covid_classifier_3

September 30, 2021

1 Covid Classifier Model

1.0.1 Goals

Classify: - Normal CXR - Viral Pneumonia CXR - COVID CXR

1.1 Create Directories for Dataset

Separate the data to use later as generators.

[ ]:
```
import os

BASE_PATH = '/home/hivini/learn/research/new-covid'
ORIGINAL_DATASET_DIR = os.path.join(BASE_PATH, 'COVID-19_Radiography_Dataset')
ORIGINAL_VIRAL_DIR = os.path.join(ORIGINAL_DATASET_DIR, 'Viral Pneumonia')
ORIGINAL_COVID_DIR = os.path.join(ORIGINAL_DATASET_DIR, 'COVID')
ORIGINAL_NORMAL_DIR = os.path.join(ORIGINAL_DATASET_DIR, 'Normal')
DATASET_DIR = os.path.join(BASE_PATH, 'small_dataset')
TRAIN_DIR = os.path.join(DATASET_DIR, 'train')
VALIDATION_DIR = os.path.join(DATASET_DIR, 'validation')
TEST_DIR = os.path.join(DATASET_DIR, 'test')
TRAIN_VIRAL_DIR = os.path.join(TRAIN_DIR, 'viral_pneumonia')
TRAIN_COVID_DIR = os.path.join(TRAIN_DIR, 'covid')
TRAIN_NORMAL_DIR = os.path.join(TRAIN_DIR, 'normal')
VALIDATION_VIRAL_DIR = os.path.join(VALIDATION_DIR, 'viral_pneumonia')
VALIDATION_COVID_DIR = os.path.join(VALIDATION_DIR, 'covid')
VALIDATION_NORMAL_DIR = os.path.join(VALIDATION_DIR, 'normal')
TEST_VIRAL_DIR = os.path.join(TEST_DIR, 'viral_pneumonia')
TEST_COVID_DIR = os.path.join(TEST_DIR, 'covid')
TEST_NORMAL_DIR = os.path.join(TEST_DIR, 'normal')


def createDir(path: str) -> None:
    if not os.path.exists(path):
        os.mkdir(path)


createDir(DATASET_DIR)
createDir(TRAIN_DIR)
```

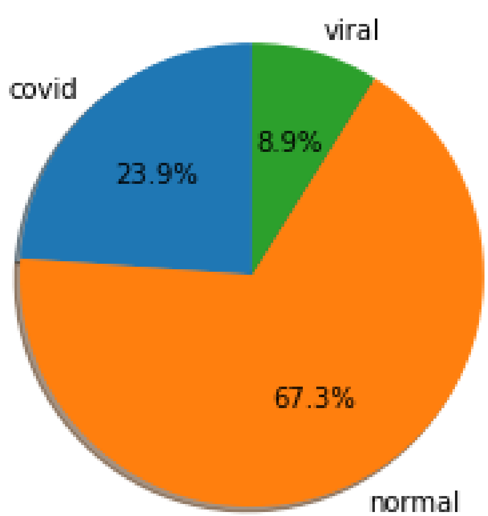
```
createDir(VALIDATION_DIR)
createDir(TEST_DIR)
createDir(TRAIN_VIRAL_DIR)
createDir(TRAIN_COVID_DIR)
createDir(TRAIN_NORMAL_DIR)
createDir(VALIDATION_VIRAL_DIR)
createDir(VALIDATION_COVID_DIR)
createDir(VALIDATION_NORMAL_DIR)
createDir(TEST_VIRAL_DIR)
createDir(TEST_COVID_DIR)
createDir(TEST_NORMAL_DIR)
```

[ ]:
```
import numpy as np
import shutil


def generate_sets(source: str):
    allFiles = os.listdir(source)
    np.random.shuffle(allFiles)
    return np.split(np.array(allFiles), [int(len(allFiles)*0.7),␣
 ↪int(len(allFiles)*0.85)])


def saveAndSeparateFiles(src_dir: str, train_dir: str, val_dir: str, test_dir):
    train_fnames, val_fnames, test_fnames = generate_sets(src_dir)
    for fname in train_fnames:
        src = os.path.join(src_dir, fname)
        dst = os.path.join(train_dir, fname)
        shutil.copyfile(src, dst)

    for fname in val_fnames:
        src = os.path.join(src_dir, fname)
        dst = os.path.join(val_dir, fname)
        shutil.copyfile(src, dst)

    for fname in test_fnames:
        src = os.path.join(src_dir, fname)
        dst = os.path.join(test_dir, fname)
        shutil.copyfile(src, dst)

create = True
if create:
    saveAndSeparateFiles(ORIGINAL_NORMAL_DIR, TRAIN_NORMAL_DIR,
                         VALIDATION_NORMAL_DIR, TEST_NORMAL_DIR)
    saveAndSeparateFiles(ORIGINAL_COVID_DIR, TRAIN_COVID_DIR,
                         VALIDATION_COVID_DIR, TEST_COVID_DIR)
    saveAndSeparateFiles(ORIGINAL_VIRAL_DIR, TRAIN_VIRAL_DIR,
```

```
                    VALIDATION_VIRAL_DIR, TEST_VIRAL_DIR)
```

1.2 Counting our images

[ ]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
normal_train = tf.io.gfile.glob(TRAIN_NORMAL_DIR + '/*')
viral_train = tf.io.gfile.glob(TRAIN_VIRAL_DIR + '/*')
covid_train = tf.io.gfile.glob(TRAIN_COVID_DIR + '/*')

# Plotting Distribution of Each Classes
image_count = {'covid': len(covid_train), 'normal': len(
    normal_train), 'viral': len(viral_train)}
fig1, ax1 = plt.subplots()
ax1.pie(image_count.values(),
        labels=image_count.keys(),
        shadow=True,
        autopct='%1.1f%%',
        startangle=90)
plt.show()
```



1.3 Create our Covnet Model

In this case we are doing a multi class classification, our total clases are 3: - Viral CXR - Covid CXR - Normal CXR

Our neural network will output neurons as 3 classes that will calculate the probability of being one using the softmax function.

[ ]:
```
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(150, 150,␣
↪1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))
model.summary()
```

```
Model: "sequential_4"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 148, 148, 64)      640
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 74, 74, 64)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 72, 72, 64)        36928
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout_4 (Dropout)          (None, 6272)              0
_________________________________________________________________
dense_10 (Dense)             (None, 512)               3211776
```

```
_________________________________________________________________
dense_11 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_12 (Dense)             (None, 64)                16448
_________________________________________________________________
dense_13 (Dense)             (None, 3)                 195
=================================================================
Total params: 3,618,755
Trainable params: 3,618,755
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
from keras import optimizers

model.compile(loss='categorical_crossentropy', optimizer=optimizers.
 ↪RMSprop(learning_rate=1e-4), metrics=['accuracy'])
```

[ ]:
```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale=1./255,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.3,
    # featurewise_center=True,
    # featurewise_std_normalization=True
)

# train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
evaluate_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    color_mode='grayscale'
)

validation_generator = test_datagen.flow_from_directory(
    VALIDATION_DIR,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    color_mode='grayscale'
)
```

```
test_generator = evaluate_datagen.flow_from_directory(
    TEST_DIR,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    color_mode='grayscale'
)
```

```
Found 10606 images belonging to 3 classes.
Found 2273 images belonging to 3 classes.
Found 2274 images belonging to 3 classes.
```

[ ]:
```
import numpy as np
from sklearn.utils import class_weight
classes = train_generator.classes
class_weights = class_weight.compute_class_weight(None,
                                                  np.unique(classes),
                                                  classes)
history = model.fit(
    train_generator,
    steps_per_epoch=100,
    epochs=100,
    validation_data=validation_generator,
    validation_steps=50,
    class_weight=dict(zip(np.unique(classes), class_weights))
)
```

```
Epoch 1/100
100/100 [==============================] - 13s 121ms/step - loss: 0.8130 -
accuracy: 0.6849 - val_loss: 0.5974 - val_accuracy: 0.7063
Epoch 2/100
100/100 [==============================] - 12s 116ms/step - loss: 0.6189 -
accuracy: 0.7138 - val_loss: 0.5399 - val_accuracy: 0.7419
Epoch 3/100
100/100 [==============================] - 11s 113ms/step - loss: 0.5885 -
accuracy: 0.7167 - val_loss: 0.5650 - val_accuracy: 0.7150
Epoch 4/100
100/100 [==============================] - 11s 113ms/step - loss: 0.5898 -
accuracy: 0.7152 - val_loss: 0.4924 - val_accuracy: 0.7769
Epoch 5/100
100/100 [==============================] - 11s 114ms/step - loss: 0.5656 -
accuracy: 0.7415 - val_loss: 0.4732 - val_accuracy: 0.7806
Epoch 6/100
100/100 [==============================] - 11s 115ms/step - loss: 0.5454 -
accuracy: 0.7481 - val_loss: 0.4923 - val_accuracy: 0.7594
Epoch 7/100
100/100 [==============================] - 11s 114ms/step - loss: 0.5470 -
```

```
accuracy: 0.7408 - val_loss: 0.4793 - val_accuracy: 0.7569
Epoch 8/100
100/100 [==============================] - 11s 114ms/step - loss: 0.5081 -
accuracy: 0.7600 - val_loss: 0.4259 - val_accuracy: 0.8044
Epoch 9/100
100/100 [==============================] - 11s 114ms/step - loss: 0.5104 -
accuracy: 0.7766 - val_loss: 0.4130 - val_accuracy: 0.8119
Epoch 10/100
100/100 [==============================] - 11s 114ms/step - loss: 0.4502 -
accuracy: 0.7976 - val_loss: 0.4419 - val_accuracy: 0.7987
Epoch 11/100
100/100 [==============================] - 11s 113ms/step - loss: 0.4590 -
accuracy: 0.8103 - val_loss: 0.4012 - val_accuracy: 0.8225
Epoch 12/100
100/100 [==============================] - 11s 113ms/step - loss: 0.4766 -
accuracy: 0.7885 - val_loss: 0.4143 - val_accuracy: 0.8169
Epoch 13/100
100/100 [==============================] - 11s 115ms/step - loss: 0.4538 -
accuracy: 0.8055 - val_loss: 0.4432 - val_accuracy: 0.7956
Epoch 14/100
100/100 [==============================] - 11s 113ms/step - loss: 0.4510 -
accuracy: 0.8093 - val_loss: 0.3827 - val_accuracy: 0.8338
Epoch 15/100
100/100 [==============================] - 11s 114ms/step - loss: 0.4259 -
accuracy: 0.8128 - val_loss: 0.3282 - val_accuracy: 0.8694
Epoch 16/100
100/100 [==============================] - 11s 113ms/step - loss: 0.4289 -
accuracy: 0.8051 - val_loss: 0.3546 - val_accuracy: 0.8550
Epoch 17/100
100/100 [==============================] - 11s 113ms/step - loss: 0.4414 -
accuracy: 0.8111 - val_loss: 0.3218 - val_accuracy: 0.8694
Epoch 18/100
100/100 [==============================] - 11s 114ms/step - loss: 0.4041 -
accuracy: 0.8271 - val_loss: 0.3071 - val_accuracy: 0.8806
Epoch 19/100
100/100 [==============================] - 12s 117ms/step - loss: 0.4433 -
accuracy: 0.8014 - val_loss: 0.3158 - val_accuracy: 0.8656
Epoch 20/100
100/100 [==============================] - 12s 117ms/step - loss: 0.4045 -
accuracy: 0.8274 - val_loss: 0.3330 - val_accuracy: 0.8669
Epoch 21/100
100/100 [==============================] - 12s 115ms/step - loss: 0.3997 -
accuracy: 0.8293 - val_loss: 0.3534 - val_accuracy: 0.8487
Epoch 22/100
100/100 [==============================] - 12s 115ms/step - loss: 0.4139 -
accuracy: 0.8151 - val_loss: 0.3023 - val_accuracy: 0.8756
Epoch 23/100
100/100 [==============================] - 12s 115ms/step - loss: 0.4071 -
```

```
accuracy: 0.8326 - val_loss: 0.3420 - val_accuracy: 0.8619
Epoch 24/100
100/100 [==============================] - 12s 115ms/step - loss: 0.3914 -
accuracy: 0.8300 - val_loss: 0.2690 - val_accuracy: 0.8950
Epoch 25/100
100/100 [==============================] - 11s 115ms/step - loss: 0.3813 -
accuracy: 0.8390 - val_loss: 0.3037 - val_accuracy: 0.8888
Epoch 26/100
100/100 [==============================] - 11s 114ms/step - loss: 0.3689 -
accuracy: 0.8448 - val_loss: 0.2795 - val_accuracy: 0.8938
Epoch 27/100
100/100 [==============================] - 11s 114ms/step - loss: 0.3647 -
accuracy: 0.8547 - val_loss: 0.2437 - val_accuracy: 0.9187
Epoch 28/100
100/100 [==============================] - 12s 115ms/step - loss: 0.3400 -
accuracy: 0.8489 - val_loss: 0.2541 - val_accuracy: 0.9075
Epoch 29/100
100/100 [==============================] - 11s 115ms/step - loss: 0.3726 -
accuracy: 0.8357 - val_loss: 0.2541 - val_accuracy: 0.9056
Epoch 30/100
100/100 [==============================] - 11s 114ms/step - loss: 0.3856 -
accuracy: 0.8304 - val_loss: 0.2521 - val_accuracy: 0.9075
Epoch 31/100
100/100 [==============================] - 11s 115ms/step - loss: 0.3652 -
accuracy: 0.8347 - val_loss: 0.2708 - val_accuracy: 0.8975
Epoch 32/100
100/100 [==============================] - 12s 115ms/step - loss: 0.3549 -
accuracy: 0.8507 - val_loss: 0.2408 - val_accuracy: 0.9131
Epoch 33/100
100/100 [==============================] - 11s 114ms/step - loss: 0.3677 -
accuracy: 0.8481 - val_loss: 0.2482 - val_accuracy: 0.9100
Epoch 34/100
100/100 [==============================] - 11s 112ms/step - loss: 0.3377 -
accuracy: 0.8553 - val_loss: 0.2251 - val_accuracy: 0.9194
Epoch 35/100
100/100 [==============================] - 11s 111ms/step - loss: 0.3647 -
accuracy: 0.8533 - val_loss: 0.2530 - val_accuracy: 0.9038
Epoch 36/100
100/100 [==============================] - 11s 113ms/step - loss: 0.3464 -
accuracy: 0.8552 - val_loss: 0.2598 - val_accuracy: 0.9038
Epoch 37/100
100/100 [==============================] - 13s 126ms/step - loss: 0.3319 -
accuracy: 0.8674 - val_loss: 0.2289 - val_accuracy: 0.9112
Epoch 38/100
100/100 [==============================] - 12s 122ms/step - loss: 0.3410 -
accuracy: 0.8579 - val_loss: 0.2326 - val_accuracy: 0.8988
Epoch 39/100
100/100 [==============================] - 12s 124ms/step - loss: 0.3319 -
```

```
accuracy: 0.8627 - val_loss: 0.2981 - val_accuracy: 0.8831
Epoch 40/100
100/100 [==============================] - 12s 119ms/step - loss: 0.3165 -
accuracy: 0.8604 - val_loss: 0.2439 - val_accuracy: 0.9025
Epoch 41/100
100/100 [==============================] - 12s 118ms/step - loss: 0.3187 -
accuracy: 0.8683 - val_loss: 0.2141 - val_accuracy: 0.9181
Epoch 42/100
100/100 [==============================] - 12s 121ms/step - loss: 0.2854 -
accuracy: 0.8959 - val_loss: 0.2215 - val_accuracy: 0.9162
Epoch 43/100
100/100 [==============================] - 13s 126ms/step - loss: 0.3037 -
accuracy: 0.8769 - val_loss: 0.2190 - val_accuracy: 0.9131
Epoch 44/100
100/100 [==============================] - 12s 121ms/step - loss: 0.2881 -
accuracy: 0.8872 - val_loss: 0.2276 - val_accuracy: 0.9144
Epoch 45/100
100/100 [==============================] - 13s 127ms/step - loss: 0.3092 -
accuracy: 0.8818 - val_loss: 0.2212 - val_accuracy: 0.9044
Epoch 46/100
100/100 [==============================] - 12s 123ms/step - loss: 0.2954 -
accuracy: 0.8760 - val_loss: 0.2167 - val_accuracy: 0.9144
Epoch 47/100
100/100 [==============================] - 14s 135ms/step - loss: 0.2905 -
accuracy: 0.8811 - val_loss: 0.2294 - val_accuracy: 0.9044
Epoch 48/100
100/100 [==============================] - 14s 136ms/step - loss: 0.2904 -
accuracy: 0.8819 - val_loss: 0.2117 - val_accuracy: 0.9106
Epoch 49/100
100/100 [==============================] - 14s 136ms/step - loss: 0.2997 -
accuracy: 0.8715 - val_loss: 0.1960 - val_accuracy: 0.9200
Epoch 50/100
100/100 [==============================] - 13s 133ms/step - loss: 0.2743 -
accuracy: 0.8952 - val_loss: 0.2117 - val_accuracy: 0.9137
Epoch 51/100
100/100 [==============================] - 12s 117ms/step - loss: 0.2819 -
accuracy: 0.8938 - val_loss: 0.2090 - val_accuracy: 0.9187
Epoch 52/100
100/100 [==============================] - 14s 139ms/step - loss: 0.2934 -
accuracy: 0.8839 - val_loss: 0.2192 - val_accuracy: 0.9087
Epoch 53/100
100/100 [==============================] - 14s 145ms/step - loss: 0.2913 -
accuracy: 0.8791 - val_loss: 0.2244 - val_accuracy: 0.9075
Epoch 54/100
100/100 [==============================] - 14s 140ms/step - loss: 0.2914 -
accuracy: 0.8822 - val_loss: 0.1670 - val_accuracy: 0.9388
Epoch 55/100
100/100 [==============================] - 14s 140ms/step - loss: 0.2682 -
```

```
accuracy: 0.8961 - val_loss: 0.1682 - val_accuracy: 0.9300
Epoch 56/100
100/100 [==============================] - 14s 135ms/step - loss: 0.2665 -
accuracy: 0.8928 - val_loss: 0.3948 - val_accuracy: 0.8181
Epoch 57/100
100/100 [==============================] - 14s 139ms/step - loss: 0.2744 -
accuracy: 0.8912 - val_loss: 0.2014 - val_accuracy: 0.9212
Epoch 58/100
100/100 [==============================] - 12s 118ms/step - loss: 0.2701 -
accuracy: 0.8951 - val_loss: 0.1754 - val_accuracy: 0.9312
Epoch 59/100
100/100 [==============================] - 12s 120ms/step - loss: 0.2642 -
accuracy: 0.8999 - val_loss: 0.1497 - val_accuracy: 0.9425
Epoch 60/100
100/100 [==============================] - 11s 113ms/step - loss: 0.2577 -
accuracy: 0.9009 - val_loss: 0.1747 - val_accuracy: 0.9294
Epoch 61/100
100/100 [==============================] - 11s 109ms/step - loss: 0.2452 -
accuracy: 0.9069 - val_loss: 0.1580 - val_accuracy: 0.9431
Epoch 62/100
100/100 [==============================] - 11s 111ms/step - loss: 0.2433 -
accuracy: 0.9078 - val_loss: 0.1676 - val_accuracy: 0.9325
Epoch 63/100
100/100 [==============================] - 11s 111ms/step - loss: 0.2741 -
accuracy: 0.8934 - val_loss: 0.1611 - val_accuracy: 0.9425
Epoch 64/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2169 -
accuracy: 0.9219 - val_loss: 0.1654 - val_accuracy: 0.9312
Epoch 65/100
100/100 [==============================] - 11s 109ms/step - loss: 0.2262 -
accuracy: 0.9121 - val_loss: 0.1585 - val_accuracy: 0.9438
Epoch 66/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2035 -
accuracy: 0.9248 - val_loss: 0.1996 - val_accuracy: 0.9144
Epoch 67/100
100/100 [==============================] - 11s 109ms/step - loss: 0.2492 -
accuracy: 0.9037 - val_loss: 0.1806 - val_accuracy: 0.9344
Epoch 68/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2425 -
accuracy: 0.9063 - val_loss: 0.1776 - val_accuracy: 0.9269
Epoch 69/100
100/100 [==============================] - 11s 109ms/step - loss: 0.2430 -
accuracy: 0.9024 - val_loss: 0.1440 - val_accuracy: 0.9481
Epoch 70/100
100/100 [==============================] - 11s 111ms/step - loss: 0.2402 -
accuracy: 0.9046 - val_loss: 0.1733 - val_accuracy: 0.9369
Epoch 71/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2517 -
```

```
accuracy: 0.9001 - val_loss: 0.1725 - val_accuracy: 0.9256
Epoch 72/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2335 -
accuracy: 0.9117 - val_loss: 0.1762 - val_accuracy: 0.9306
Epoch 73/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2250 -
accuracy: 0.9133 - val_loss: 0.2168 - val_accuracy: 0.9125
Epoch 74/100
100/100 [==============================] - 11s 110ms/step - loss: 0.1916 -
accuracy: 0.9227 - val_loss: 0.1379 - val_accuracy: 0.9488
Epoch 75/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2196 -
accuracy: 0.9155 - val_loss: 0.1598 - val_accuracy: 0.9406
Epoch 76/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2140 -
accuracy: 0.9174 - val_loss: 0.1666 - val_accuracy: 0.9287
Epoch 77/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2151 -
accuracy: 0.9234 - val_loss: 0.1949 - val_accuracy: 0.9294
Epoch 78/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2155 -
accuracy: 0.9168 - val_loss: 0.1566 - val_accuracy: 0.9344
Epoch 79/100
100/100 [==============================] - 11s 110ms/step - loss: 0.2224 -
accuracy: 0.9164 - val_loss: 0.1453 - val_accuracy: 0.9400
Epoch 80/100
100/100 [==============================] - 12s 118ms/step - loss: 0.2410 -
accuracy: 0.9095 - val_loss: 0.1690 - val_accuracy: 0.9312
Epoch 81/100
100/100 [==============================] - 12s 121ms/step - loss: 0.2179 -
accuracy: 0.9161 - val_loss: 0.1570 - val_accuracy: 0.9431
Epoch 82/100
100/100 [==============================] - 12s 116ms/step - loss: 0.1987 -
accuracy: 0.9190 - val_loss: 0.1607 - val_accuracy: 0.9388
Epoch 83/100
100/100 [==============================] - 11s 114ms/step - loss: 0.1962 -
accuracy: 0.9281 - val_loss: 0.1208 - val_accuracy: 0.9563
Epoch 84/100
100/100 [==============================] - 11s 113ms/step - loss: 0.2143 -
accuracy: 0.9086 - val_loss: 0.1680 - val_accuracy: 0.9219
Epoch 85/100
100/100 [==============================] - 11s 114ms/step - loss: 0.2012 -
accuracy: 0.9212 - val_loss: 0.1663 - val_accuracy: 0.9300
Epoch 86/100
100/100 [==============================] - 11s 114ms/step - loss: 0.2519 -
accuracy: 0.9025 - val_loss: 0.1517 - val_accuracy: 0.9375
Epoch 87/100
100/100 [==============================] - 11s 113ms/step - loss: 0.1992 -
```

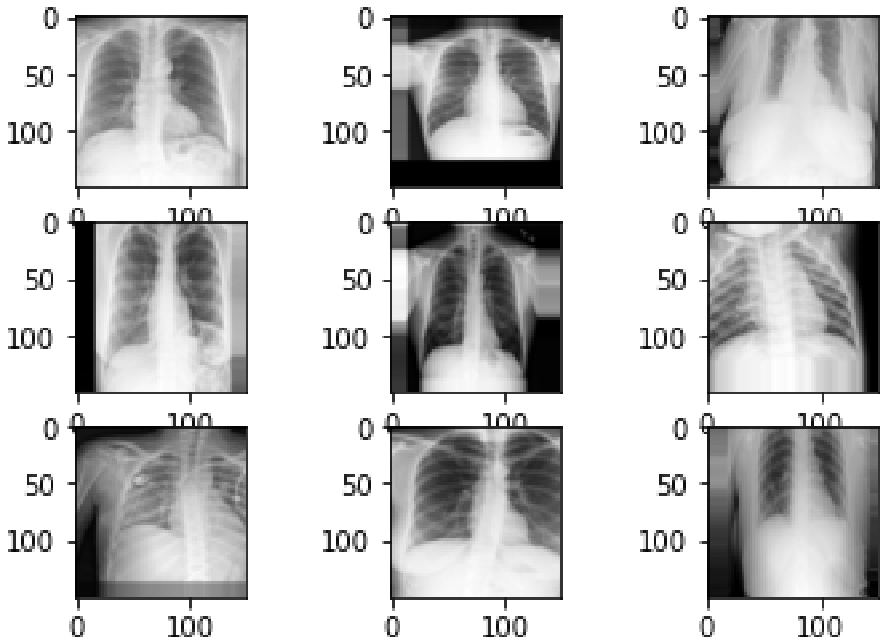
```
accuracy: 0.9240 - val_loss: 0.1565 - val_accuracy: 0.9375
Epoch 88/100
100/100 [==============================] - 11s 114ms/step - loss: 0.1983 -
accuracy: 0.9210 - val_loss: 0.1777 - val_accuracy: 0.9337
Epoch 89/100
100/100 [==============================] - 11s 114ms/step - loss: 0.2105 -
accuracy: 0.9240 - val_loss: 0.1317 - val_accuracy: 0.9494
Epoch 90/100
100/100 [==============================] - 11s 114ms/step - loss: 0.2170 -
accuracy: 0.9108 - val_loss: 0.1361 - val_accuracy: 0.9506
Epoch 91/100
100/100 [==============================] - 11s 113ms/step - loss: 0.1906 -
accuracy: 0.9278 - val_loss: 0.1412 - val_accuracy: 0.9406
Epoch 92/100
100/100 [==============================] - 11s 115ms/step - loss: 0.1864 -
accuracy: 0.9289 - val_loss: 0.1727 - val_accuracy: 0.9325
Epoch 93/100
100/100 [==============================] - 11s 115ms/step - loss: 0.2063 -
accuracy: 0.9267 - val_loss: 0.1666 - val_accuracy: 0.9300
Epoch 94/100
100/100 [==============================] - 11s 114ms/step - loss: 0.2017 -
accuracy: 0.9186 - val_loss: 0.1320 - val_accuracy: 0.9513
Epoch 95/100
100/100 [==============================] - 11s 114ms/step - loss: 0.1946 -
accuracy: 0.9246 - val_loss: 0.1445 - val_accuracy: 0.9413
Epoch 96/100
100/100 [==============================] - 11s 113ms/step - loss: 0.1888 -
accuracy: 0.9337 - val_loss: 0.1154 - val_accuracy: 0.9606
Epoch 97/100
100/100 [==============================] - 12s 120ms/step - loss: 0.1898 -
accuracy: 0.9289 - val_loss: 0.1809 - val_accuracy: 0.9287
Epoch 98/100
100/100 [==============================] - 12s 118ms/step - loss: 0.1866 -
accuracy: 0.9334 - val_loss: 0.1171 - val_accuracy: 0.9563
Epoch 99/100
100/100 [==============================] - 11s 114ms/step - loss: 0.1926 -
accuracy: 0.9279 - val_loss: 0.1420 - val_accuracy: 0.9538
Epoch 100/100
100/100 [==============================] - 11s 114ms/step - loss: 0.1886 -
accuracy: 0.9299 - val_loss: 0.1520 - val_accuracy: 0.9400
```

[ ]:
```
model.save(os.path.join(BASE_PATH, 'covid_classifier_result.h5'))
```

[ ]:
```
test_loss, test_acc = model.evaluate(test_generator)
```

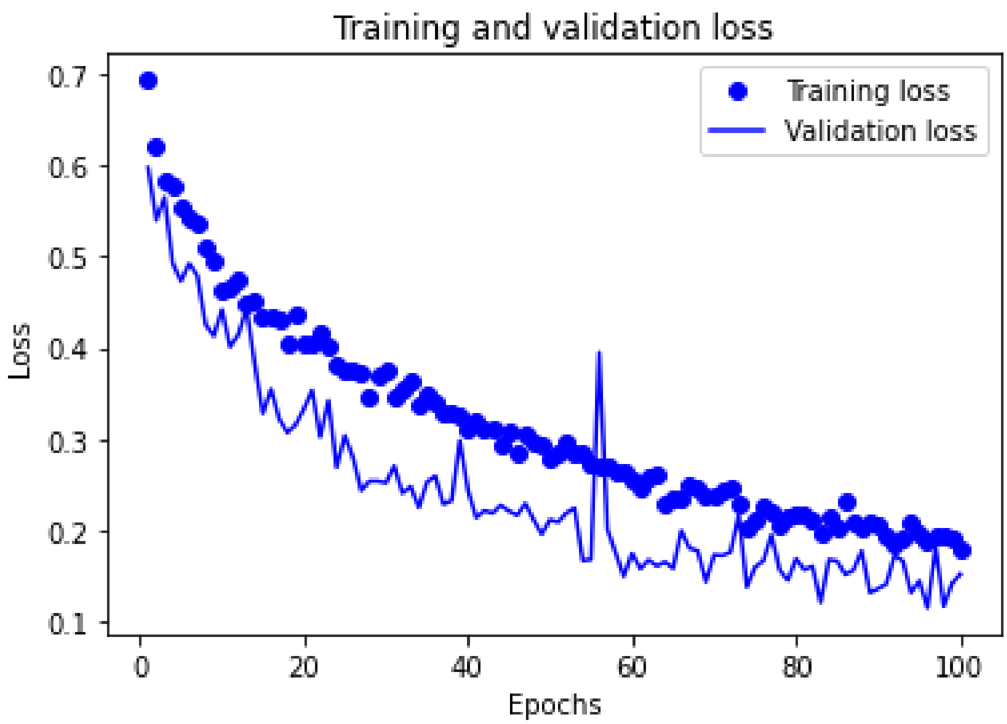
```
72/72 [==============================] - 4s 51ms/step - loss: 0.1974 - accuracy:
0.9305
```

[ ]:
```python
for X_batch, y_batch in train_generator:
        # create a grid of 3x3 images
        for i in range(0, 9):
                plt.subplot(330 + 1 + i)
                plt.imshow(X_batch[i].reshape(150, 150), cmap=plt.
 ↪get_cmap('gray'))
        # show the plot
        plt.show()
        break
```

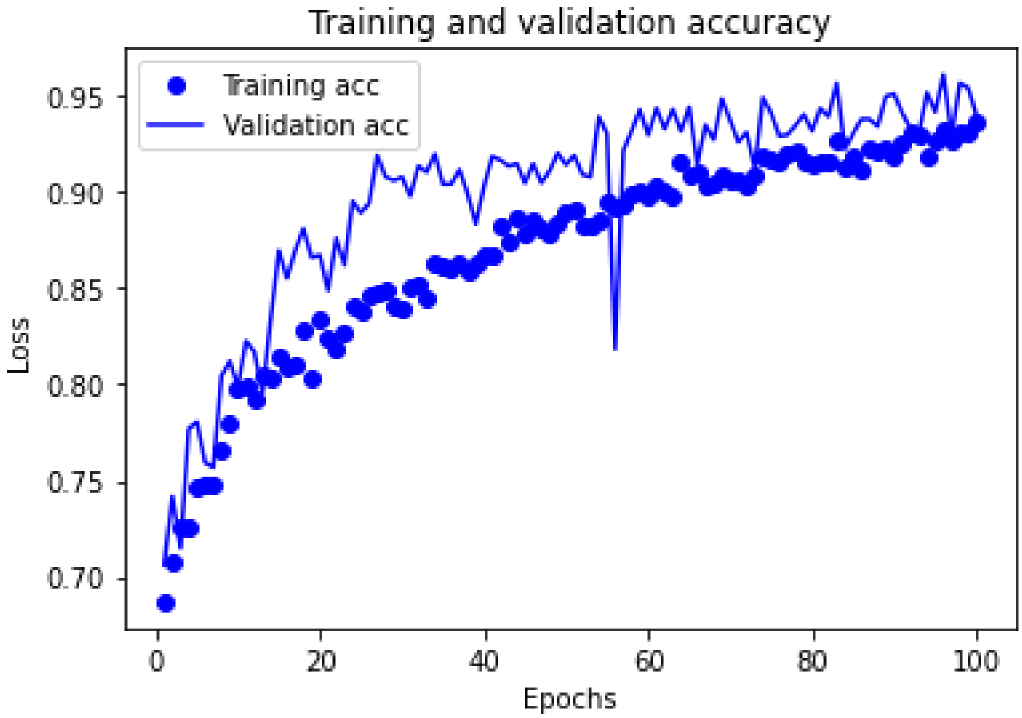


[ ]:
```python
import matplotlib.pyplot as plt

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)
# bo is for blue dot.
plt.plot(epochs, loss, 'bo', label='Training loss')
# b is for solid blue line
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
plt.legend()

plt.show()
```



[ ]:
```
plt.clf()

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



Training and validation accuracy
Training acc
Validation acc
0.95
0.90
0.85
0.80
0.75
0.70
Loss
0
20
40
60
80
100
Epochs