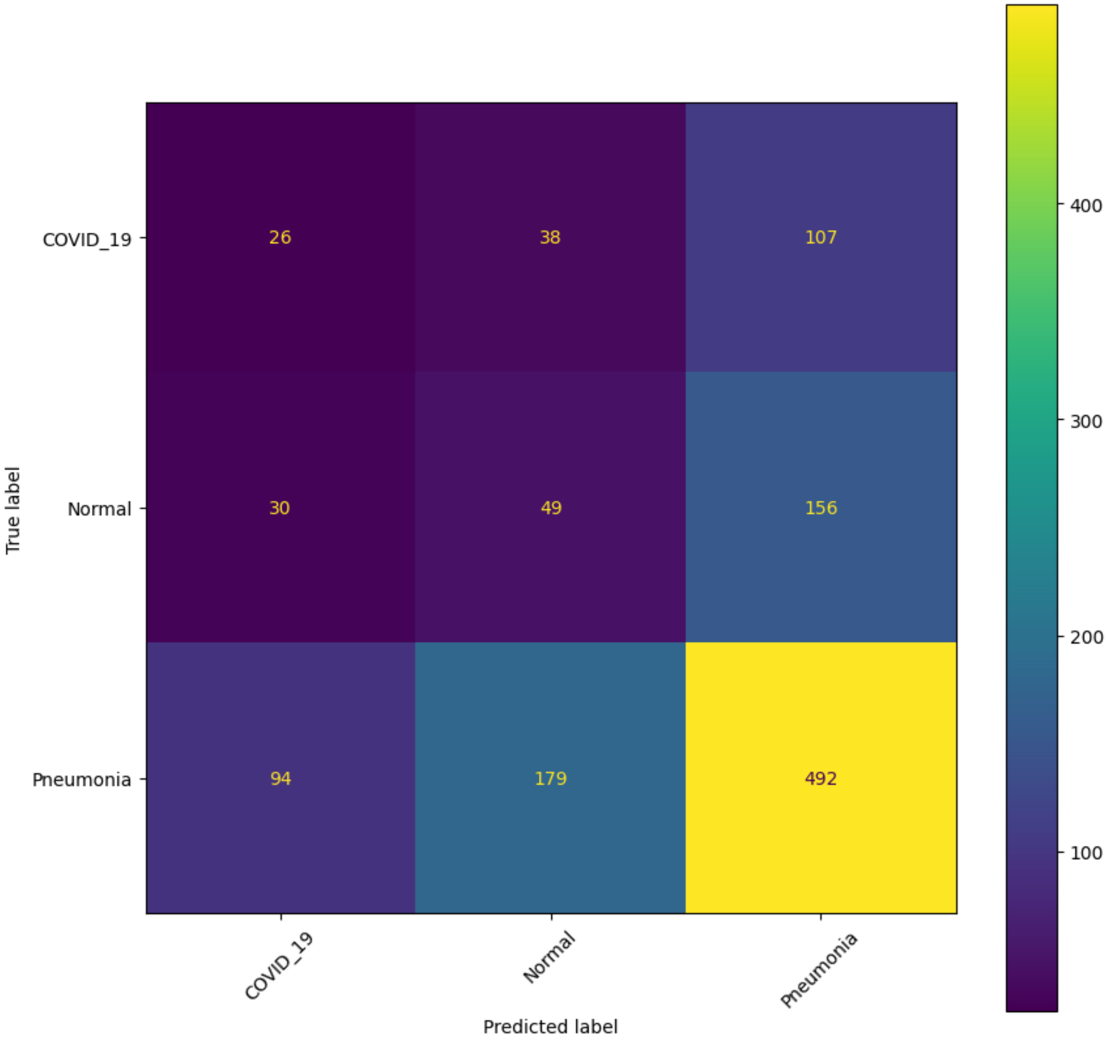
```
Found 40 images belonging to 3 classes.
2/2 [==============================] - 2s 430ms/step - loss: 0.2046 - acc:
0.9500
Test Loss: 0.20456652343273163
Test Accuracy: 0.949999988079071
```

[ ]:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
validation_steps = validation_generator.samples // validation_generator.
  ↪batch_size
predictions = model.predict(validation_generator, steps=validation_steps)
predicted_classes = np.argmax(predictions, axis=1)

# Get the true labels from the validation generator
true_classes = validation_generator.classes
class_labels = list(validation_generator.class_indices.keys())

# Predict the classes for the entire validation dataset
validation_generator.reset()  # Ensure generator starts from the beginning
predictions = model.predict(validation_generator, steps=validation_generator.
  ↪samples // validation_generator.batch_size + 1)
predicted_classes = np.argmax(predictions, axis=1)
```

```
# Ensure the number of predictions matches the number of samples
predicted_classes = predicted_classes[:len(true_classes)]

# Compute the confusion matrix
cm = confusion_matrix(true_classes, predicted_classes)
cmd = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_labels)

# Plot the confusion matrix
fig, ax = plt.subplots(figsize=(10, 10))
cmd.plot(ax=ax)
plt.xticks(rotation=45)
plt.show()
```

```
36/36 [==============================] - 60s 2s/step
37/37 [==============================] - 62s 2s/step
```


COVID_19
Normal
Pneumonia
True label
COVID_19
Normal
Pneumonia
Predicted label