covid_classifier_wtf

October 7, 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.

```
[ ]: # Matriz de confusion, cambiar learnings rates (learning rates dinamicos),␣
      ↪dropouts 0.3 & 0.2, Batch Normalization
     # K-Fold o avg de modelos
     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)
```

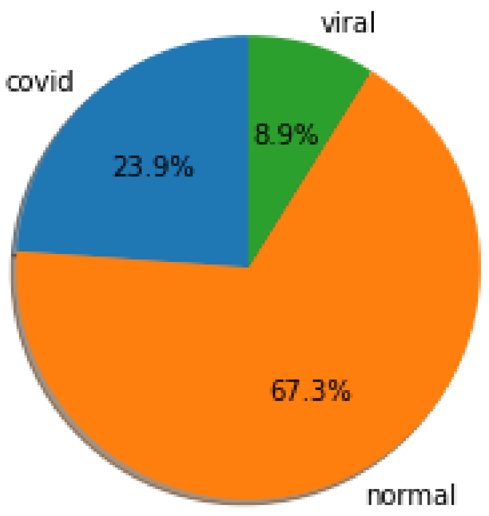
```
createDir(DATASET_DIR)
createDir(TRAIN_DIR)
createDir(VALIDATION_DIR)
createDir(TEST_DIR)
createDir(TRAIN_VIRAL_DIR)
createDir(TRAIN_COVID_DIR)
createDir(TRAIN_NORMAL_DIR)
createDir(VALIDATION_VIRAL_DIR)
createDir(VALIDATION_COVID_DIR)
createDir(VALIDATION_NORMAL_DIR)
createDir(TEST_VIRAL_DIR)
createDir(TEST_COVID_DIR)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

```
2021-10-07 13:49:55.648448: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudart.so.10.1
```



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.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.BatchNormalization())
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(64, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))
model.summary()
```

```
2021-10-07 13:50:03.288062: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not
creating XLA devices, tf_xla_enable_xla_devices not set
2021-10-07 13:50:03.298978: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcuda.so.1
2021-10-07 13:50:03.806129: E
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to
read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-10-07 13:50:03.806351: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1720] Found device 0 with
properties:
pciBusID: 0000:01:00.0 name: NVIDIA GeForce RTX 2080 with Max-Q Design
computeCapability: 7.5
coreClock: 1.215GHz coreCount: 46 deviceMemorySize: 8.00GiB
deviceMemoryBandwidth: 357.69GiB/s
2021-10-07 13:50:03.806382: I
```

```
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudart.so.10.1
2021-10-07 13:50:03.849032: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublas.so.10
2021-10-07 13:50:03.849127: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublasLt.so.10
2021-10-07 13:50:03.883778: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcufft.so.10
2021-10-07 13:50:03.890443: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcurand.so.10
2021-10-07 13:50:03.933915: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusolver.so.10
2021-10-07 13:50:03.956571: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusparse.so.10
2021-10-07 13:50:04.028411: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudnn.so.7
2021-10-07 13:50:04.029592: E
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to
read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-10-07 13:50:04.030835: E
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to
read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-10-07 13:50:04.031200: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu
devices: 0
2021-10-07 13:50:04.032485: I tensorflow/core/platform/cpu_feature_guard.cc:142]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2021-10-07 13:50:04.034695: E
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to
read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-10-07 13:50:04.035106: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1720] Found device 0 with
properties:
pciBusID: 0000:01:00.0 name: NVIDIA GeForce RTX 2080 with Max-Q Design
```

```
computeCapability: 7.5
coreClock: 1.215GHz coreCount: 46 deviceMemorySize: 8.00GiB
deviceMemoryBandwidth: 357.69GiB/s
2021-10-07 13:50:04.035145: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudart.so.10.1
2021-10-07 13:50:04.035182: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublas.so.10
2021-10-07 13:50:04.035196: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublasLt.so.10
2021-10-07 13:50:04.035207: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcufft.so.10
2021-10-07 13:50:04.035218: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcurand.so.10
2021-10-07 13:50:04.035229: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusolver.so.10
2021-10-07 13:50:04.035241: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusparse.so.10
2021-10-07 13:50:04.035252: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudnn.so.7
2021-10-07 13:50:04.036078: E
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to
read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-10-07 13:50:04.037319: E
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to
read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-10-07 13:50:04.037644: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu
devices: 0
2021-10-07 13:50:04.038126: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudart.so.10.1
2021-10-07 13:50:05.930271: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1261] Device interconnect
StreamExecutor with strength 1 edge matrix:
2021-10-07 13:50:05.930294: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1267]      0
2021-10-07 13:50:05.930300: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1280] 0:   N
```

```
2021-10-07 13:50:05.932035: E
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to
read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-10-07 13:50:05.932292: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1489] Could not identify NUMA
node of platform GPU id 0, defaulting to 0.  Your kernel may not have been built
with NUMA support.
2021-10-07 13:50:05.932953: E
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to
read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-10-07 13:50:05.933895: E
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:927] could not open file to
read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-10-07 13:50:05.934174: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1406] Created TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:0 with 6575 MB memory) -> physical
GPU (device: 0, name: NVIDIA GeForce RTX 2080 with Max-Q Design, pci bus id:
0000:01:00.0, compute capability: 7.5)
2021-10-07 13:50:05.936429: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not
creating XLA devices, tf_xla_enable_xla_devices not set

Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 64)      640

_________________________________________________________________
batch_normalization (BatchNo (None, 148, 148, 64)      256

_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 64)        0

_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        36928

_________________________________________________________________
batch_normalization_1 (Batch (None, 72, 72, 64)        256

_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0

_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856

_________________________________________________________________
batch_normalization_2 (Batch (None, 34, 34, 128)       512

_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0

_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584

_________________________________________________________________
```

```
batch_normalization_3 (Batch (None, 15, 15, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dropout (Dropout)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3211776
_________________________________________________________________
dense_1 (Dense)              (None, 64)                32832
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 195
=================================================================
Total params: 3,505,347
Trainable params: 3,504,579
Non-trainable params: 768
_________________________________________________________________
```

[ ]:
```
from keras import optimizers

lr_schedule = optimizers.schedules.ExponentialDecay(
    initial_learning_rate=1e-4,
    decay_steps=1000,
    decay_rate=0.9)
optimizer = optimizers.Adam(learning_rate=lr_schedule)

model.compile(loss='categorical_crossentropy', optimizer=optimizer,␣
 ↪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
)

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
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint

classes = train_generator.classes
class_weights = class_weight.compute_class_weight(None,
                                                  np.unique(classes),
                                                  classes)
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=30)
mc = ModelCheckpoint('best_model.h5', monitor='val_accuracy', mode='max',␣
 ↪verbose=1, save_best_only=True)
history = model.fit(
    train_generator,
    steps_per_epoch=100,
    epochs=200,
    validation_data=validation_generator,
    validation_steps=50,
    class_weight=dict(zip(np.unique(classes), class_weights)),
    callbacks=[es, mc]
)
```

```
Epoch 1/200
```

```
100/100 [==============================] - 8s 78ms/step - loss: 0.0232 -
accuracy: 0.9921 - val_loss: 0.0749 - val_accuracy: 0.9762

Epoch 00001: val_accuracy improved from -inf to 0.97625, saving model to
best_model.h5
Epoch 2/200
100/100 [==============================] - 8s 75ms/step - loss: 0.0141 -
accuracy: 0.9961 - val_loss: 0.0923 - val_accuracy: 0.9725

Epoch 00002: val_accuracy did not improve from 0.97625
Epoch 3/200
100/100 [==============================] - 7s 73ms/step - loss: 0.0059 -
accuracy: 0.9976 - val_loss: 0.0738 - val_accuracy: 0.9806

Epoch 00003: val_accuracy improved from 0.97625 to 0.98062, saving model to
best_model.h5
Epoch 4/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0042 -
accuracy: 0.9995 - val_loss: 0.0655 - val_accuracy: 0.9806

Epoch 00004: val_accuracy did not improve from 0.98062
Epoch 5/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0046 -
accuracy: 0.9987 - val_loss: 0.0784 - val_accuracy: 0.9775

Epoch 00005: val_accuracy did not improve from 0.98062
Epoch 6/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0051 -
accuracy: 0.9981 - val_loss: 0.0854 - val_accuracy: 0.9806

Epoch 00006: val_accuracy did not improve from 0.98062
Epoch 7/200
100/100 [==============================] - 7s 71ms/step - loss: 0.0025 -
accuracy: 0.9997 - val_loss: 0.0886 - val_accuracy: 0.9769

Epoch 00007: val_accuracy did not improve from 0.98062
Epoch 8/200
100/100 [==============================] - 7s 71ms/step - loss: 0.0013 -
accuracy: 1.0000 - val_loss: 0.0639 - val_accuracy: 0.9812

Epoch 00008: val_accuracy improved from 0.98062 to 0.98125, saving model to
best_model.h5
Epoch 9/200
100/100 [==============================] - 7s 71ms/step - loss: 0.0046 -
accuracy: 0.9983 - val_loss: 0.0980 - val_accuracy: 0.9769

Epoch 00009: val_accuracy did not improve from 0.98125
Epoch 10/200
```

```
100/100 [==============================] - 7s 72ms/step - loss: 0.0023 -
accuracy: 1.0000 - val_loss: 0.0875 - val_accuracy: 0.9794

Epoch 00010: val_accuracy did not improve from 0.98125
Epoch 11/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0017 -
accuracy: 1.0000 - val_loss: 0.0798 - val_accuracy: 0.9787

Epoch 00011: val_accuracy did not improve from 0.98125
Epoch 12/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0033 -
accuracy: 0.9979 - val_loss: 0.0825 - val_accuracy: 0.9794

Epoch 00012: val_accuracy did not improve from 0.98125
Epoch 13/200
100/100 [==============================] - 7s 71ms/step - loss: 0.0036 -
accuracy: 0.9996 - val_loss: 0.0865 - val_accuracy: 0.9775

Epoch 00013: val_accuracy did not improve from 0.98125
Epoch 14/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0020 -
accuracy: 0.9991 - val_loss: 0.0892 - val_accuracy: 0.9781

Epoch 00014: val_accuracy did not improve from 0.98125
Epoch 15/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0021 -
accuracy: 0.9997 - val_loss: 0.1173 - val_accuracy: 0.9737

Epoch 00015: val_accuracy did not improve from 0.98125
Epoch 16/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0090 -
accuracy: 0.9987 - val_loss: 0.0780 - val_accuracy: 0.9769

Epoch 00016: val_accuracy did not improve from 0.98125
Epoch 17/200
100/100 [==============================] - 7s 71ms/step - loss: 0.0025 -
accuracy: 0.9993 - val_loss: 0.0858 - val_accuracy: 0.9750

Epoch 00017: val_accuracy did not improve from 0.98125
Epoch 18/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0037 -
accuracy: 0.9990 - val_loss: 0.0847 - val_accuracy: 0.9775

Epoch 00018: val_accuracy did not improve from 0.98125
Epoch 19/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0020 -
accuracy: 0.9995 - val_loss: 0.1008 - val_accuracy: 0.9750
```

```
Epoch 00019: val_accuracy did not improve from 0.98125
Epoch 20/200
100/100 [==============================] - 7s 72ms/step - loss: 0.0014 -
accuracy: 0.9995 - val_loss: 0.0972 - val_accuracy: 0.9744

Epoch 00020: val_accuracy did not improve from 0.98125
Epoch 21/200
100/100 [==============================] - 7s 73ms/step - loss: 0.0024 -
accuracy: 0.9990 - val_loss: 0.0974 - val_accuracy: 0.9762

Epoch 00021: val_accuracy did not improve from 0.98125
Epoch 22/200
100/100 [==============================] - 7s 74ms/step - loss: 0.0023 -
accuracy: 0.9989 - val_loss: 0.0952 - val_accuracy: 0.9750

Epoch 00022: val_accuracy did not improve from 0.98125
Epoch 23/200
100/100 [==============================] - 8s 76ms/step - loss: 0.0016 -
accuracy: 0.9998 - val_loss: 0.0894 - val_accuracy: 0.9769

Epoch 00023: val_accuracy did not improve from 0.98125
Epoch 24/200
100/100 [==============================] - 8s 75ms/step - loss: 0.0013 -
accuracy: 1.0000 - val_loss: 0.0841 - val_accuracy: 0.9812

Epoch 00024: val_accuracy did not improve from 0.98125
Epoch 25/200
100/100 [==============================] - 7s 74ms/step - loss: 9.2295e-04 -
accuracy: 1.0000 - val_loss: 0.0828 - val_accuracy: 0.9812

Epoch 00025: val_accuracy did not improve from 0.98125
Epoch 26/200
100/100 [==============================] - 7s 73ms/step - loss: 0.0012 -
accuracy: 0.9997 - val_loss: 0.1128 - val_accuracy: 0.9794

Epoch 00026: val_accuracy did not improve from 0.98125
Epoch 27/200
100/100 [==============================] - 7s 74ms/step - loss: 0.0018 -
accuracy: 0.9992 - val_loss: 0.1101 - val_accuracy: 0.9750

Epoch 00027: val_accuracy did not improve from 0.98125
Epoch 28/200
100/100 [==============================] - 7s 73ms/step - loss: 6.4674e-04 -
accuracy: 1.0000 - val_loss: 0.1106 - val_accuracy: 0.9737

Epoch 00028: val_accuracy did not improve from 0.98125
Epoch 29/200
100/100 [==============================] - 8s 76ms/step - loss: 0.0014 -
```

```
accuracy: 0.9996 - val_loss: 0.0765 - val_accuracy: 0.9831

Epoch 00029: val_accuracy improved from 0.98125 to 0.98312, saving model to
best_model.h5
Epoch 30/200
100/100 [==============================] - 8s 84ms/step - loss: 0.0015 -
accuracy: 0.9997 - val_loss: 0.0852 - val_accuracy: 0.9794

Epoch 00030: val_accuracy did not improve from 0.98312
Epoch 31/200
100/100 [==============================] - 8s 80ms/step - loss: 0.0014 -
accuracy: 0.9996 - val_loss: 0.0893 - val_accuracy: 0.9794

Epoch 00031: val_accuracy did not improve from 0.98312
Epoch 32/200
100/100 [==============================] - 8s 82ms/step - loss: 5.5736e-04 -
accuracy: 0.9999 - val_loss: 0.0717 - val_accuracy: 0.9825

Epoch 00032: val_accuracy did not improve from 0.98312
Epoch 33/200
100/100 [==============================] - 8s 78ms/step - loss: 0.0028 -
accuracy: 0.9991 - val_loss: 0.0628 - val_accuracy: 0.9825

Epoch 00033: val_accuracy did not improve from 0.98312
Epoch 34/200
100/100 [==============================] - 8s 83ms/step - loss: 0.0034 -
accuracy: 0.9983 - val_loss: 0.1164 - val_accuracy: 0.9825

Epoch 00034: val_accuracy did not improve from 0.98312
Epoch 35/200
100/100 [==============================] - 9s 86ms/step - loss: 0.0013 -
accuracy: 0.9998 - val_loss: 0.0890 - val_accuracy: 0.9794

Epoch 00035: val_accuracy did not improve from 0.98312
Epoch 36/200
100/100 [==============================] - 8s 83ms/step - loss: 5.6289e-04 -
accuracy: 1.0000 - val_loss: 0.0951 - val_accuracy: 0.9812

Epoch 00036: val_accuracy did not improve from 0.98312
Epoch 37/200
100/100 [==============================] - 8s 77ms/step - loss: 0.0014 -
accuracy: 0.9996 - val_loss: 0.1011 - val_accuracy: 0.9794

Epoch 00037: val_accuracy did not improve from 0.98312
Epoch 38/200
100/100 [==============================] - 7s 74ms/step - loss: 3.9309e-04 -
accuracy: 1.0000 - val_loss: 0.0883 - val_accuracy: 0.9794
```

```
Epoch 00038: val_accuracy did not improve from 0.98312
Epoch 39/200
100/100 [==============================] - 8s 75ms/step - loss: 8.8637e-04 -
accuracy: 0.9998 - val_loss: 0.0796 - val_accuracy: 0.9800

Epoch 00039: val_accuracy did not improve from 0.98312
Epoch 40/200
100/100 [==============================] - 8s 76ms/step - loss: 4.3791e-04 -
accuracy: 0.9999 - val_loss: 0.0798 - val_accuracy: 0.9831

Epoch 00040: val_accuracy did not improve from 0.98312
Epoch 41/200
100/100 [==============================] - 8s 77ms/step - loss: 5.9304e-04 -
accuracy: 1.0000 - val_loss: 0.0959 - val_accuracy: 0.9794

Epoch 00041: val_accuracy did not improve from 0.98312
Epoch 42/200
100/100 [==============================] - 8s 77ms/step - loss: 3.6425e-04 -
accuracy: 1.0000 - val_loss: 0.0924 - val_accuracy: 0.9787

Epoch 00042: val_accuracy did not improve from 0.98312
Epoch 43/200
100/100 [==============================] - 8s 75ms/step - loss: 7.4958e-04 -
accuracy: 0.9996 - val_loss: 0.0998 - val_accuracy: 0.9794

Epoch 00043: val_accuracy did not improve from 0.98312
Epoch 44/200
100/100 [==============================] - 8s 76ms/step - loss: 0.0016 -
accuracy: 0.9996 - val_loss: 0.1049 - val_accuracy: 0.9762

Epoch 00044: val_accuracy did not improve from 0.98312
Epoch 45/200
100/100 [==============================] - 8s 78ms/step - loss: 0.0012 -
accuracy: 0.9997 - val_loss: 0.1189 - val_accuracy: 0.9756

Epoch 00045: val_accuracy did not improve from 0.98312
Epoch 46/200
100/100 [==============================] - 8s 76ms/step - loss: 0.0012 -
accuracy: 0.9995 - val_loss: 0.1218 - val_accuracy: 0.9725

Epoch 00046: val_accuracy did not improve from 0.98312
Epoch 47/200
100/100 [==============================] - 8s 78ms/step - loss: 7.8107e-04 -
accuracy: 0.9998 - val_loss: 0.0764 - val_accuracy: 0.9837

Epoch 00047: val_accuracy improved from 0.98312 to 0.98375, saving model to
best_model.h5
Epoch 48/200
```

```
100/100 [==============================] - 8s 77ms/step - loss: 5.1781e-04 -
accuracy: 1.0000 - val_loss: 0.0733 - val_accuracy: 0.9825

Epoch 00048: val_accuracy did not improve from 0.98375
Epoch 49/200
100/100 [==============================] - 8s 77ms/step - loss: 0.0014 -
accuracy: 0.9991 - val_loss: 0.1069 - val_accuracy: 0.9781

Epoch 00049: val_accuracy did not improve from 0.98375
Epoch 50/200
100/100 [==============================] - 8s 78ms/step - loss: 8.5566e-04 -
accuracy: 0.9998 - val_loss: 0.0906 - val_accuracy: 0.9819

Epoch 00050: val_accuracy did not improve from 0.98375
Epoch 51/200
100/100 [==============================] - 8s 78ms/step - loss: 3.6788e-04 -
accuracy: 1.0000 - val_loss: 0.0768 - val_accuracy: 0.9837

Epoch 00051: val_accuracy did not improve from 0.98375
Epoch 52/200
100/100 [==============================] - 9s 91ms/step - loss: 4.9678e-04 -
accuracy: 0.9999 - val_loss: 0.1097 - val_accuracy: 0.9769

Epoch 00052: val_accuracy did not improve from 0.98375
Epoch 53/200
100/100 [==============================] - 8s 81ms/step - loss: 5.1972e-04 -
accuracy: 0.9999 - val_loss: 0.0952 - val_accuracy: 0.9812

Epoch 00053: val_accuracy did not improve from 0.98375
Epoch 54/200
100/100 [==============================] - 8s 76ms/step - loss: 2.9776e-04 -
accuracy: 1.0000 - val_loss: 0.0991 - val_accuracy: 0.9794

Epoch 00054: val_accuracy did not improve from 0.98375
Epoch 55/200
100/100 [==============================] - 8s 79ms/step - loss: 3.7309e-04 -
accuracy: 1.0000 - val_loss: 0.0879 - val_accuracy: 0.9812

Epoch 00055: val_accuracy did not improve from 0.98375
Epoch 56/200
100/100 [==============================] - 9s 88ms/step - loss: 2.9517e-04 -
accuracy: 1.0000 - val_loss: 0.0893 - val_accuracy: 0.9825

Epoch 00056: val_accuracy did not improve from 0.98375
Epoch 57/200
100/100 [==============================] - 8s 78ms/step - loss: 1.8530e-04 -
accuracy: 1.0000 - val_loss: 0.0965 - val_accuracy: 0.9787
```

```
Epoch 00057: val_accuracy did not improve from 0.98375
Epoch 58/200
100/100 [==============================] - 8s 82ms/step - loss: 0.0038 -
accuracy: 0.9964 - val_loss: 0.1203 - val_accuracy: 0.9731

Epoch 00058: val_accuracy did not improve from 0.98375
Epoch 59/200
100/100 [==============================] - 8s 78ms/step - loss: 3.5136e-04 -
accuracy: 1.0000 - val_loss: 0.0907 - val_accuracy: 0.9800

Epoch 00059: val_accuracy did not improve from 0.98375
Epoch 60/200
100/100 [==============================] - 8s 81ms/step - loss: 8.2233e-04 -
accuracy: 0.9999 - val_loss: 0.0882 - val_accuracy: 0.9837

Epoch 00060: val_accuracy did not improve from 0.98375
Epoch 61/200
100/100 [==============================] - 8s 78ms/step - loss: 0.0012 -
accuracy: 0.9993 - val_loss: 0.0869 - val_accuracy: 0.9787

Epoch 00061: val_accuracy did not improve from 0.98375
Epoch 62/200
100/100 [==============================] - 8s 78ms/step - loss: 0.0017 -
accuracy: 0.9987 - val_loss: 0.1048 - val_accuracy: 0.9812

Epoch 00062: val_accuracy did not improve from 0.98375
Epoch 63/200
100/100 [==============================] - 8s 78ms/step - loss: 0.0020 -
accuracy: 0.9991 - val_loss: 0.0650 - val_accuracy: 0.9844

Epoch 00063: val_accuracy improved from 0.98375 to 0.98438, saving model to
best_model.h5
Epoch 00063: early stopping
```

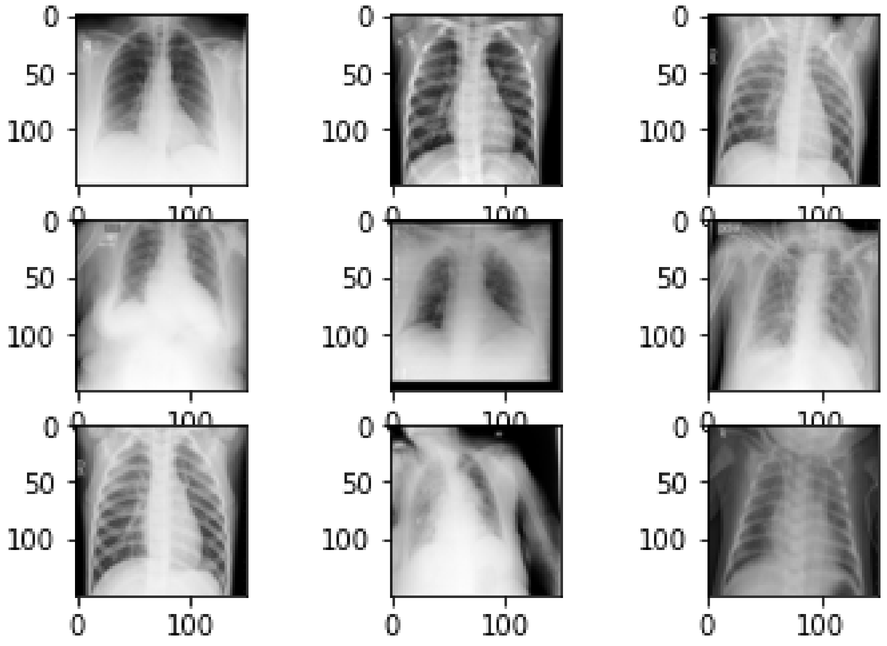
```
[ ]: model.save(os.path.join(BASE_PATH, 'covid_classifier_result.h5'))
```

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

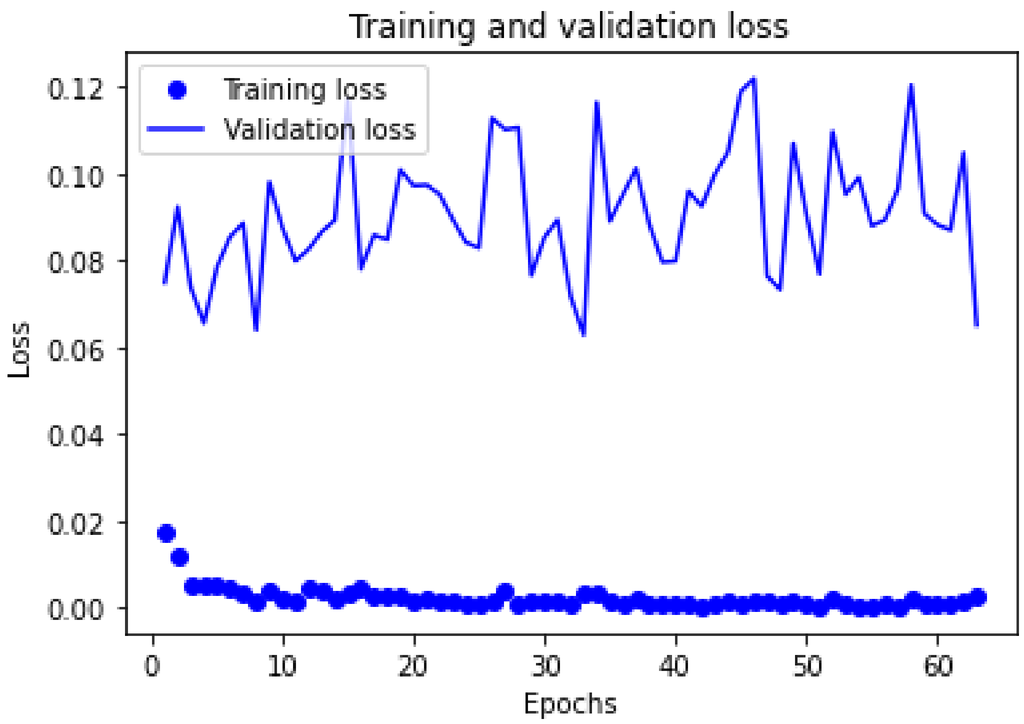
```
72/72 [==============================] - 4s 54ms/step - loss: 0.0877 - accuracy:
0.9811
```

```
[ ]: 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'))
```

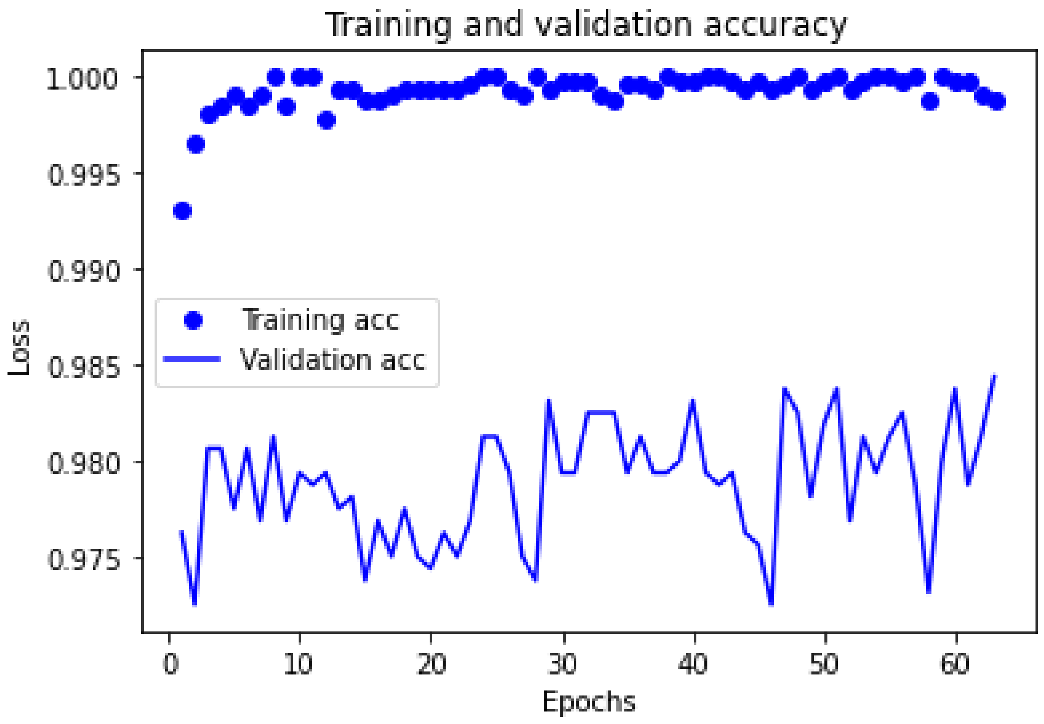
```
        # show the plot
        plt.show()
        break
```



[ ]:
```
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
Loss
Epochs
1.000
0.995
0.990
0.985
0.980
0.975
0
10
20
30
40
50
60
● Training acc
Validation acc