Final code - Project - ANLY 535

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator,load_img, img_to_array
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPooling2D,GlobalAveragePooling2D
from keras.layers import Activation, Dropout, BatchNormalization, Flatten, Dense, AvgPool2D,MaxPool2D
from keras.applications.vgg16 import VGG16, preprocess_input
from keras.optimizers import Adam, SGD
import tensorflow as tf
from tensorflow.keras.preprocessing import image
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [ ]:
```
path='/content/gdrive/MyDrive/COVID-19_Radiography_Dataset'
savepath = '/content/gdrive/MyDrive/COVID-19_Radiography_Dataset_pp'
```

In [ ]:
```
from skimage.filters.rank import equalize
from skimage.morphology import disk
import os
import skimage.io as io
import skimage.color as color
import skimage.transform as trf
import numpy as np
for directory in sorted(os.listdir(path)):
        data_path = path +"/"+ directory
        save_path = savepath +"/"+ directory
        for im in os.listdir(data_path)[:]:
            image = io.imread(f"{data_path}/{im}")
            #image - np.array(image)
            image = equalize(image, disk(70))
            #image = color.rgb2gray(image)
            #image = trf.resize(image, (image_sz, image_sz))
            #images.append(image)
            #labels.append(directory)
            io.imsave(f"{save_path}/{im}", image)
```

In [ ]:
```
image_prepro_path = '/content/gdrive/MyDrive/COVID-19_Radiography_Dataset_pp'
classes=["COVID",  "Normal"]
num_classes = len(classes)
batch_size=32
target_size=(299, 299)
# Splitting the dataset into 70:30 by ImageGenerator
data_gen = ImageDataGenerator(rotation_range=15, zoom_range=0.2, width_shift_range=0.2, height_shift_range=0.2, shear_rar

# Loading images to train generator
train_set = data_gen.flow_from_directory(directory=image_prepro_path,
                                         target_size=target_size,
                                         class_mode='categorical',
                                         subset='training',
                                         shuffle=True, classes=classes,
                                         batch_size=batch_size,
                                         color_mode="grayscale")
# Loading images to test generator
test_set = data_gen.flow_from_directory(directory=image_prepro_path,
                                         target_size=target_size,
                                         class_mode='categorical',
                                         subset='validation',
                                         shuffle=True, classes=classes,
                                         batch_size=batch_size,
                                         color_mode="grayscale")
```

```
Found 9674 images belonging to 2 classes.
Found 4144 images belonging to 2 classes.
```

In [ ]:
```
def covid_model():

    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(299, 299, 1)))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(BatchNormalization(momentum=0.9))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(BatchNormalization(momentum=0.9))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(BatchNormalization(momentum=0.9))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
   # model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(128, activation='relu'))
```

```
    model.add(Dense(num_classes, activation='softmax'))
    # compile model
    #optimizer_sgd = SGD(lr=0.01, momentum=0.9)
    opt=Adam(learning_rate=0.008)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=["accuracy"])
    model.summary()
    return model
```

In [ ]:
```
#checkpoint
import tensorflow
from tensorflow.keras.callbacks import ModelCheckpoint
early_stop = tensorflow.keras.callbacks.EarlyStopping(monitor='val_loss', patience= 10)
callbacks_list= ModelCheckpoint('/content/gdrive/MyDrive/ANLY535_Models/Weights-{epoch:03d}--{val_loss:.5f}.hdf5', monitc
callbacks = [early_stop, callbacks_list]
```

In [ ]:
```
covidmodel = covid_model()
# fitting the model
history = covidmodel.fit_generator(train_set, steps_per_epoch=len(train_set) // batch_size, validation_steps=len(test_set
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 297, 297, 32)      320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 295, 295, 32)      9248
_________________________________________________________________
batch_normalization (BatchNo (None, 295, 295, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 147, 147, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 145, 145, 64)      18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 143, 143, 64)      36928
_________________________________________________________________
batch_normalization_1 (Batch (None, 143, 143, 64)      256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 71, 71, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 69, 69, 128)       73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 67, 67, 128)       147584
_________________________________________________________________
batch_normalization_2 (Batch (None, 67, 67, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 33, 33, 128)       0
_________________________________________________________________
```

```
flatten (Flatten)            (None, 139392)            0         
_________________________________________________________________
dense (Dense)                (None, 128)               17842304  
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258       
=================================================================
Total params: 18,129,890
Trainable params: 18,129,442
Non-trainable params: 448
_________________________________________________________________
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py:1915: UserWarning: `Model.fit_generator` is deprecated an
d will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/100
9/9 - 231s - loss: 69.1473 - accuracy: 0.6007 - val_loss: 66.7738 - val_accuracy: 0.6250
Epoch 2/100
9/9 - 176s - loss: 8.2342 - accuracy: 0.6146 - val_loss: 42.0559 - val_accuracy: 0.2344
Epoch 3/100
9/9 - 174s - loss: 3.7898 - accuracy: 0.6701 - val_loss: 6.9383 - val_accuracy: 0.5859
Epoch 4/100
9/9 - 162s - loss: 5.7808 - accuracy: 0.6701 - val_loss: 1.6611 - val_accuracy: 0.6875
Epoch 5/100
9/9 - 155s - loss: 1.8932 - accuracy: 0.6979 - val_loss: 0.8715 - val_accuracy: 0.7891
Epoch 6/100
9/9 - 161s - loss: 1.7165 - accuracy: 0.6875 - val_loss: 0.8035 - val_accuracy: 0.7734
Epoch 7/100
9/9 - 158s - loss: 1.2044 - accuracy: 0.7257 - val_loss: 0.6898 - val_accuracy: 0.6875
Epoch 8/100
9/9 - 153s - loss: 0.8640 - accuracy: 0.7292 - val_loss: 0.7870 - val_accuracy: 0.7734
Epoch 9/100
9/9 - 144s - loss: 0.6553 - accuracy: 0.7431 - val_loss: 0.8317 - val_accuracy: 0.6953
Epoch 10/100
9/9 - 131s - loss: 0.9062 - accuracy: 0.7049 - val_loss: 0.6439 - val_accuracy: 0.7422
Epoch 11/100
9/9 - 136s - loss: 0.6554 - accuracy: 0.6840 - val_loss: 0.6541 - val_accuracy: 0.7031
Epoch 12/100
9/9 - 129s - loss: 0.7534 - accuracy: 0.7535 - val_loss: 0.6643 - val_accuracy: 0.7891
Epoch 13/100
9/9 - 133s - loss: 0.5671 - accuracy: 0.7639 - val_loss: 0.6682 - val_accuracy: 0.6797
Epoch 14/100
9/9 - 120s - loss: 0.6372 - accuracy: 0.7188 - val_loss: 0.6886 - val_accuracy: 0.6953
Epoch 15/100
9/9 - 122s - loss: 0.6424 - accuracy: 0.6632 - val_loss: 0.7424 - val_accuracy: 0.7031
Epoch 16/100
9/9 - 110s - loss: 0.7041 - accuracy: 0.7083 - val_loss: 0.6075 - val_accuracy: 0.7031
Epoch 17/100
9/9 - 111s - loss: 0.6063 - accuracy: 0.7014 - val_loss: 0.5735 - val_accuracy: 0.7578
Epoch 18/100
9/9 - 102s - loss: 0.6522 - accuracy: 0.7396 - val_loss: 0.6335 - val_accuracy: 0.6719
```

```
Epoch 19/100
9/9 - 111s - loss: 0.7350 - accuracy: 0.7014 - val_loss: 0.6588 - val_accuracy: 0.7031
Epoch 20/100
9/9 - 100s - loss: 0.6394 - accuracy: 0.6910 - val_loss: 0.5658 - val_accuracy: 0.7578
Epoch 21/100
9/9 - 105s - loss: 0.6352 - accuracy: 0.7188 - val_loss: 0.5705 - val_accuracy: 0.7422
Epoch 22/100
9/9 - 91s - loss: 0.5974 - accuracy: 0.7118 - val_loss: 0.6001 - val_accuracy: 0.7109
Epoch 23/100
9/9 - 82s - loss: 0.6043 - accuracy: 0.7083 - val_loss: 0.6276 - val_accuracy: 0.6797
Epoch 24/100
9/9 - 91s - loss: 0.6069 - accuracy: 0.7361 - val_loss: 0.6506 - val_accuracy: 0.7344
Epoch 25/100
9/9 - 79s - loss: 0.6074 - accuracy: 0.7049 - val_loss: 0.5516 - val_accuracy: 0.7656
Epoch 26/100
9/9 - 80s - loss: 0.5855 - accuracy: 0.7292 - val_loss: 0.5834 - val_accuracy: 0.7266
Epoch 27/100
9/9 - 80s - loss: 0.5811 - accuracy: 0.7326 - val_loss: 0.6480 - val_accuracy: 0.6797
Epoch 28/100
9/9 - 72s - loss: 0.6588 - accuracy: 0.6667 - val_loss: 0.5577 - val_accuracy: 0.7578
Epoch 29/100
9/9 - 70s - loss: 0.5841 - accuracy: 0.7326 - val_loss: 0.5430 - val_accuracy: 0.7734
Epoch 30/100
9/9 - 75s - loss: 0.5983 - accuracy: 0.7153 - val_loss: 0.5435 - val_accuracy: 0.7734
Epoch 31/100
9/9 - 72s - loss: 0.5929 - accuracy: 0.7153 - val_loss: 0.5788 - val_accuracy: 0.7344
Epoch 32/100
9/9 - 79s - loss: 0.5638 - accuracy: 0.7500 - val_loss: 0.6344 - val_accuracy: 0.6719
Epoch 33/100
9/9 - 77s - loss: 0.5751 - accuracy: 0.7604 - val_loss: 0.6815 - val_accuracy: 0.7422
Epoch 34/100
9/9 - 54s - loss: 0.5690 - accuracy: 0.7847 - val_loss: 0.6685 - val_accuracy: 0.6719
Epoch 35/100
9/9 - 58s - loss: 0.5628 - accuracy: 0.7535 - val_loss: 0.5768 - val_accuracy: 0.7500
Epoch 36/100
9/9 - 57s - loss: 0.5919 - accuracy: 0.7222 - val_loss: 0.5623 - val_accuracy: 0.7500
Epoch 37/100
9/9 - 61s - loss: 0.6260 - accuracy: 0.7465 - val_loss: 0.5180 - val_accuracy: 0.7891
Epoch 38/100
9/9 - 53s - loss: 0.5824 - accuracy: 0.7326 - val_loss: 0.5426 - val_accuracy: 0.7656
Epoch 39/100
9/9 - 60s - loss: 0.6080 - accuracy: 0.7083 - val_loss: 0.5955 - val_accuracy: 0.7188
Epoch 40/100
9/9 - 52s - loss: 0.5519 - accuracy: 0.7604 - val_loss: 0.5871 - val_accuracy: 0.7266
Epoch 41/100
9/9 - 46s - loss: 0.5804 - accuracy: 0.7361 - val_loss: 0.5879 - val_accuracy: 0.7266
Epoch 42/100
9/9 - 49s - loss: 0.5450 - accuracy: 0.7674 - val_loss: 0.6834 - val_accuracy: 0.7266
Epoch 43/100
```

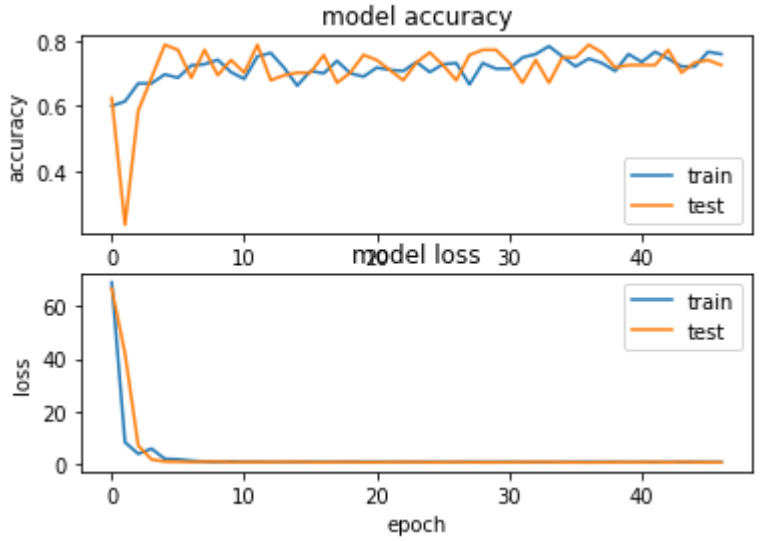
```
9/9 - 47s - loss: 0.5663 - accuracy: 0.7465 - val_loss: 0.5348 - val_accuracy: 0.7734
Epoch 44/100
9/9 - 42s - loss: 0.5912 - accuracy: 0.7222 - val_loss: 0.7126 - val_accuracy: 0.7031
Epoch 45/100
9/9 - 48s - loss: 0.5919 - accuracy: 0.7222 - val_loss: 0.5789 - val_accuracy: 0.7344
Epoch 46/100
9/9 - 47s - loss: 0.5429 - accuracy: 0.7674 - val_loss: 0.5714 - val_accuracy: 0.7422
Epoch 47/100
9/9 - 41s - loss: 0.5504 - accuracy: 0.7604 - val_loss: 0.5874 - val_accuracy: 0.7266
```

In [ ]:

```
plt.subplot(2,1,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')

plt.subplot(2,1,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')

plt.show()
```



Reference model - VGG19

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img, img_to_array
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D,GlobalAveragePooling2D, Conv3D, AveragePooling2D
from tensorflow.keras.layers import Activation, Dropout, BatchNormalization, Flatten, Dense, AvgPool2D,MaxPool2D
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.optimizers import Adam, SGD
import tensorflow as tf
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.vgg19 import preprocess_input
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [ ]:
```
# path='/content/gdrive/MyDrive/COVID-19_Radiography_Dataset'
# savepath = '/content/gdrive/MyDrive/COVID-19_Radiography_Dataset_pp'
```

In [ ]:
```
# from skimage.filters.rank import equalize
# from skimage.morphology import disk
# import os
# import skimage.io as io
# import skimage.color as color
# import skimage.transform as trf
# import numpy as np
# for directory in sorted(os.listdir(path)):
#         data_path = path +"/"+ directory
#         save_path = savepath +"/"+ directory
#         for im in os.listdir(data_path)[:]:
#             image = io.imread(f"{data_path}/{im}")
#             #image - np.array(image)
#             image = equalize(image, disk(70))
#             #image = color.rgb2gray(image)
#             #image = trf.resize(image, (image_sz, image_sz))
#             #images.append(image)
```

```
#             #labels.append(directory)
#             io.imsave(f"{save_path}/{im}", image)
```

In [ ]:
```
image_prepro_path = '/content/gdrive/MyDrive/COVID-19_Radiography_Dataset_pp'
classes=["COVID",  "Normal"]
num_classes = len(classes)
batch_size=32
target_size=(224, 224)
# Splitting the dataset into 70:30 by ImageGenerator
data_gen = ImageDataGenerator(preprocessing_function=preprocess_input,rotation_range=15, zoom_range=0.2, width_shift_rang

# Loading images to train generator
train_set = data_gen.flow_from_directory(directory=image_prepro_path,
                                         target_size=target_size,
                                         class_mode='categorical',
                                         subset='training',
                                         shuffle=True, classes=classes,
                                         batch_size=batch_size,
                                         color_mode="rgb")
# Loading images to test generator
test_set = data_gen.flow_from_directory(directory=image_prepro_path,
                                         target_size=target_size,
                                         class_mode='categorical',
                                         subset='validation',
                                         shuffle=True, classes=classes,
                                         batch_size=batch_size,
                                         color_mode="rgb")
```

```
Found 9674 images belonging to 2 classes.
Found 4144 images belonging to 2 classes.
```

In [ ]:
```
#checkpoint
import tensorflow
from tensorflow.keras.callbacks import ModelCheckpoint
early_stop = tensorflow.keras.callbacks.EarlyStopping(monitor='val_loss', patience= 10)
callbacks_list= ModelCheckpoint('/content/gdrive/MyDrive/ANLY535_Models/Weights-{epoch:03d}--{val_loss:.5f}.hdf5', monitc
callbacks = [early_stop, callbacks_list]
```

In [ ]:
```
from tensorflow.keras.models import Model
import tensorflow.keras as keras

vgg = VGG19(include_top=False, weights='imagenet', input_shape=(224,224,3),pooling='max')
output = vgg.layers[-1].output
output = tensorflow.keras.layers.Flatten()(output)
vgg = Model(vgg.input, output)
```

```
for layer in vgg.layers:
    layer.trainable = False

model = Sequential()
model.add(vgg)
model.add(Dense(128, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.summary()

opt=Adam(learning_rate=0.008)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=["accuracy"])
#model.evaluate_generator(test_set)
# # fitting the model
history1 = model.fit_generator(train_set, steps_per_epoch=len(train_set) // batch_size, validation_steps=len(test_set) //
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf
_kernels_notop.h5
80142336/80134624 [==============================] - 2s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model (Functional)           (None, 512)               20024384
_________________________________________________________________
dense (Dense)                (None, 128)               65664
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 20,090,306
Trainable params: 65,922
Non-trainable params: 20,024,384
_________________________________________________________________
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1940: UserWarning: `Model.fit_generator
` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/100
9/9 - 376s - loss: 1.8034 - accuracy: 0.4931 - val_loss: 0.5413 - val_accuracy: 0.7812
Epoch 2/100
9/9 - 342s - loss: 0.6463 - accuracy: 0.7396 - val_loss: 0.6111 - val_accuracy: 0.7344
Epoch 3/100
9/9 - 337s - loss: 0.6089 - accuracy: 0.6910 - val_loss: 0.4376 - val_accuracy: 0.7891
Epoch 4/100
9/9 - 328s - loss: 0.5388 - accuracy: 0.7674 - val_loss: 0.5233 - val_accuracy: 0.6953
Epoch 5/100
9/9 - 309s - loss: 0.5045 - accuracy: 0.7500 - val_loss: 0.5024 - val_accuracy: 0.7969
Epoch 6/100
9/9 - 310s - loss: 0.4591 - accuracy: 0.7917 - val_loss: 0.3737 - val_accuracy: 0.7969
```

```
Epoch 7/100
9/9 - 292s - loss: 0.4851 - accuracy: 0.7743 - val_loss: 0.5100 - val_accuracy: 0.7344
Epoch 8/100
9/9 - 272s - loss: 0.5211 - accuracy: 0.8264 - val_loss: 0.4539 - val_accuracy: 0.7734
Epoch 9/100
9/9 - 258s - loss: 0.5026 - accuracy: 0.7569 - val_loss: 0.5146 - val_accuracy: 0.6953
Epoch 10/100
9/9 - 263s - loss: 0.4186 - accuracy: 0.8194 - val_loss: 0.3779 - val_accuracy: 0.8125
Epoch 11/100
9/9 - 238s - loss: 0.4285 - accuracy: 0.8299 - val_loss: 0.3711 - val_accuracy: 0.8281
Epoch 12/100
9/9 - 244s - loss: 0.4712 - accuracy: 0.7778 - val_loss: 0.4357 - val_accuracy: 0.7812
Epoch 13/100
9/9 - 229s - loss: 0.5227 - accuracy: 0.7708 - val_loss: 0.3338 - val_accuracy: 0.8281
Epoch 14/100
9/9 - 237s - loss: 0.4213 - accuracy: 0.7986 - val_loss: 0.3370 - val_accuracy: 0.8750
Epoch 15/100
9/9 - 210s - loss: 0.3910 - accuracy: 0.8333 - val_loss: 0.3625 - val_accuracy: 0.8359
Epoch 16/100
9/9 - 201s - loss: 0.4021 - accuracy: 0.8160 - val_loss: 0.3064 - val_accuracy: 0.8906
Epoch 17/100
9/9 - 201s - loss: 0.3653 - accuracy: 0.8611 - val_loss: 0.3239 - val_accuracy: 0.8516
Epoch 18/100
9/9 - 200s - loss: 0.3253 - accuracy: 0.8819 - val_loss: 0.3762 - val_accuracy: 0.8359
Epoch 19/100
9/9 - 180s - loss: 0.3577 - accuracy: 0.8750 - val_loss: 0.2538 - val_accuracy: 0.8984
Epoch 20/100
9/9 - 190s - loss: 0.3747 - accuracy: 0.8576 - val_loss: 0.3028 - val_accuracy: 0.8984
Epoch 21/100
9/9 - 175s - loss: 0.3354 - accuracy: 0.8958 - val_loss: 0.2798 - val_accuracy: 0.8984
Epoch 22/100
9/9 - 178s - loss: 0.2731 - accuracy: 0.8993 - val_loss: 0.2843 - val_accuracy: 0.8516
Epoch 23/100
9/9 - 183s - loss: 0.3195 - accuracy: 0.8819 - val_loss: 0.3826 - val_accuracy: 0.8750
Epoch 24/100
9/9 - 157s - loss: 0.4259 - accuracy: 0.8264 - val_loss: 0.2658 - val_accuracy: 0.8984
Epoch 25/100
9/9 - 160s - loss: 0.3766 - accuracy: 0.8056 - val_loss: 0.3660 - val_accuracy: 0.8359
Epoch 26/100
9/9 - 153s - loss: 0.3309 - accuracy: 0.8611 - val_loss: 0.4116 - val_accuracy: 0.7969
Epoch 27/100
9/9 - 142s - loss: 0.3330 - accuracy: 0.8576 - val_loss: 0.2903 - val_accuracy: 0.9062
Epoch 28/100
9/9 - 136s - loss: 0.3944 - accuracy: 0.8647 - val_loss: 0.2355 - val_accuracy: 0.8984
Epoch 29/100
9/9 - 131s - loss: 0.3224 - accuracy: 0.8889 - val_loss: 0.2830 - val_accuracy: 0.8594
Epoch 30/100
9/9 - 137s - loss: 0.3216 - accuracy: 0.8576 - val_loss: 0.3658 - val_accuracy: 0.8594
Epoch 31/100
```

```
9/9 - 119s - loss: 0.4742 - accuracy: 0.8125 - val_loss: 0.2673 - val_accuracy: 0.8828
Epoch 32/100
9/9 - 117s - loss: 0.3636 - accuracy: 0.8576 - val_loss: 0.3485 - val_accuracy: 0.8750
Epoch 33/100
9/9 - 125s - loss: 0.3489 - accuracy: 0.8472 - val_loss: 0.4496 - val_accuracy: 0.8125
Epoch 34/100
9/9 - 110s - loss: 0.3378 - accuracy: 0.8472 - val_loss: 0.2952 - val_accuracy: 0.9141
Epoch 35/100
9/9 - 116s - loss: 0.3548 - accuracy: 0.8368 - val_loss: 0.3977 - val_accuracy: 0.8828
Epoch 36/100
9/9 - 119s - loss: 0.3742 - accuracy: 0.8368 - val_loss: 0.3220 - val_accuracy: 0.8828
Epoch 37/100
9/9 - 108s - loss: 0.3331 - accuracy: 0.8542 - val_loss: 0.2928 - val_accuracy: 0.8594
Epoch 38/100
9/9 - 101s - loss: 0.4664 - accuracy: 0.7744 - val_loss: 0.3407 - val_accuracy: 0.8750
```

In [ ]:

```
def covid_model():

    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(299, 299, 1)))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(BatchNormalization(momentum=0.9))
    model.add(AveragePooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(BatchNormalization(momentum=0.9))
    model.add(AveragePooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(BatchNormalization(momentum=0.9))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
   # model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))
    # compile model
    #optimizer_sgd = SGD(lr=0.01, momentum=0.9)
    opt=Adam(learning_rate=0.008)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=["accuracy"])
    model.summary()
    return model
```

In [ ]:

```
# covidmodel = covid_model()
# # fitting the model
# history = covidmodel.fit_generator(train_set, steps_per_epoch=len(train_set) // batch_size, validation_steps=len(test_s
```

In [ ]:
```
plt.subplot(2,1,1)
plt.plot(history1.history['accuracy'])
plt.plot(history1.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')

plt.subplot(2,1,2)
plt.plot(history1.history['loss'])
plt.plot(history1.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')

plt.show()
```