COVID-19-Image-Classification-phase2-large_c19_dataset

November 24, 2021

1 Covid19 Image Classification - Phase 2

Abstract: Our Aim is to detect Covid19 from chest X-rays. The covid19 image dataset is small with 251 training and 60 test images belonging to 'NoFinding', 'Covid19' and 'Pneumonia' respectively. This dataset small and is insufficient to generalize. So for the purpoe of our project, in Phase-I we will first use NIH X-ray image data to retrain and finetune pretrained model architecture such as ResNet50V2, MobileNetV2 and VGG16.

Following the results from phase 1, we selected the best model to be _____ and in phase2 we will focus on classifing our target dataset of covid images

[15]:
```
# common imports
import os
import numpy as np
import matplotlib.pylab as plt
import pandas as pd
from glob import glob
from pathlib import Path
from functools import partial

from sklearn.model_selection import train_test_split

os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"
# prevent VRAM occupied
os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'

# TensorFlow 2.0 is required
import tensorflow as tf
from tensorflow import keras
assert tf.__version__ >= "2.0"

from keras.applications.resnet_v2 import preprocess_input
from keras.preprocessing.image import ImageDataGenerator

import warnings
warnings.filterwarnings('ignore')
```

[16]:
```
#change working directory  - as the images are located in data/raw in the␣
 ↪project folder (Execute this cell Once)

if '/notebooks' in os.getcwd():
    os.chdir("../")
    print("set the project directory as working directory")
else:
    print(os.getcwd())
```

```
/home/jay/Temple/myprojects/covid19-image-detection-tfl
```

[17]:
```
# Import functions for trianing the model
%load_ext autoreload
%autoreload 2
import src.models.train_model as train_model

# load tensorboard extension
%reload_ext tensorboard
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[18]:
```
# Constants
SEED =42
IMAGE_SIZE = (224,224)
IMAGE_SHAPE = (224,224,3)
BATCH_SIZE = 32
SHUFFLE = True
TARGET_WIDTH= 224
TARGET_HEGIHT =224
NUM_CLASSES = 3 # number of ClassesNUM
NUM_EPOCHS = 10
log_folder = 'logs' # logs folder
```

1.1 Preprocess Images

[19]:
```
BATCH_SIZE =1

covid_image_train_data_gen = ImageDataGenerator(
    preprocessing_function= preprocess_input,
    validation_split=0.2,
    featurewise_center=False,  # set input mean to 0 over the dataset
    samplewise_center=True, #Boolean. Set each sample mean to 0.
    samplewise_std_normalization = False, #Boolean. Divide each input by its␣
 ↪std.
    featurewise_std_normalization=False, # divide inputs by std of the dataset
    horizontal_flip = True, #Boolean. Randomly flip inputs horizontally.
```

```
    vertical_flip = False,  #Boolean. Randomly flip inputs vertically.
    zca_whitening=False,  # apply ZCA whitening
    height_shift_range= 0.05, #float: fraction of total height, if < 1, or␣
↪pixels if >= 1.
    width_shift_range=0.1,  #float: fraction of total height, if < 1, or pixels␣
↪if >= 1.
    rotation_range=20, #Int. Degree range for random rotations. 0 -180 degrees
    shear_range = 0.1, #Float. Shear Intensity (Shear angle in␣
↪counter-clockwise direction in degrees)
    fill_mode = 'nearest', #One of {"constant", "nearest", "reflect" or "wrap"}.
↪ Default is 'nearest'.
    zoom_range=0.15) #Float or [lower, upper]. Range for random zoom. If a␣
↪float, [lower, upper] = [1-zoom_range, 1+zoom_range]


# covid_image_test_data_gen = ImageDataGenerator(
#     preprocessing_function= preprocess_input,
#     featurewise_center=False,  # set input mean to 0 over the dataset
#     samplewise_center=True, #Boolean. Set each sample mean to 0.
#     samplewise_std_normalization = False, #Boolean. Divide each input by its␣
↪std.
#     featurewise_std_normalization=False, # divide inputs by std of the dataset
#     horizontal_flip = True, #Boolean. Randomly flip inputs horizontally.
#     vertical_flip = False,  #Boolean. Randomly flip inputs vertically.
#     zca_whitening=False,  # apply ZCA whitening
#     height_shift_range= 0.05, #float: fraction of total height, if < 1, or␣
↪pixels if >= 1.
#     width_shift_range=0.1,  #float: fraction of total height, if < 1, or␣
↪pixels if >= 1.
#     rotation_range=20, #Int. Degree range for random rotations. 0 -180 degrees
#     shear_range = 0.1, #Float. Shear Intensity (Shear angle in␣
↪counter-clockwise direction in degrees)
#     fill_mode = 'nearest', #One of {"constant", "nearest", "reflect" or␣
↪"wrap"}. Default is 'nearest'.
#     zoom_range=0.15) #Float or [lower, upper]. Range for random zoom. If a␣
↪float, [lower, upper] = [1-zoom_range, 1+zoom_range]


covid_train_ds = covid_image_train_data_gen.flow_from_directory(
  'data/raw/covid19-images',
  subset="training",
  class_mode='categorical',
  #classes=['Covid19', 'Normal', 'Pneumonia'],
  seed=SEED,
  target_size=IMAGE_SIZE,
  batch_size=BATCH_SIZE)
```

```
covid_valid_ds = covid_image_train_data_gen.flow_from_directory(
  'data/raw/covid19-images',
  subset="validation",
  #classes=['Covid19', 'Normal', 'Pneumonia'],
  class_mode='categorical',
  seed=SEED,
  target_size=IMAGE_SIZE,
  batch_size=BATCH_SIZE)

# covid_test_generator = covid_image_test_data_gen.flow_from_directory(
#   'data/raw/Covid19-dataset/test/',
#   class_mode='categorical',
#   classes=['Covid', 'Normal', 'Viral Pneumonia'],
#   seed=SEED,
#   target_size=IMAGE_SIZE,
#   batch_size=BATCH_SIZE)
```

```
Found 2995 images belonging to 3 classes.
Found 748 images belonging to 3 classes.
```

1.2 Reload and train the Saved Model

[27]:
```
def train_validate_and_test_reconstructed_model(saved_model_path: str,
                                                model_name: str,
                                                covid_train_ds,
                                                covid_valid_ds,
                                                #freeze_layers: bool = False,
                                                num_classes: int = 3,
                                                num_epochs: int = 50,
                                                batch_size: int = 32,
                                                activation_func: str =␣
 ↪'softmax',
                                                learning_rate: float = 0.001,
                                                ft_learning_rate:float = 0.
 ↪00001,
                                                fine_tune_at_layer: int = 0):

    # It can be used to reconstruct the model identically.
    reconstructed_model = keras.models.load_model(saved_model_path)


    new_layer = reconstructed_model.layers[-2].output
    new_layer = keras.layers.Dense(256, activation="relu")(new_layer)
    new_layer = keras.layers.Dropout(0.3)(new_layer)
    output = keras.layers.Dense(num_classes, activation=activation_func,␣
 ↪name='output')(new_layer)
```

```
    new_model = keras.Model(inputs = reconstructed_model.input, outputs =␣
↪output)
    new_model = train_model.compile_classifier(new_model, learning_rate)


    new_model_history = train_model.fit_model(new_model, covid_train_ds,␣
↪covid_valid_ds,num_epochs=num_epochs,batch_size=batch_size, patience=10)

    print(f'{model_name} Accuracy and Loss plots')
    train_model.plot_accuracy_and_loss(new_model_history)

    print("\n")
    #fine_tune model_name
    new_model_ft = train_model.
↪fine_tune_model(new_model,ft_learning_rate,optimizer='Adam',fine_tune_at_layer=fine_tune_at

    print("\n")
    print(f'Fine-Tuned {model_name} Training and Validation: ')
    new_model_ft_history = train_model.fit_model(new_model_ft, covid_train_ds,
            covid_valid_ds,
            num_epochs=num_epochs,batch_size=batch_size)

    print(f'Fine-Tuned {model_name} Accuracy and Loss plots')
    train_model.plot_accuracy_and_loss(new_model_ft_history)
```

2 Covid19 Image Classification - Phase 2

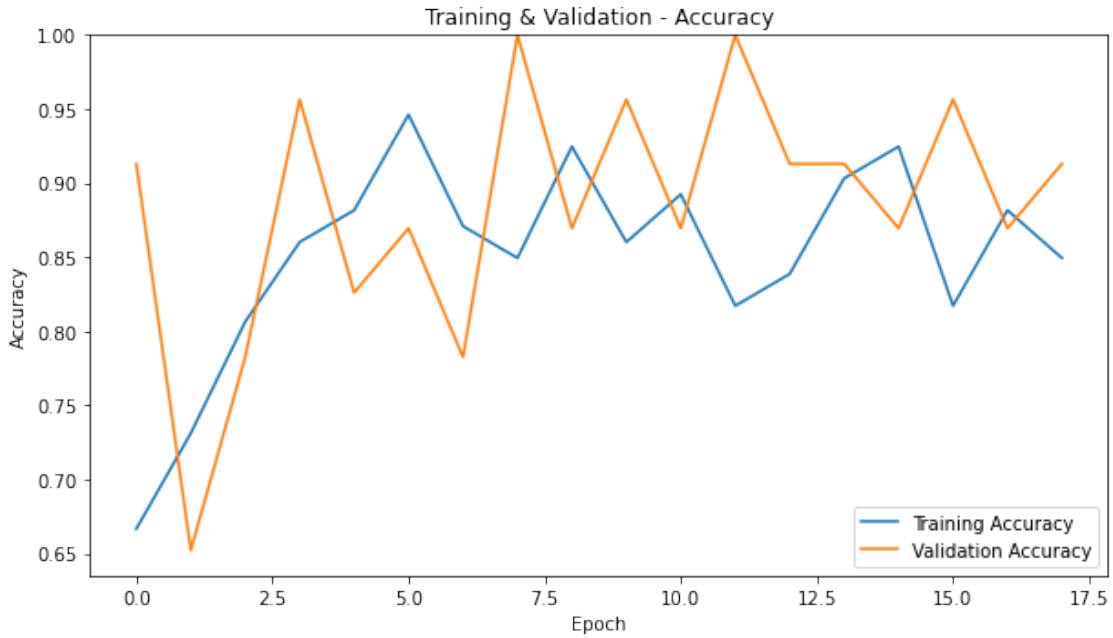
2.1 Experiement 1 - Classification (softmax)

[28]:
```
train_validate_and_test_reconstructed_model('models/ResNet50V2exp1.h5',␣
↪'ResNet50V2', covid_train_ds, covid_valid_ds)
```

```
Epoch 1/50
93/93 [==============================] - 5s 37ms/step - loss: 24.3086 -
accuracy: 0.6667 - val_loss: 2.0511 - val_accuracy: 0.9130

Epoch 00001: val_loss improved from inf to 2.05106, saving model to
models/my_model.h5
Epoch 2/50
93/93 [==============================] - 2s 24ms/step - loss: 20.4335 -
accuracy: 0.7312 - val_loss: 24.0654 - val_accuracy: 0.6522

Epoch 00002: val_loss did not improve from 2.05106
Epoch 3/50
93/93 [==============================] - 2s 25ms/step - loss: 13.6937 -
accuracy: 0.8065 - val_loss: 9.3051 - val_accuracy: 0.7826
```

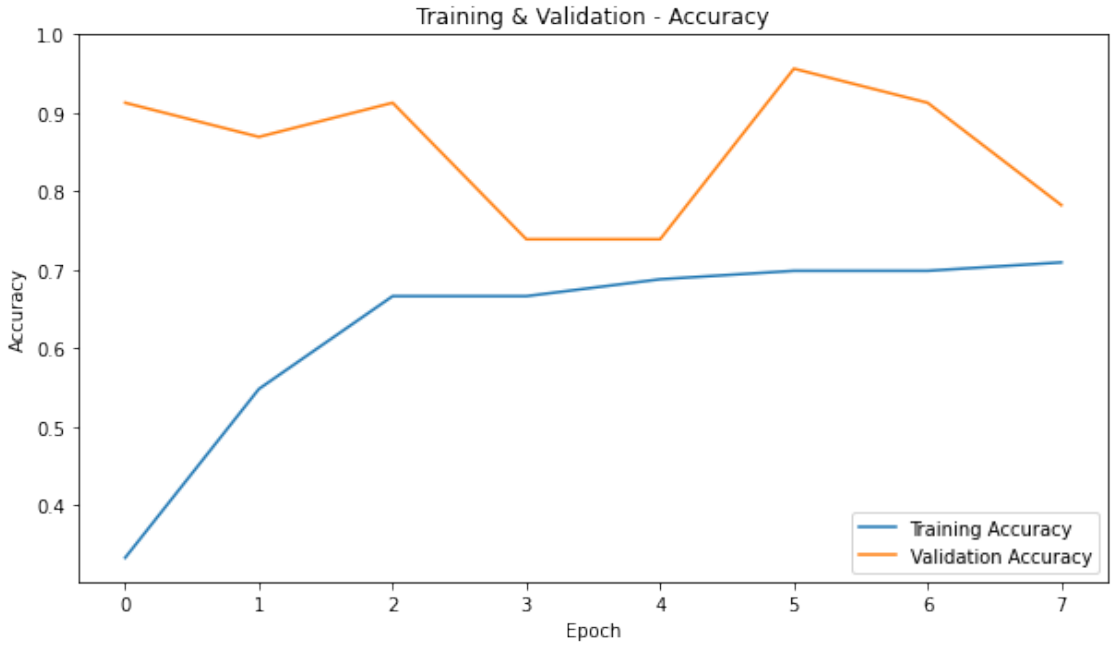
```
Epoch 00003: val_loss did not improve from 2.05106
Epoch 4/50
93/93 [==============================] - 2s 25ms/step - loss: 7.9056 - accuracy:
0.8602 - val_loss: 0.5326 - val_accuracy: 0.9565

Epoch 00004: val_loss improved from 2.05106 to 0.53256, saving model to
models/my_model.h5
Epoch 5/50
93/93 [==============================] - 2s 25ms/step - loss: 5.4650 - accuracy:
0.8817 - val_loss: 11.4568 - val_accuracy: 0.8261

Epoch 00005: val_loss did not improve from 0.53256
Epoch 6/50
93/93 [==============================] - 2s 24ms/step - loss: 3.8362 - accuracy:
0.9462 - val_loss: 6.3909 - val_accuracy: 0.8696

Epoch 00006: val_loss did not improve from 0.53256
Epoch 7/50
93/93 [==============================] - 2s 25ms/step - loss: 4.6690 - accuracy:
0.8710 - val_loss: 12.0049 - val_accuracy: 0.7826

Epoch 00007: val_loss did not improve from 0.53256
Epoch 8/50
93/93 [==============================] - 2s 24ms/step - loss: 10.1658 -
accuracy: 0.8495 - val_loss: 1.7728e-05 - val_accuracy: 1.0000

Epoch 00008: val_loss improved from 0.53256 to 0.00002, saving model to
models/my_model.h5
Epoch 9/50
93/93 [==============================] - 2s 25ms/step - loss: 4.4288 - accuracy:
0.9247 - val_loss: 0.3658 - val_accuracy: 0.8696

Epoch 00009: val_loss did not improve from 0.00002
Epoch 10/50
93/93 [==============================] - 2s 24ms/step - loss: 13.4189 -
accuracy: 0.8602 - val_loss: 4.9259 - val_accuracy: 0.9565

Epoch 00010: val_loss did not improve from 0.00002
Epoch 11/50
93/93 [==============================] - 2s 25ms/step - loss: 5.0701 - accuracy:
0.8925 - val_loss: 3.5642 - val_accuracy: 0.8696

Epoch 00011: val_loss did not improve from 0.00002
Epoch 12/50
93/93 [==============================] - 2s 24ms/step - loss: 10.2491 -
accuracy: 0.8172 - val_loss: 4.4649e-05 - val_accuracy: 1.0000
```

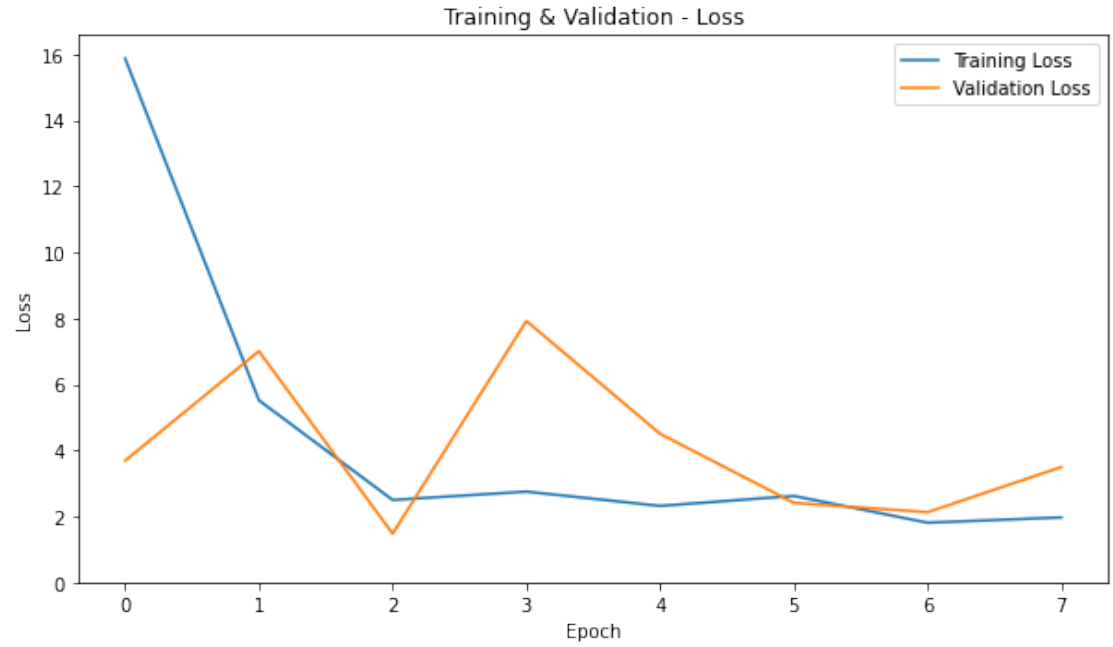
```
Epoch 00012: val_loss did not improve from 0.00002
Epoch 13/50
93/93 [==============================] - 2s 25ms/step - loss: 5.9266 - accuracy:
0.8387 - val_loss: 0.9011 - val_accuracy: 0.9130

Epoch 00013: val_loss did not improve from 0.00002
Epoch 14/50
93/93 [==============================] - 2s 25ms/step - loss: 3.8189 - accuracy:
0.9032 - val_loss: 1.7077 - val_accuracy: 0.9130

Epoch 00014: val_loss did not improve from 0.00002
Epoch 15/50
93/93 [==============================] - 2s 24ms/step - loss: 2.5285 - accuracy:
0.9247 - val_loss: 4.7214 - val_accuracy: 0.8696

Epoch 00015: val_loss did not improve from 0.00002
Epoch 16/50
93/93 [==============================] - 2s 24ms/step - loss: 9.0155 - accuracy:
0.8172 - val_loss: 1.3423 - val_accuracy: 0.9565

Epoch 00016: val_loss did not improve from 0.00002
Epoch 17/50
93/93 [==============================] - 2s 23ms/step - loss: 7.1489 - accuracy:
0.8817 - val_loss: 5.9851 - val_accuracy: 0.8696

Epoch 00017: val_loss did not improve from 0.00002
Epoch 18/50
93/93 [==============================] - 2s 25ms/step - loss: 4.8928 - accuracy:
0.8495 - val_loss: 2.3790 - val_accuracy: 0.9130

Epoch 00018: val_loss did not improve from 0.00002
Restoring model weights from the end of the best epoch.
Epoch 00018: early stopping
ResNet50V2 Accuracy and Loss plots
```




```
Fine-Tuned ResNet50V2 Training and Validation:
Epoch 1/50
93/93 [==============================] - 8s 61ms/step - loss: 15.8717 -
accuracy: 0.3333 - val_loss: 3.6887 - val_accuracy: 0.9130
```

```
Epoch 00001: val_loss improved from inf to 3.68873, saving model to
models/my_model.h5
Epoch 2/50
93/93 [==============================] - 4s 47ms/step - loss: 5.5189 - accuracy:
0.5484 - val_loss: 7.0053 - val_accuracy: 0.8696

Epoch 00002: val_loss did not improve from 3.68873
Epoch 3/50
93/93 [==============================] - 4s 45ms/step - loss: 2.4983 - accuracy:
0.6667 - val_loss: 1.4767 - val_accuracy: 0.9130

Epoch 00003: val_loss improved from 3.68873 to 1.47672, saving model to
models/my_model.h5
Epoch 4/50
93/93 [==============================] - 4s 47ms/step - loss: 2.7512 - accuracy:
0.6667 - val_loss: 7.9211 - val_accuracy: 0.7391

Epoch 00004: val_loss did not improve from 1.47672
Epoch 5/50
93/93 [==============================] - 4s 45ms/step - loss: 2.3185 - accuracy:
0.6882 - val_loss: 4.5016 - val_accuracy: 0.7391

Epoch 00005: val_loss did not improve from 1.47672
Epoch 6/50
93/93 [==============================] - 4s 47ms/step - loss: 2.6205 - accuracy:
0.6989 - val_loss: 2.4128 - val_accuracy: 0.9565

Epoch 00006: val_loss did not improve from 1.47672
Epoch 7/50
93/93 [==============================] - 4s 46ms/step - loss: 1.8091 - accuracy:
0.6989 - val_loss: 2.1276 - val_accuracy: 0.9130

Epoch 00007: val_loss did not improve from 1.47672
Epoch 8/50
93/93 [==============================] - 4s 46ms/step - loss: 1.9703 - accuracy:
0.7097 - val_loss: 3.4960 - val_accuracy: 0.7826

Epoch 00008: val_loss did not improve from 1.47672
Restoring model weights from the end of the best epoch.
Epoch 00008: early stopping
Fine-Tuned ResNet50V2 Accuracy and Loss plots
```





2.2 Training and Validating ResNet50V2 on Covid19 dataset with no transfer learning

[29]:
```
# Constants
SEED =42
IMAGE_SIZE = (224,224)
IMAGE_SHAPE = (224,224,3)
```

```
BATCH_SIZE = 32
SHUFFLE = True
NUM_CLASSES = 15 # number of ClassesNUM
NUM_EPOCHS = 10
PRETRAINED_MODELS = ['ResNet50V2', 'MobileNetV2', 'VGG16']

# Train and validate function
def train_and_validate_model(model_name,
                             train_generator,
                             valid_generator,
                             save_model_filepath: str,
                             logs_dir: str,
                             patience: int =5,
                             freeze_layers:bool = True,
                             activation: str = 'softmax',
                             learning_rate: float =0.01,
                             fine_tune_learning_rate: float = 0.0001,
                             fine_tune_at_layer:int = 186,
                             num_epochs:int = NUM_EPOCHS,
                             num_classes: int = NUM_CLASSES,
                             batch_size: int = BATCH_SIZE,
                             input_shape: int = IMAGE_SHAPE):

    print(model_name)

    my_model = train_model.
↪get_base_model_with_new_toplayer(base_model=model_name,
                                                              freeze_layers =␣
↪freeze_layers,
                                                              num_classes =␣
↪num_classes,
                                                             ␣
↪activation_func=activation,
                                                              learning_rate =␣
↪learning_rate,
                                                              input_shape =␣
↪input_shape)

    my_model_history = train_model.fit_model(my_model,
                                             train_generator,
                                             valid_generator,
                                             num_epochs=num_epochs,
                                             batch_size=batch_size,
                                            ␣
↪checkpoint_filepath=save_model_filepath,
                                             logs_dir = logs_dir,
```

```
                                            patience = patience)

    print(f'{model_name} Accuracy and Loss plots')
    train_model.plot_accuracy_and_loss(my_model_history)


    print("\n")
    #fine_tune model_name
    model_ft = train_model.fine_tune_model(my_model,
                                           fine_tune_learning_rate,
                                           optimizer='Adam',
                                          ␣
↪fine_tune_at_layer=fine_tune_at_layer,
                                           activation_func=activation)


    print("\n")
    print(f'Fine-Tuned {model_name} Training and Validation: ')
    model_ft_history = train_model.fit_model(model_ft, train_generator,
            valid_generator, num_epochs=num_epochs,batch_size=batch_size)
    print(f'Fine-Tuned {model_name} Accuracy and Loss plots')
    train_model.plot_accuracy_and_loss(model_ft_history)
    return model_ft
```

[33]:
```
model_name = 'ResNet50V2'
save_model_filepath = 'models/Covid19'+ model_name + 'exp1.h5'
logs_dir = 'logs/fit/Covid19ResNet50V2exp1'
model = train_and_validate_model(model_name = model_name,
                                 train_generator=covid_train_ds,
                                 valid_generator=covid_valid_ds,
                                 save_model_filepath=save_model_filepath,
                                 logs_dir=logs_dir,
                                 num_classes=3,
                                 num_epochs=50,
                                 patience=10)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_8419/3251177164.py in <module>
      2 save_model_filepath = 'models/Covid19'+ model_name + 'exp1.h5'
      3 logs_dir = 'logs/fit/Covid19ResNet50V2exp1'
----> 4 model = train_and_validate_model(model_name = model_name,
      5                                 train_generator=covid_train_ds,
      6                                 valid_generator=covid_valid_ds,
```

```
TypeError: train_and_validate_model() got an unexpected keyword argument␣
↪'patience'
```

[8]:
```
# clear gpu memory
from numba import cuda
device = cuda.get_current_device()
device.reset()
```

[ ]:
```

```