covid_classifier_1

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)
```

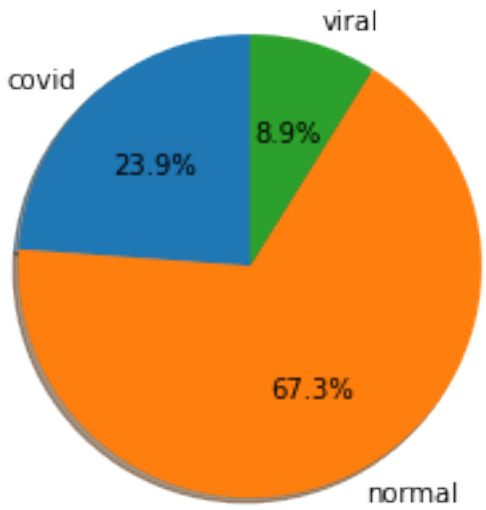
[ ]:
```
import numpy as np
import shutil


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()
```

```
2021-09-30 01:22:41.779534: 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.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(3, activation='softmax'))
model.summary()
```

```
2021-09-30 01:22:44.646687: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not
creating XLA devices, tf_xla_enable_xla_devices not set
2021-09-30 01:22:44.657558: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcuda.so.1
2021-09-30 01:22:45.038396: 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.
2021-09-30 01:22:45.038739: 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
2021-09-30 01:22:45.038807: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudart.so.10.1
2021-09-30 01:22:45.057482: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublas.so.10
```

```
2021-09-30 01:22:45.057605: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublasLt.so.10
2021-09-30 01:22:45.072147: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcufft.so.10
2021-09-30 01:22:45.072943: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcurand.so.10
2021-09-30 01:22:45.095702: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusolver.so.10
2021-09-30 01:22:45.097552: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusparse.so.10
2021-09-30 01:22:45.128164: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudnn.so.7
2021-09-30 01:22:45.129250: 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.
2021-09-30 01:22:45.130509: 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.
2021-09-30 01:22:45.130989: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu
devices: 0
2021-09-30 01:22:45.132504: 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.
2021-09-30 01:22:45.135055: 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.
2021-09-30 01:22:45.135393: 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
2021-09-30 01:22:45.135449: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
```

```
opened dynamic library libcudart.so.10.1
2021-09-30 01:22:45.135491: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublas.so.10
2021-09-30 01:22:45.135506: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcublasLt.so.10
2021-09-30 01:22:45.135518: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcufft.so.10
2021-09-30 01:22:45.135530: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcurand.so.10
2021-09-30 01:22:45.135541: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusolver.so.10
2021-09-30 01:22:45.135554: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcusparse.so.10
2021-09-30 01:22:45.135566: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudnn.so.7
2021-09-30 01:22:45.136524: 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.
2021-09-30 01:22:45.138045: 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.
2021-09-30 01:22:45.138519: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1862] Adding visible gpu
devices: 0
2021-09-30 01:22:45.138597: I
tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully
opened dynamic library libcudart.so.10.1
2021-09-30 01:22:46.591699: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1261] Device interconnect
StreamExecutor with strength 1 edge matrix:
2021-09-30 01:22:46.591724: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1267]      0
2021-09-30 01:22:46.591750: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1280] 0:   N
2021-09-30 01:22:46.593287: 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.
2021-09-30 01:22:46.593573: 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.
2021-09-30 01:22:46.594621: 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.
2021-09-30 01:22:46.596058: 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.
2021-09-30 01:22:46.596541: 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)
2021-09-30 01:22:46.597146: 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
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
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, 3)                 1539
=================================================================
Total params: 3,472,323
```

```
Trainable params: 3,472,323
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
from keras import optimizers

model.compile(loss='categorical_crossentropy', optimizer=optimizers.
 ↪RMSprop(learning_rate=1e-5), metrics=['accuracy'])
```

[ ]:
```
from keras.preprocessing.image import 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,
# )

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.
```

[ ]:
```
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,
    shuffle=True,
    validation_data=validation_generator,
    validation_steps=50,
    class_weight=dict(zip(np.unique(classes), class_weights))
)
```

```
Epoch 1/100
100/100 [==============================] - 9s 85ms/step - loss: 0.3406 -
accuracy: 0.8584 - val_loss: 0.3364 - val_accuracy: 0.8606
Epoch 2/100
100/100 [==============================] - 8s 78ms/step - loss: 0.3343 -
accuracy: 0.8603 - val_loss: 0.3347 - val_accuracy: 0.8619
Epoch 3/100
100/100 [==============================] - 8s 77ms/step - loss: 0.3300 -
accuracy: 0.8630 - val_loss: 0.3377 - val_accuracy: 0.8625
Epoch 4/100
100/100 [==============================] - 7s 74ms/step - loss: 0.3245 -
accuracy: 0.8664 - val_loss: 0.3175 - val_accuracy: 0.8712
Epoch 5/100
100/100 [==============================] - 8s 77ms/step - loss: 0.3085 -
accuracy: 0.8775 - val_loss: 0.3267 - val_accuracy: 0.8712
Epoch 6/100
100/100 [==============================] - 7s 74ms/step - loss: 0.3062 -
accuracy: 0.8756 - val_loss: 0.3107 - val_accuracy: 0.8756
Epoch 7/100
100/100 [==============================] - 7s 73ms/step - loss: 0.3105 -
accuracy: 0.8737 - val_loss: 0.3179 - val_accuracy: 0.8712
Epoch 8/100
100/100 [==============================] - 8s 77ms/step - loss: 0.3180 -
accuracy: 0.8753 - val_loss: 0.3281 - val_accuracy: 0.8581
Epoch 9/100
100/100 [==============================] - 8s 77ms/step - loss: 0.3087 -
accuracy: 0.8812 - val_loss: 0.3090 - val_accuracy: 0.8706
```

```
Epoch 10/100
100/100 [==============================] - 8s 77ms/step - loss: 0.3178 -
accuracy: 0.8731 - val_loss: 0.2996 - val_accuracy: 0.8856
Epoch 11/100
100/100 [==============================] - 8s 76ms/step - loss: 0.3025 -
accuracy: 0.8813 - val_loss: 0.3088 - val_accuracy: 0.8687
Epoch 12/100
100/100 [==============================] - 8s 76ms/step - loss: 0.3150 -
accuracy: 0.8706 - val_loss: 0.3030 - val_accuracy: 0.8794
Epoch 13/100
100/100 [==============================] - 8s 77ms/step - loss: 0.3057 -
accuracy: 0.8778 - val_loss: 0.3149 - val_accuracy: 0.8788
Epoch 14/100
100/100 [==============================] - 7s 74ms/step - loss: 0.3047 -
accuracy: 0.8816 - val_loss: 0.2868 - val_accuracy: 0.8975
Epoch 15/100
100/100 [==============================] - 7s 74ms/step - loss: 0.3012 -
accuracy: 0.8777 - val_loss: 0.2959 - val_accuracy: 0.8831
Epoch 16/100
100/100 [==============================] - 7s 74ms/step - loss: 0.3001 -
accuracy: 0.8769 - val_loss: 0.2755 - val_accuracy: 0.8869
Epoch 17/100
100/100 [==============================] - 7s 74ms/step - loss: 0.3012 -
accuracy: 0.8800 - val_loss: 0.2990 - val_accuracy: 0.8831
Epoch 18/100
100/100 [==============================] - 7s 73ms/step - loss: 0.3009 -
accuracy: 0.8819 - val_loss: 0.2775 - val_accuracy: 0.8956
Epoch 19/100
100/100 [==============================] - 7s 74ms/step - loss: 0.2848 -
accuracy: 0.8866 - val_loss: 0.2945 - val_accuracy: 0.8925
Epoch 20/100
100/100 [==============================] - 7s 75ms/step - loss: 0.3050 -
accuracy: 0.8747 - val_loss: 0.2808 - val_accuracy: 0.8963
Epoch 21/100
100/100 [==============================] - 7s 73ms/step - loss: 0.3036 -
accuracy: 0.8866 - val_loss: 0.3028 - val_accuracy: 0.8869
Epoch 22/100
100/100 [==============================] - 8s 75ms/step - loss: 0.3020 -
accuracy: 0.8781 - val_loss: 0.2811 - val_accuracy: 0.8988
Epoch 23/100
100/100 [==============================] - 7s 74ms/step - loss: 0.2819 -
accuracy: 0.8884 - val_loss: 0.2797 - val_accuracy: 0.8788
Epoch 24/100
100/100 [==============================] - 7s 74ms/step - loss: 0.3051 -
accuracy: 0.8759 - val_loss: 0.2993 - val_accuracy: 0.8850
Epoch 25/100
100/100 [==============================] - 8s 75ms/step - loss: 0.3113 -
accuracy: 0.8747 - val_loss: 0.2880 - val_accuracy: 0.8963
```

```
Epoch 26/100
100/100 [==============================] - 8s 75ms/step - loss: 0.3096 -
accuracy: 0.8828 - val_loss: 0.2654 - val_accuracy: 0.8988
Epoch 27/100
100/100 [==============================] - 7s 75ms/step - loss: 0.2781 -
accuracy: 0.8863 - val_loss: 0.2794 - val_accuracy: 0.8894
Epoch 28/100
100/100 [==============================] - 7s 74ms/step - loss: 0.2738 -
accuracy: 0.8909 - val_loss: 0.2869 - val_accuracy: 0.8913
Epoch 29/100
100/100 [==============================] - 8s 76ms/step - loss: 0.2783 -
accuracy: 0.8928 - val_loss: 0.2801 - val_accuracy: 0.8969
Epoch 30/100
100/100 [==============================] - 7s 74ms/step - loss: 0.2815 -
accuracy: 0.8928 - val_loss: 0.2817 - val_accuracy: 0.8981
Epoch 31/100
100/100 [==============================] - 8s 76ms/step - loss: 0.2790 -
accuracy: 0.8850 - val_loss: 0.2765 - val_accuracy: 0.8931
Epoch 32/100
100/100 [==============================] - 7s 74ms/step - loss: 0.2709 -
accuracy: 0.8928 - val_loss: 0.2681 - val_accuracy: 0.9031
Epoch 33/100
100/100 [==============================] - 8s 75ms/step - loss: 0.2776 -
accuracy: 0.8969 - val_loss: 0.2824 - val_accuracy: 0.8881
Epoch 34/100
100/100 [==============================] - 7s 75ms/step - loss: 0.2594 -
accuracy: 0.8991 - val_loss: 0.2544 - val_accuracy: 0.9062
Epoch 35/100
100/100 [==============================] - 8s 75ms/step - loss: 0.2657 -
accuracy: 0.8994 - val_loss: 0.2598 - val_accuracy: 0.9075
Epoch 36/100
100/100 [==============================] - 8s 75ms/step - loss: 0.2638 -
accuracy: 0.8988 - val_loss: 0.2842 - val_accuracy: 0.8963
Epoch 37/100
100/100 [==============================] - 7s 74ms/step - loss: 0.2845 -
accuracy: 0.8938 - val_loss: 0.2679 - val_accuracy: 0.9044
Epoch 38/100
100/100 [==============================] - 8s 76ms/step - loss: 0.2739 -
accuracy: 0.8981 - val_loss: 0.2654 - val_accuracy: 0.9006
Epoch 39/100
100/100 [==============================] - 8s 75ms/step - loss: 0.2620 -
accuracy: 0.8944 - val_loss: 0.2652 - val_accuracy: 0.8988
Epoch 40/100
100/100 [==============================] - 8s 76ms/step - loss: 0.2700 -
accuracy: 0.8978 - val_loss: 0.2706 - val_accuracy: 0.8994
Epoch 41/100
100/100 [==============================] - 8s 75ms/step - loss: 0.2593 -
accuracy: 0.8925 - val_loss: 0.2694 - val_accuracy: 0.8963
```

```
Epoch 42/100
100/100 [==============================] - 7s 74ms/step - loss: 0.2681 -
accuracy: 0.8972 - val_loss: 0.2588 - val_accuracy: 0.9087
Epoch 43/100
100/100 [==============================] - 8s 75ms/step - loss: 0.2666 -
accuracy: 0.8953 - val_loss: 0.2682 - val_accuracy: 0.8994
Epoch 44/100
100/100 [==============================] - 8s 75ms/step - loss: 0.2848 -
accuracy: 0.8875 - val_loss: 0.2542 - val_accuracy: 0.9125
Epoch 45/100
100/100 [==============================] - 8s 77ms/step - loss: 0.2702 -
accuracy: 0.8941 - val_loss: 0.2463 - val_accuracy: 0.9069
Epoch 46/100
100/100 [==============================] - 9s 91ms/step - loss: 0.2605 -
accuracy: 0.8969 - val_loss: 0.2559 - val_accuracy: 0.9025
Epoch 47/100
100/100 [==============================] - 9s 86ms/step - loss: 0.2613 -
accuracy: 0.9045 - val_loss: 0.2513 - val_accuracy: 0.9075
Epoch 48/100
100/100 [==============================] - 9s 85ms/step - loss: 0.2678 -
accuracy: 0.8913 - val_loss: 0.2461 - val_accuracy: 0.9150
Epoch 49/100
100/100 [==============================] - 8s 84ms/step - loss: 0.2698 -
accuracy: 0.9022 - val_loss: 0.2519 - val_accuracy: 0.9081
Epoch 50/100
100/100 [==============================] - 9s 85ms/step - loss: 0.2572 -
accuracy: 0.9075 - val_loss: 0.2642 - val_accuracy: 0.9000
Epoch 51/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2669 -
accuracy: 0.8941 - val_loss: 0.2456 - val_accuracy: 0.9156
Epoch 52/100
100/100 [==============================] - 8s 84ms/step - loss: 0.2622 -
accuracy: 0.8994 - val_loss: 0.2599 - val_accuracy: 0.9050
Epoch 53/100
100/100 [==============================] - 8s 85ms/step - loss: 0.2511 -
accuracy: 0.9025 - val_loss: 0.2506 - val_accuracy: 0.9087
Epoch 54/100
100/100 [==============================] - 9s 86ms/step - loss: 0.2451 -
accuracy: 0.9067 - val_loss: 0.2602 - val_accuracy: 0.9038
Epoch 55/100
100/100 [==============================] - 9s 85ms/step - loss: 0.2435 -
accuracy: 0.9054 - val_loss: 0.2525 - val_accuracy: 0.9112
Epoch 56/100
100/100 [==============================] - 8s 85ms/step - loss: 0.2510 -
accuracy: 0.9078 - val_loss: 0.2397 - val_accuracy: 0.9150
Epoch 57/100
100/100 [==============================] - 8s 85ms/step - loss: 0.2544 -
accuracy: 0.9009 - val_loss: 0.2403 - val_accuracy: 0.9150
```

```
Epoch 58/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2582 -
accuracy: 0.9038 - val_loss: 0.2531 - val_accuracy: 0.9100
Epoch 59/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2464 -
accuracy: 0.9041 - val_loss: 0.2515 - val_accuracy: 0.9069
Epoch 60/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2473 -
accuracy: 0.9050 - val_loss: 0.2273 - val_accuracy: 0.9231
Epoch 61/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2566 -
accuracy: 0.9092 - val_loss: 0.2432 - val_accuracy: 0.9112
Epoch 62/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2431 -
accuracy: 0.9104 - val_loss: 0.2207 - val_accuracy: 0.9194
Epoch 63/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2426 -
accuracy: 0.9069 - val_loss: 0.2353 - val_accuracy: 0.9137
Epoch 64/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2398 -
accuracy: 0.9082 - val_loss: 0.2503 - val_accuracy: 0.9069
Epoch 65/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2378 -
accuracy: 0.9079 - val_loss: 0.2447 - val_accuracy: 0.9050
Epoch 66/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2435 -
accuracy: 0.9085 - val_loss: 0.2348 - val_accuracy: 0.9206
Epoch 67/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2465 -
accuracy: 0.9019 - val_loss: 0.2488 - val_accuracy: 0.9038
Epoch 68/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2572 -
accuracy: 0.9041 - val_loss: 0.2244 - val_accuracy: 0.9175
Epoch 69/100
100/100 [==============================] - 8s 84ms/step - loss: 0.2285 -
accuracy: 0.9144 - val_loss: 0.2295 - val_accuracy: 0.9206
Epoch 70/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2354 -
accuracy: 0.9059 - val_loss: 0.2318 - val_accuracy: 0.9212
Epoch 71/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2325 -
accuracy: 0.9119 - val_loss: 0.2462 - val_accuracy: 0.9131
Epoch 72/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2243 -
accuracy: 0.9172 - val_loss: 0.2499 - val_accuracy: 0.9025
Epoch 73/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2331 -
accuracy: 0.9076 - val_loss: 0.2234 - val_accuracy: 0.9219
```

```
Epoch 74/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2450 -
accuracy: 0.9116 - val_loss: 0.2207 - val_accuracy: 0.9137
Epoch 75/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2347 -
accuracy: 0.9103 - val_loss: 0.2278 - val_accuracy: 0.9219
Epoch 76/100
100/100 [==============================] - 8s 84ms/step - loss: 0.2307 -
accuracy: 0.9147 - val_loss: 0.2229 - val_accuracy: 0.9244
Epoch 77/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2284 -
accuracy: 0.9084 - val_loss: 0.2179 - val_accuracy: 0.9219
Epoch 78/100
100/100 [==============================] - 8s 84ms/step - loss: 0.2140 -
accuracy: 0.9166 - val_loss: 0.2363 - val_accuracy: 0.9087
Epoch 79/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2259 -
accuracy: 0.9200 - val_loss: 0.2265 - val_accuracy: 0.9119
Epoch 80/100
100/100 [==============================] - 8s 84ms/step - loss: 0.2414 -
accuracy: 0.9066 - val_loss: 0.2295 - val_accuracy: 0.9212
Epoch 81/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2447 -
accuracy: 0.9100 - val_loss: 0.2356 - val_accuracy: 0.9150
Epoch 82/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2150 -
accuracy: 0.9259 - val_loss: 0.2348 - val_accuracy: 0.9131
Epoch 83/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2227 -
accuracy: 0.9137 - val_loss: 0.2274 - val_accuracy: 0.9156
Epoch 84/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2290 -
accuracy: 0.9126 - val_loss: 0.2289 - val_accuracy: 0.9187
Epoch 85/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2287 -
accuracy: 0.9129 - val_loss: 0.2184 - val_accuracy: 0.9225
Epoch 86/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2264 -
accuracy: 0.9129 - val_loss: 0.2289 - val_accuracy: 0.9106
Epoch 87/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2183 -
accuracy: 0.9206 - val_loss: 0.2234 - val_accuracy: 0.9175
Epoch 88/100
100/100 [==============================] - 8s 84ms/step - loss: 0.2321 -
accuracy: 0.9144 - val_loss: 0.2208 - val_accuracy: 0.9200
Epoch 89/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2277 -
accuracy: 0.9150 - val_loss: 0.2127 - val_accuracy: 0.9256
```

```
Epoch 90/100
100/100 [==============================] - 8s 85ms/step - loss: 0.2076 -
accuracy: 0.9225 - val_loss: 0.2174 - val_accuracy: 0.9137
Epoch 91/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2257 -
accuracy: 0.9177 - val_loss: 0.2092 - val_accuracy: 0.9194
Epoch 92/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2267 -
accuracy: 0.9131 - val_loss: 0.2184 - val_accuracy: 0.9175
Epoch 93/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2059 -
accuracy: 0.9239 - val_loss: 0.2041 - val_accuracy: 0.9275
Epoch 94/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2183 -
accuracy: 0.9162 - val_loss: 0.2218 - val_accuracy: 0.9181
Epoch 95/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2177 -
accuracy: 0.9241 - val_loss: 0.2143 - val_accuracy: 0.9194
Epoch 96/100
100/100 [==============================] - 8s 83ms/step - loss: 0.2201 -
accuracy: 0.9150 - val_loss: 0.2184 - val_accuracy: 0.9256
Epoch 97/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2119 -
accuracy: 0.9181 - val_loss: 0.2095 - val_accuracy: 0.9281
Epoch 98/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2173 -
accuracy: 0.9145 - val_loss: 0.2106 - val_accuracy: 0.9231
Epoch 99/100
100/100 [==============================] - 8s 82ms/step - loss: 0.2004 -
accuracy: 0.9283 - val_loss: 0.2095 - val_accuracy: 0.9294
Epoch 100/100
100/100 [==============================] - 9s 88ms/step - loss: 0.2190 -
accuracy: 0.9172 - val_loss: 0.2198 - val_accuracy: 0.9225
```

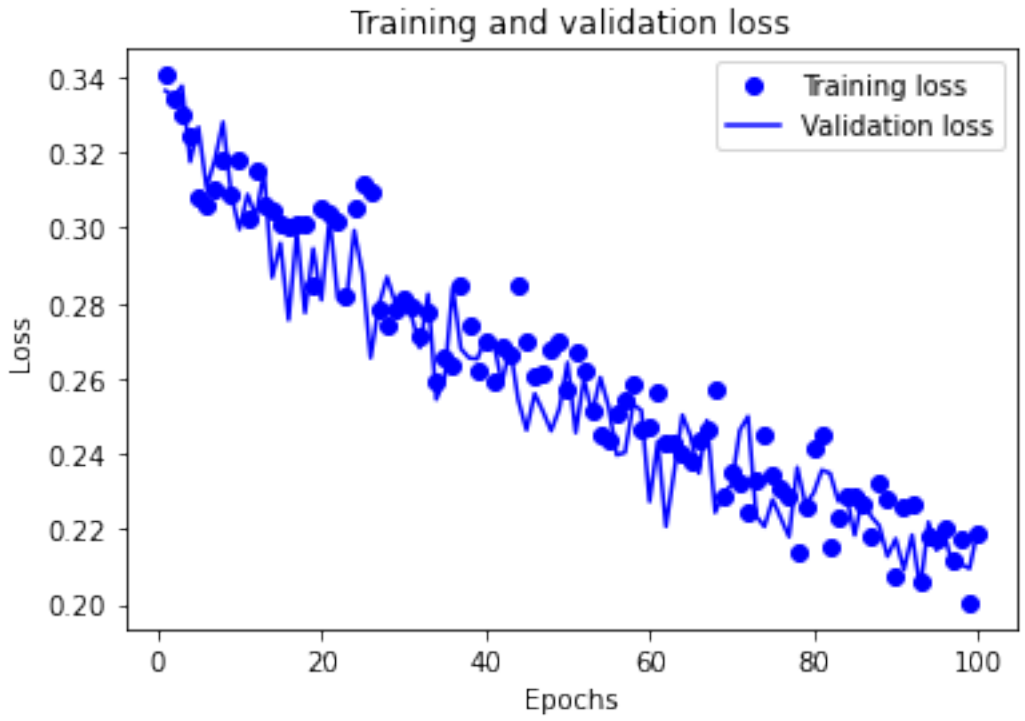
[ ]:
```
model.save(os.path.join(BASE_PATH, 'covid_classifier_result.h5'))
```

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

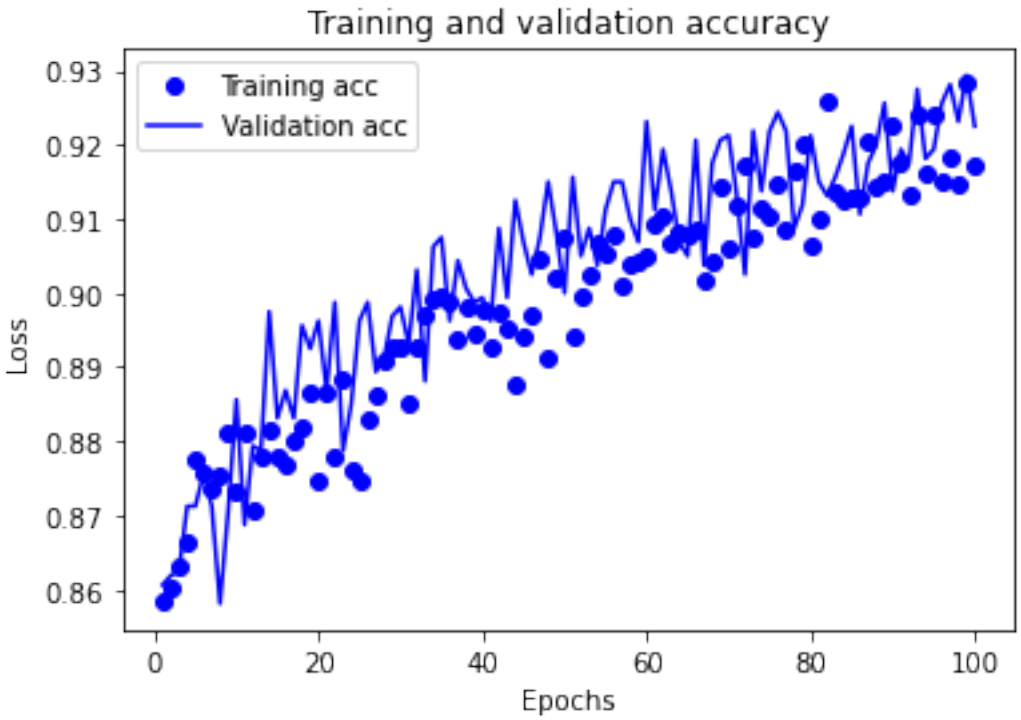
```
72/72 [==============================] - 4s 52ms/step - loss: 0.2366 - accuracy:
0.9156
```

[ ]:
```
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
Loss
Epochs
0.93
0.92
0.91
0.90
0.89
0.88
0.87
0.86
0
20
40
60
80
100