COVID-19 Detection from Lung X-rays

Import Libraries

In [1]:
```
from keras import backend as K
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.models import Sequential, Model
from keras.applications.vgg16 import VGG16, preprocess_input
from keras.optimizers import Adam, SGD, RMSprop

import tensorflow as tf

import os
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
```

```
Using TensorFlow backend.
```

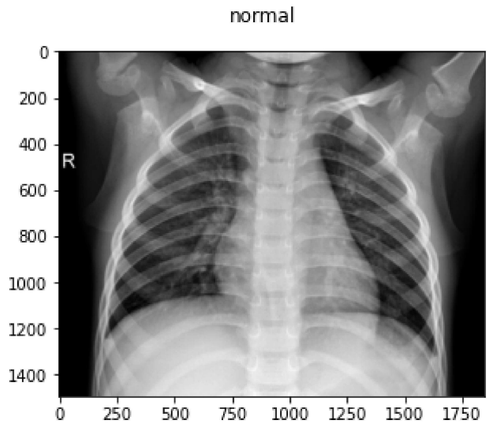
In [2]:
```
DATASET_DIR = "../input/covid-19-x-ray-10000-images/dataset"
```

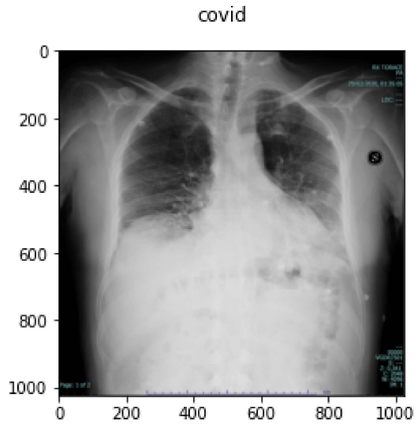
In [3]:
```
os.listdir(DATASET_DIR)
```

Out[3]:
```
['normal', 'covid']
```

In [4]:
```
import glob
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline

normal_images = []
for img_path in glob.glob(DATASET_DIR + '/normal/*'):
    normal_images.append(mpimg.imread(img_path))

fig = plt.figure()
fig.suptitle('normal')
plt.imshow(normal_images[0], cmap='gray')

covid_images = []
for img_path in glob.glob(DATASET_DIR + '/covid/*'):
    covid_images.append(mpimg.imread(img_path))

fig = plt.figure()
fig.suptitle('covid')
plt.imshow(covid_images[0], cmap='gray')
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x7c827f1fd0f0>
```





In [5]:
```
print(len(normal_images))
print(len(covid_images))
```

```
28
70
```

In [6]:
```
IMG_W = 150
IMG_H = 150
CHANNELS = 3

INPUT_SHAPE = (IMG_W, IMG_H, CHANNELS)
NB_CLASSES = 2
EPOCHS = 48
BATCH_SIZE = 6
```

In [7]:
```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=INPUT_SHAPE))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(64,(3,3)))
model.add(Activation("relu"))
model.add(Conv2D(250,(3,3)))
model.add(Activation("relu"))

model.add(Conv2D(128,(3,3)))
model.add(Activation("relu"))
model.add(AvgPool2D(2,2))
model.add(Conv2D(64,(3,3)))
model.add(Activation("relu"))
model.add(AvgPool2D(2,2))

model.add(Conv2D(256,(2,2)))
model.add(Activation("relu"))
model.add(MaxPool2D(2,2))

model.add(Flatten())
model.add(Dense(32))
model.add(Dropout(0.25))
model.add(Dense(1))
model.add(Activation("sigmoid"))
```

In [8]:
```
model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [9]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 148, 148, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 72, 72, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 34, 34, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 250)       144250
_________________________________________________________________
activation_4 (Activation)    (None, 32, 32, 250)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 30, 30, 128)       288128
_________________________________________________________________
activation_5 (Activation)    (None, 30, 30, 128)       0
_________________________________________________________________
average_pooling2d_1 (Average (None, 15, 15, 128)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 13, 13, 64)        73792
_________________________________________________________________
activation_6 (Activation)    (None, 13, 13, 64)        0
_________________________________________________________________
average_pooling2d_2 (Average (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 5, 5, 256)         65792
_________________________________________________________________
activation_7 (Activation)    (None, 5, 5, 256)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                32800
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
_________________________________________________________________
activation_8 (Activation)    (None, 1)                 0
=================================================================
Total params: 633,435
Trainable params: 633,435
Non-trainable params: 0
_________________________________________________________________
```

In [10]:

```
train_datagen = ImageDataGenerator(rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
```

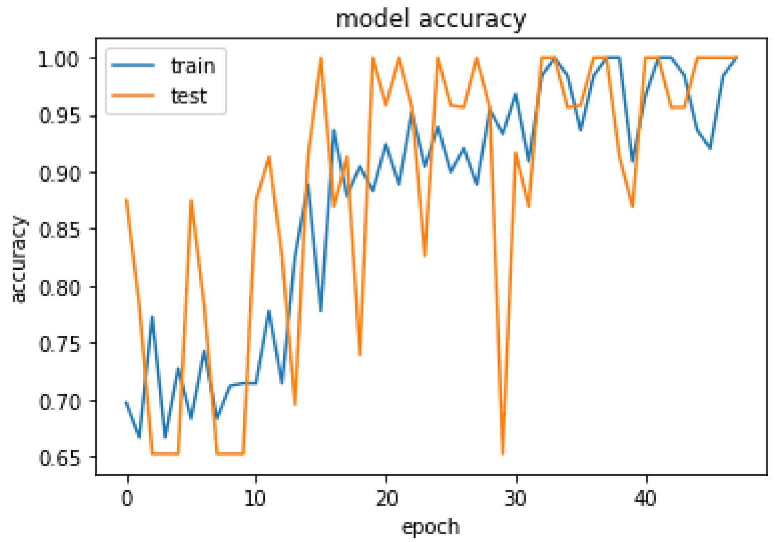
```
    horizontal_flip=True,
    validation_split=0.3)

train_generator = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=(IMG_H, IMG_W),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training')

validation_generator = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=(IMG_H, IMG_W),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle= False,
    subset='validation')

history = model.fit_generator(
    train_generator,
    steps_per_epoch = train_generator.samples // BATCH_SIZE,
    validation_data = validation_generator,
    validation_steps = validation_generator.samples // BATCH_SIZE,
    epochs = EPOCHS)
```

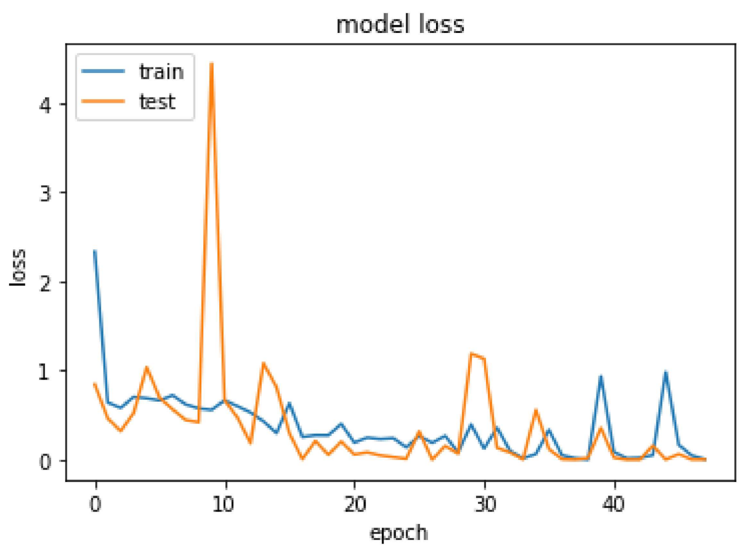
```
Found 69 images belonging to 2 classes.
Found 29 images belonging to 2 classes.
Epoch 1/48
11/11 [==============================] - 4s 341ms/step - loss: 2.3287 - accuracy: 0.6970 - val_loss: 0.8433 - val_accuracy: 0.8750
Epoch 2/48
11/11 [==============================] - 3s 240ms/step - loss: 0.6496 - accuracy: 0.6667 - val_loss: 0.4624 - val_accuracy: 0.7826
Epoch 3/48
11/11 [==============================] - 3s 229ms/step - loss: 0.5798 - accuracy: 0.7727 - val_loss: 0.3195 - val_accuracy: 0.6522
Epoch 4/48
11/11 [==============================] - 2s 164ms/step - loss: 0.7043 - accuracy: 0.6667 - val_loss: 0.5229 - val_accuracy: 0.6522
Epoch 5/48
11/11 [==============================] - 2s 154ms/step - loss: 0.6883 - accuracy: 0.7273 - val_loss: 1.0366 - val_accuracy: 0.6522
Epoch 6/48
11/11 [==============================] - 2s 175ms/step - loss: 0.6698 - accuracy: 0.6833 - val_loss: 0.6933 - val_accuracy: 0.8750
Epoch 7/48
11/11 [==============================] - 3s 245ms/step - loss: 0.7216 - accuracy: 0.7424 - val_loss: 0.5640 - val_accuracy: 0.7826
Epoch 8/48
11/11 [==============================] - 2s 192ms/step - loss: 0.6202 - accuracy: 0.6833 - val_loss: 0.4449 - val_accuracy: 0.6522
Epoch 9/48
11/11 [==============================] - 2s 143ms/step - loss: 0.5749 - accuracy: 0.7121 - val_loss: 0.4195 - val_accuracy: 0.6522
Epoch 10/48
11/11 [==============================] - 2s 169ms/step - loss: 0.5585 - accuracy: 0.7143 - val_loss: 4.4365 - val_accuracy: 0.6522
Epoch 11/48
11/11 [==============================] - 2s 181ms/step - loss: 0.6922 - accuracy: 0.7143 - val_loss: 0.6672 - val_accuracy: 0.8750
Epoch 12/48
11/11 [==============================] - 3s 242ms/step - loss: 0.6844 - accuracy: 0.7778 - val_loss: 0.4699 - val_accuracy: 0.9130
Epoch 13/48
11/11 [==============================] - 2s 164ms/step - loss: 0.5349 - accuracy: 0.7143 - val_loss: 0.1836 - val_accuracy: 0.8261
Epoch 14/48
11/11 [==============================] - 2s 178ms/step - loss: 0.4271 - accuracy: 0.8254 - val_loss: 1.0802 - val_accuracy: 0.6957
Epoch 15/48
11/11 [==============================] - 2s 173ms/step - loss: 0.2864 - accuracy: 0.8889 - val_loss: 0.8135 - val_accuracy: 0.9130
Epoch 16/48
11/11 [==============================] - 2s 169ms/step - loss: 0.6410 - accuracy: 0.7778 - val_loss: 0.2952 - val_accuracy: 1.0000
Epoch 17/48
11/11 [==============================] - 3s 252ms/step - loss: 0.2512 - accuracy: 0.9365 - val_loss: 0.0054 - val_accuracy: 0.8696
Epoch 18/48
11/11 [==============================] - 2s 156ms/step - loss: 0.2743 - accuracy: 0.8788 - val_loss: 0.2087 - val_accuracy: 0.9130
Epoch 19/48
11/11 [==============================] - 2s 164ms/step - loss: 0.2644 - accuracy: 0.9048 - val_loss: 0.0543 - val_accuracy: 0.7391
Epoch 20/48
11/11 [==============================] - 2s 172ms/step - loss: 0.3993 - accuracy: 0.8833 - val_loss: 0.2020 - val_accuracy: 1.0000
Epoch 21/48
11/11 [==============================] - 2s 171ms/step - loss: 0.1909 - accuracy: 0.9242 - val_loss: 0.0572 - val_accuracy: 0.9583
Epoch 22/48
11/11 [==============================] - 3s 264ms/step - loss: 0.2396 - accuracy: 0.8889 - val_loss: 0.0822 - val_accuracy: 1.0000
Epoch 23/48
11/11 [==============================] - 2s 153ms/step - loss: 0.2237 - accuracy: 0.9524 - val_loss: 0.0473 - val_accuracy: 0.9565
Epoch 24/48
11/11 [==============================] - 2s 148ms/step - loss: 0.2514 - accuracy: 0.9048 - val_loss: 0.0281 - val_accuracy: 0.8261
Epoch 25/48
11/11 [==============================] - 2s 164ms/step - loss: 0.1379 - accuracy: 0.9394 - val_loss: 0.0106 - val_accuracy: 1.0000
Epoch 26/48
11/11 [==============================] - 2s 169ms/step - loss: 0.2424 - accuracy: 0.9000 - val_loss: 0.3152 - val_accuracy: 0.9583
Epoch 27/48
11/11 [==============================] - 3s 257ms/step - loss: 0.1805 - accuracy: 0.9206 - val_loss: 1.9593e-05 - val_accuracy: 0.9565
Epoch 28/48
11/11 [==============================] - 2s 195ms/step - loss: 0.2558 - accuracy: 0.8889 - val_loss: 0.1535 - val_accuracy: 1.0000
Epoch 29/48
11/11 [==============================] - 2s 160ms/step - loss: 0.0820 - accuracy: 0.9545 - val_loss: 0.0664 - val_accuracy: 0.9565
```

```
Epoch 30/48
11/11 [==============================] - 1s 122ms/step - loss: 0.3616 - accuracy: 0.9333 - val_loss: 1.1867 - val_accuracy: 0.6522
Epoch 31/48
11/11 [==============================] - 2s 191ms/step - loss: 0.1188 - accuracy: 0.9683 - val_loss: 1.1302 - val_accuracy: 0.9167
Epoch 32/48
11/11 [==============================] - 3s 277ms/step - loss: 0.3581 - accuracy: 0.9091 - val_loss: 0.1322 - val_accuracy: 0.8696
Epoch 33/48
11/11 [==============================] - 2s 155ms/step - loss: 0.0989 - accuracy: 0.9841 - val_loss: 0.0804 - val_accuracy: 1.0000
Epoch 34/48
11/11 [==============================] - 2s 140ms/step - loss: 0.0138 - accuracy: 1.0000 - val_loss: 0.0031 - val_accuracy: 1.0000
Epoch 35/48
11/11 [==============================] - 2s 162ms/step - loss: 0.0610 - accuracy: 0.9841 - val_loss: 0.5573 - val_accuracy: 0.9565
Epoch 36/48
11/11 [==============================] - 2s 174ms/step - loss: 0.3330 - accuracy: 0.9365 - val_loss: 0.1187 - val_accuracy: 0.9583
Epoch 37/48
11/11 [==============================] - 3s 231ms/step - loss: 0.0474 - accuracy: 0.9841 - val_loss: 0.0056 - val_accuracy: 1.0000
Epoch 38/48
11/11 [==============================] - 3s 230ms/step - loss: 0.0156 - accuracy: 1.0000 - val_loss: 6.3188e-05 - val_accuracy: 1.0000
Epoch 39/48
11/11 [==============================] - 2s 172ms/step - loss: 6.9047e-04 - accuracy: 1.0000 - val_loss: 0.0187 - val_accuracy: 0.9130
Epoch 40/48
11/11 [==============================] - 2s 147ms/step - loss: 0.9334 - accuracy: 0.9091 - val_loss: 0.3541 - val_accuracy: 0.8696
Epoch 41/48
11/11 [==============================] - 2s 191ms/step - loss: 0.0794 - accuracy: 0.9667 - val_loss: 0.0229 - val_accuracy: 1.0000
Epoch 42/48
11/11 [==============================] - 3s 247ms/step - loss: 0.0144 - accuracy: 1.0000 - val_loss: 3.1693e-04 - val_accuracy: 1.0000
Epoch 43/48
11/11 [==============================] - 2s 154ms/step - loss: 0.0187 - accuracy: 1.0000 - val_loss: 3.3372e-04 - val_accuracy: 0.9565
Epoch 44/48
11/11 [==============================] - 2s 162ms/step - loss: 0.0491 - accuracy: 0.9848 - val_loss: 0.1558 - val_accuracy: 0.9565
Epoch 45/48
11/11 [==============================] - 2s 161ms/step - loss: 0.9525 - accuracy: 0.9365 - val_loss: 0.0018 - val_accuracy: 1.0000
Epoch 46/48
11/11 [==============================] - 2s 169ms/step - loss: 0.1586 - accuracy: 0.9206 - val_loss: 0.0619 - val_accuracy: 1.0000
Epoch 47/48
11/11 [==============================] - 3s 253ms/step - loss: 0.0445 - accuracy: 0.9841 - val_loss: 0.0016 - val_accuracy: 1.0000
Epoch 48/48
11/11 [==============================] - 2s 163ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.0019 - val_accuracy: 1.0000
```

In [11]:

```
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='upper left')
plt.show()

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 left')
plt.show()
```





In [12]:
```
print("training_accuracy", history.history['accuracy'][-1])
print("validation_accuracy", history.history['val_accuracy'][-1])
```

```
training_accuracy 1.0
validation_accuracy 1.0
```

In [13]:
```
label = validation_generator.classes
```

In [14]:
```
pred= model.predict(validation_generator)
predicted_class_indices=np.argmax(pred,axis=1)
labels = (validation_generator.class_indices)
labels2 = dict((v,k) for k,v in labels.items())
predictions = [labels2[k] for k in predicted_class_indices]
print(predicted_class_indices)
print (labels)
print (predictions)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
{'covid': 0, 'normal': 1}
['covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid',
'covid', 'covid', 'covid', 'covid', 'covid', 'covid', 'covid']
```

In [15]:
```
from sklearn.metrics import confusion_matrix
```

```
cf = confusion_matrix(predicted_class_indices,label)
cf
```

Out[15]:
```
array([[21,  8],
       [ 0,  0]])
```

In [16]:
```
exp_series = pd.Series(label)
pred_series = pd.Series(predicted_class_indices)
pd.crosstab(exp_series, pred_series, rownames=['Actual'], colnames=['Predicted'],margins=True)
```

Out[16]:

Actual \ Predicted	0	All
0	21	21
1	8	8
All	29	29

In [17]:
```
plt.matshow(cf)
plt.title('Confusion Matrix Plot')
plt.colorbar()
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show();
```