covid_classifier_2

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.

[ ]:
```python
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)
```

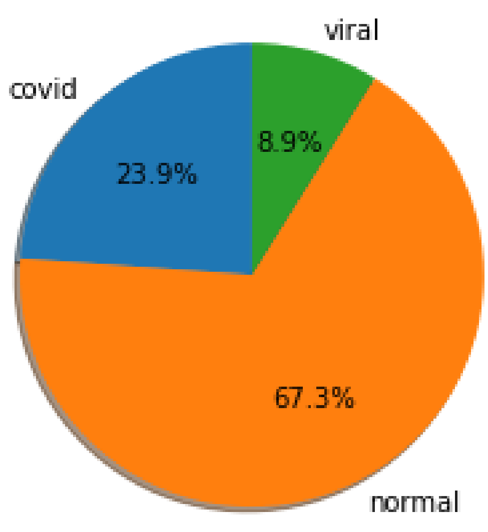
```
createDir(VALIDATION_DIR)
createDir(TEST_DIR)
createDir(TRAIN_VIRAL_DIR)
createDir(TRAIN_COVID_DIR)
createDir(TRAIN_NORMAL_DIR)
createDir(VALIDATION_VIRAL_DIR)
createDir(VALIDATION_COVID_DIR)
createDir(VALIDATION_NORMAL_DIR)
createDir(TEST_VIRAL_DIR)
createDir(TEST_COVID_DIR)
createDir(TEST_NORMAL_DIR)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

[ ]:
```
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(3, activation='softmax'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 148, 148, 64)      640
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 74, 74, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 72, 72, 64)        36928
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6272)              0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_5 (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=10,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2
)

# 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 119ms/step - loss: 0.9537 -
accuracy: 0.6400 - val_loss: 0.8224 - val_accuracy: 0.6775
Epoch 2/100
100/100 [==============================] - 12s 119ms/step - loss: 0.8262 -
accuracy: 0.6758 - val_loss: 0.8173 - val_accuracy: 0.6712
Epoch 3/100
100/100 [==============================] - 12s 119ms/step - loss: 0.8045 -
accuracy: 0.6834 - val_loss: 0.8115 - val_accuracy: 0.6700
Epoch 4/100
100/100 [==============================] - 12s 115ms/step - loss: 0.8132 -
accuracy: 0.6665 - val_loss: 0.7934 - val_accuracy: 0.6756
Epoch 5/100
100/100 [==============================] - 12s 122ms/step - loss: 0.8047 -
accuracy: 0.6610 - val_loss: 0.7668 - val_accuracy: 0.6794
Epoch 6/100
100/100 [==============================] - 12s 116ms/step - loss: 0.7865 -
accuracy: 0.6644 - val_loss: 0.7415 - val_accuracy: 0.6812
Epoch 7/100
100/100 [==============================] - 11s 111ms/step - loss: 0.7822 -
accuracy: 0.6617 - val_loss: 0.7383 - val_accuracy: 0.6744
Epoch 8/100
100/100 [==============================] - 12s 116ms/step - loss: 0.7599 -
accuracy: 0.6662 - val_loss: 0.7171 - val_accuracy: 0.6888
Epoch 9/100
100/100 [==============================] - 12s 117ms/step - loss: 0.7470 -
```

```
accuracy: 0.6685 - val_loss: 0.7059 - val_accuracy: 0.6837
Epoch 10/100
100/100 [==============================] - 12s 115ms/step - loss: 0.7285 -
accuracy: 0.6759 - val_loss: 0.6827 - val_accuracy: 0.6969
Epoch 11/100
100/100 [==============================] - 12s 115ms/step - loss: 0.6806 -
accuracy: 0.7065 - val_loss: 0.6837 - val_accuracy: 0.6669
Epoch 12/100
100/100 [==============================] - 11s 114ms/step - loss: 0.7051 -
accuracy: 0.6776 - val_loss: 0.6502 - val_accuracy: 0.6862
Epoch 13/100
100/100 [==============================] - 11s 111ms/step - loss: 0.6970 -
accuracy: 0.6901 - val_loss: 0.6487 - val_accuracy: 0.6756
Epoch 14/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6782 -
accuracy: 0.6754 - val_loss: 0.6110 - val_accuracy: 0.7106
Epoch 15/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6445 -
accuracy: 0.7017 - val_loss: 0.6159 - val_accuracy: 0.7125
Epoch 16/100
100/100 [==============================] - 11s 111ms/step - loss: 0.6584 -
accuracy: 0.7042 - val_loss: 0.6519 - val_accuracy: 0.6881
Epoch 17/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6580 -
accuracy: 0.6917 - val_loss: 0.6196 - val_accuracy: 0.6956
Epoch 18/100
100/100 [==============================] - 11s 109ms/step - loss: 0.6433 -
accuracy: 0.7024 - val_loss: 0.6033 - val_accuracy: 0.7038
Epoch 19/100
100/100 [==============================] - 11s 111ms/step - loss: 0.6682 -
accuracy: 0.6827 - val_loss: 0.6144 - val_accuracy: 0.7044
Epoch 20/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6130 -
accuracy: 0.7279 - val_loss: 0.6078 - val_accuracy: 0.7206
Epoch 21/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6449 -
accuracy: 0.7048 - val_loss: 0.5783 - val_accuracy: 0.7381
Epoch 22/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6523 -
accuracy: 0.6854 - val_loss: 0.5921 - val_accuracy: 0.7262
Epoch 23/100
100/100 [==============================] - 11s 111ms/step - loss: 0.6264 -
accuracy: 0.7128 - val_loss: 0.5878 - val_accuracy: 0.7312
Epoch 24/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6380 -
accuracy: 0.7121 - val_loss: 0.5761 - val_accuracy: 0.7419
Epoch 25/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6196 -
```

```
accuracy: 0.7250 - val_loss: 0.5553 - val_accuracy: 0.7556
Epoch 26/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6225 -
accuracy: 0.7127 - val_loss: 0.5789 - val_accuracy: 0.7350
Epoch 27/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6297 -
accuracy: 0.7243 - val_loss: 0.5608 - val_accuracy: 0.7437
Epoch 28/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6106 -
accuracy: 0.7332 - val_loss: 0.5649 - val_accuracy: 0.7487
Epoch 29/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6110 -
accuracy: 0.7042 - val_loss: 0.5674 - val_accuracy: 0.7419
Epoch 30/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6052 -
accuracy: 0.7358 - val_loss: 0.5522 - val_accuracy: 0.7575
Epoch 31/100
100/100 [==============================] - 11s 112ms/step - loss: 0.5990 -
accuracy: 0.7334 - val_loss: 0.5519 - val_accuracy: 0.7425
Epoch 32/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6087 -
accuracy: 0.7033 - val_loss: 0.5609 - val_accuracy: 0.7500
Epoch 33/100
100/100 [==============================] - 11s 112ms/step - loss: 0.6143 -
accuracy: 0.7088 - val_loss: 0.5594 - val_accuracy: 0.7550
Epoch 34/100
100/100 [==============================] - 12s 116ms/step - loss: 0.6136 -
accuracy: 0.7224 - val_loss: 0.5524 - val_accuracy: 0.7400
Epoch 35/100
100/100 [==============================] - 11s 112ms/step - loss: 0.5905 -
accuracy: 0.7323 - val_loss: 0.5398 - val_accuracy: 0.7638
Epoch 36/100
100/100 [==============================] - 11s 111ms/step - loss: 0.6116 -
accuracy: 0.7174 - val_loss: 0.5394 - val_accuracy: 0.7581
Epoch 37/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5938 -
accuracy: 0.7259 - val_loss: 0.5455 - val_accuracy: 0.7538
Epoch 38/100
100/100 [==============================] - 11s 111ms/step - loss: 0.6153 -
accuracy: 0.7274 - val_loss: 0.5357 - val_accuracy: 0.7575
Epoch 39/100
100/100 [==============================] - 11s 110ms/step - loss: 0.6060 -
accuracy: 0.7330 - val_loss: 0.5393 - val_accuracy: 0.7563
Epoch 40/100
100/100 [==============================] - 11s 111ms/step - loss: 0.6075 -
accuracy: 0.7092 - val_loss: 0.5671 - val_accuracy: 0.7437
Epoch 41/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5789 -
```

```
accuracy: 0.7405 - val_loss: 0.5893 - val_accuracy: 0.7375
Epoch 42/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5712 -
accuracy: 0.7513 - val_loss: 0.5270 - val_accuracy: 0.7750
Epoch 43/100
100/100 [==============================] - 11s 111ms/step - loss: 0.6171 -
accuracy: 0.7214 - val_loss: 0.5326 - val_accuracy: 0.7669
Epoch 44/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5878 -
accuracy: 0.7387 - val_loss: 0.5476 - val_accuracy: 0.7506
Epoch 45/100
100/100 [==============================] - 11s 111ms/step - loss: 0.6022 -
accuracy: 0.7390 - val_loss: 0.5414 - val_accuracy: 0.7563
Epoch 46/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5821 -
accuracy: 0.7488 - val_loss: 0.5314 - val_accuracy: 0.7588
Epoch 47/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5806 -
accuracy: 0.7413 - val_loss: 0.5283 - val_accuracy: 0.7656
Epoch 48/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5696 -
accuracy: 0.7502 - val_loss: 0.5179 - val_accuracy: 0.7619
Epoch 49/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5821 -
accuracy: 0.7324 - val_loss: 0.5244 - val_accuracy: 0.7669
Epoch 50/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5829 -
accuracy: 0.7485 - val_loss: 0.5414 - val_accuracy: 0.7663
Epoch 51/100
100/100 [==============================] - 11s 112ms/step - loss: 0.5866 -
accuracy: 0.7424 - val_loss: 0.5173 - val_accuracy: 0.7725
Epoch 52/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5597 -
accuracy: 0.7463 - val_loss: 0.5271 - val_accuracy: 0.7644
Epoch 53/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5942 -
accuracy: 0.7307 - val_loss: 0.5129 - val_accuracy: 0.7744
Epoch 54/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5896 -
accuracy: 0.7314 - val_loss: 0.5101 - val_accuracy: 0.7794
Epoch 55/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5595 -
accuracy: 0.7499 - val_loss: 0.5392 - val_accuracy: 0.7487
Epoch 56/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5729 -
accuracy: 0.7468 - val_loss: 0.5432 - val_accuracy: 0.7544
Epoch 57/100
100/100 [==============================] - 11s 112ms/step - loss: 0.5632 -
```

```
accuracy: 0.7448 - val_loss: 0.4985 - val_accuracy: 0.7800
Epoch 58/100
100/100 [==============================] - 11s 112ms/step - loss: 0.5652 -
accuracy: 0.7560 - val_loss: 0.5221 - val_accuracy: 0.7669
Epoch 59/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5829 -
accuracy: 0.7239 - val_loss: 0.5089 - val_accuracy: 0.7694
Epoch 60/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5826 -
accuracy: 0.7381 - val_loss: 0.5054 - val_accuracy: 0.7831
Epoch 61/100
100/100 [==============================] - 11s 112ms/step - loss: 0.5683 -
accuracy: 0.7509 - val_loss: 0.5249 - val_accuracy: 0.7619
Epoch 62/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5547 -
accuracy: 0.7524 - val_loss: 0.5034 - val_accuracy: 0.7788
Epoch 63/100
100/100 [==============================] - 11s 112ms/step - loss: 0.5529 -
accuracy: 0.7458 - val_loss: 0.5000 - val_accuracy: 0.7763
Epoch 64/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5528 -
accuracy: 0.7494 - val_loss: 0.5254 - val_accuracy: 0.7675
Epoch 65/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5585 -
accuracy: 0.7490 - val_loss: 0.4941 - val_accuracy: 0.7812
Epoch 66/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5547 -
accuracy: 0.7511 - val_loss: 0.4922 - val_accuracy: 0.7788
Epoch 67/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5540 -
accuracy: 0.7615 - val_loss: 0.5056 - val_accuracy: 0.7862
Epoch 68/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5598 -
accuracy: 0.7401 - val_loss: 0.4791 - val_accuracy: 0.7869
Epoch 69/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5411 -
accuracy: 0.7593 - val_loss: 0.4973 - val_accuracy: 0.7837
Epoch 70/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5554 -
accuracy: 0.7529 - val_loss: 0.4862 - val_accuracy: 0.7800
Epoch 71/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5698 -
accuracy: 0.7493 - val_loss: 0.4747 - val_accuracy: 0.7937
Epoch 72/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5664 -
accuracy: 0.7317 - val_loss: 0.4811 - val_accuracy: 0.7919
Epoch 73/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5534 -
```

```
accuracy: 0.7394 - val_loss: 0.4754 - val_accuracy: 0.7975
Epoch 74/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5084 -
accuracy: 0.7782 - val_loss: 0.4670 - val_accuracy: 0.8000
Epoch 75/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5417 -
accuracy: 0.7628 - val_loss: 0.4808 - val_accuracy: 0.7956
Epoch 76/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5528 -
accuracy: 0.7458 - val_loss: 0.4905 - val_accuracy: 0.7794
Epoch 77/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5407 -
accuracy: 0.7647 - val_loss: 0.4765 - val_accuracy: 0.7862
Epoch 78/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5265 -
accuracy: 0.7591 - val_loss: 0.4688 - val_accuracy: 0.7881
Epoch 79/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5112 -
accuracy: 0.7800 - val_loss: 0.4798 - val_accuracy: 0.7831
Epoch 80/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5181 -
accuracy: 0.7623 - val_loss: 0.4721 - val_accuracy: 0.7881
Epoch 81/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5318 -
accuracy: 0.7644 - val_loss: 0.4583 - val_accuracy: 0.8031
Epoch 82/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5071 -
accuracy: 0.7814 - val_loss: 0.4860 - val_accuracy: 0.7781
Epoch 83/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5160 -
accuracy: 0.7815 - val_loss: 0.5427 - val_accuracy: 0.7581
Epoch 84/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5431 -
accuracy: 0.7606 - val_loss: 0.4740 - val_accuracy: 0.7869
Epoch 85/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5195 -
accuracy: 0.7743 - val_loss: 0.4657 - val_accuracy: 0.7837
Epoch 86/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5346 -
accuracy: 0.7605 - val_loss: 0.4598 - val_accuracy: 0.7944
Epoch 87/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5028 -
accuracy: 0.7759 - val_loss: 0.4592 - val_accuracy: 0.8006
Epoch 88/100
100/100 [==============================] - 11s 110ms/step - loss: 0.5186 -
accuracy: 0.7687 - val_loss: 0.4587 - val_accuracy: 0.7994
Epoch 89/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5015 -
```

```
accuracy: 0.7822 - val_loss: 0.4449 - val_accuracy: 0.8031
Epoch 90/100
100/100 [==============================] - 11s 112ms/step - loss: 0.5249 -
accuracy: 0.7682 - val_loss: 0.4493 - val_accuracy: 0.8062
Epoch 91/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5195 -
accuracy: 0.7724 - val_loss: 0.4576 - val_accuracy: 0.8019
Epoch 92/100
100/100 [==============================] - 11s 111ms/step - loss: 0.4969 -
accuracy: 0.7869 - val_loss: 0.4500 - val_accuracy: 0.8094
Epoch 93/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5366 -
accuracy: 0.7698 - val_loss: 0.4435 - val_accuracy: 0.8138
Epoch 94/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5191 -
accuracy: 0.7720 - val_loss: 0.4438 - val_accuracy: 0.8169
Epoch 95/100
100/100 [==============================] - 11s 111ms/step - loss: 0.4957 -
accuracy: 0.7876 - val_loss: 0.4683 - val_accuracy: 0.7950
Epoch 96/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5064 -
accuracy: 0.7822 - val_loss: 0.4726 - val_accuracy: 0.7906
Epoch 97/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5072 -
accuracy: 0.7829 - val_loss: 0.4356 - val_accuracy: 0.8075
Epoch 98/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5142 -
accuracy: 0.7752 - val_loss: 0.4291 - val_accuracy: 0.8200
Epoch 99/100
100/100 [==============================] - 11s 111ms/step - loss: 0.5336 -
accuracy: 0.7623 - val_loss: 0.4490 - val_accuracy: 0.7962
Epoch 100/100
100/100 [==============================] - 11s 111ms/step - loss: 0.4940 -
accuracy: 0.7820 - val_loss: 0.4467 - val_accuracy: 0.8069
```

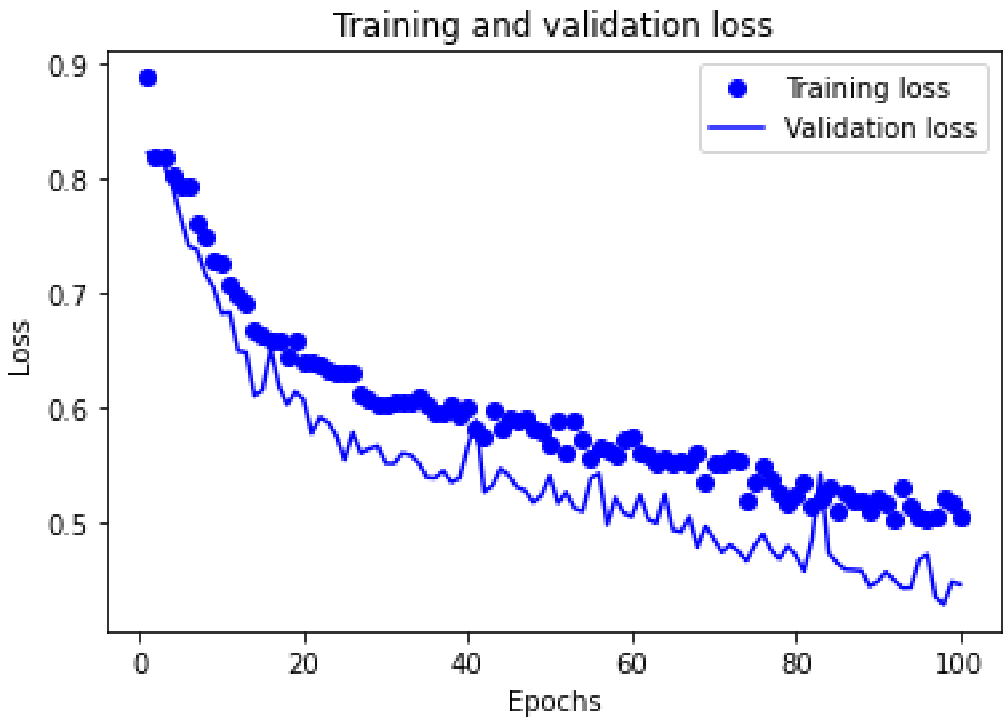
[ ]:
```
model.save(os.path.join(BASE_PATH, 'covid_classifier_result.h5'))
```

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

```
72/72 [==============================] - 10s 140ms/step - loss: 0.4323 -
accuracy: 0.8223
```

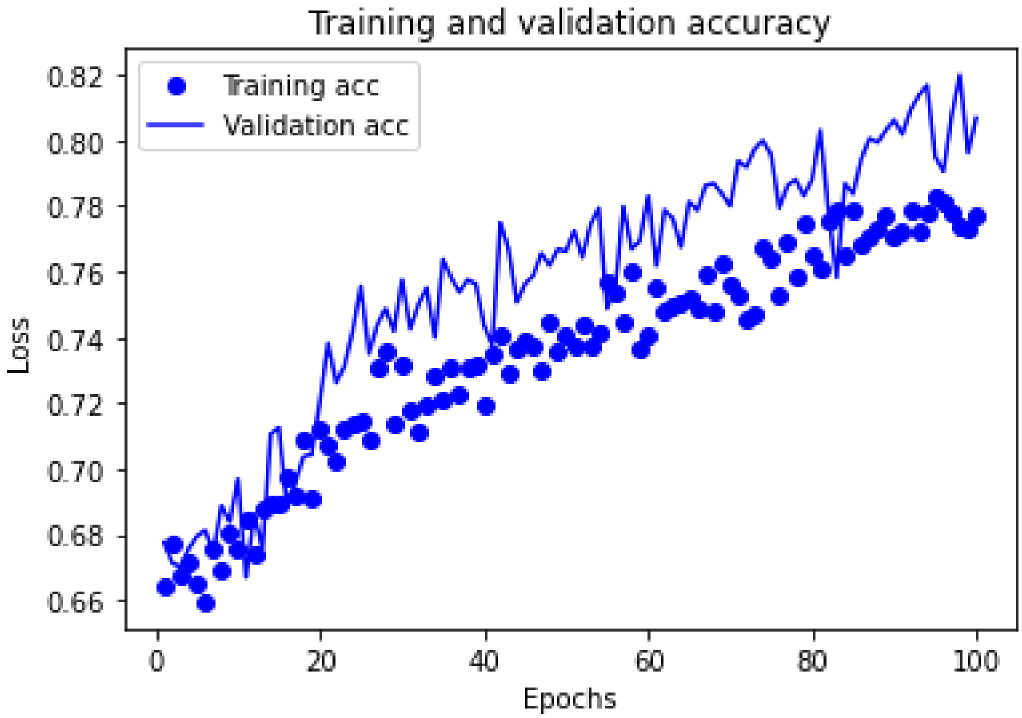
[ ]:
```
import matplotlib.pyplot as plt

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
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0
20
40
60
80
100