In [1]:

```
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation,Dropout,Flatten,Dense
from keras import backend as k
import numpy as np
```

```
Using TensorFlow backend.
```

In [2]:

```
img_width,img_height=150,150
```

In [52]:

```
train_data_dir="Dataset/Train"
validation_data_dir="dataset/Val"
nb_train_samples=200
nb_validation_samples=50
epochs=10
batch_size=20
```

In [53]:

```
if  k.image_data_format() == 'channels_first':
    input_shape = (3,img_width, img_height)
else:
    input_shape = (img_width, img_height, 3)
```

In [54]:

```
train_datagen = ImageDataGenerator(   # data augmentation
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size, class_mode ="binary")

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,class_mode ="binary")
```

```
Found 288 images belonging to 2 classes.
Found 60 images belonging to 2 classes.
```

In [55]:

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

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

```
model.add(Conv2D(64, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 149, 149, 32)      416
_________________________________________________________________
activation_6 (Activation)    (None, 149, 149, 32)      0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 73, 73, 32)        4128
_________________________________________________________________
activation_7 (Activation)    (None, 73, 73, 32)        0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 35, 35, 64)        8256
_________________________________________________________________
activation_8 (Activation)    (None, 35, 35, 64)        0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                1183808
_________________________________________________________________
activation_9 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_10 (Activation)   (None, 1)                 0
=================================================================
Total params: 1,196,673
Trainable params: 1,196,673
Non-trainable params: 0
_________________________________________________________________
```

In [56]:

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

In [57]:

```
model.fit_generator(train_generator,
    steps_per_epoch = nb_train_samples // batch_size,
    epochs = epochs,
    validation_data = validation_generator)
```

```
Epoch 1/10
10/10 [==============================] - 36s 4s/step - loss: 0.6738 - accuracy: 0.6117 - val_loss:
0.4696 - val_accuracy: 0.9333
Epoch 2/10
10/10 [==============================] - 10s 1s/step - loss: 0.3905 - accuracy: 0.8404 - val_loss:
0.0773 - val_accuracy: 0.9500
Epoch 3/10
10/10 [==============================] - 13s 1s/step - loss: 0.3353 - accuracy: 0.8800 - val_loss:
0.1846 - val_accuracy: 0.9667
```

```
0.1846 - val_accuracy: 0.9667
Epoch 4/10
10/10 [==============================] - 12s 1s/step - loss: 0.2696 - accuracy: 0.8989 - val_loss:
0.1811 - val_accuracy: 0.9167
Epoch 5/10
10/10 [==============================] - 12s 1s/step - loss: 0.1724 - accuracy: 0.9500 - val_loss:
0.2651 - val_accuracy: 0.9667
Epoch 6/10
10/10 [==============================] - 12s 1s/step - loss: 0.1877 - accuracy: 0.9415 - val_loss:
0.0657 - val_accuracy: 0.9833
Epoch 7/10
10/10 [==============================] - 13s 1s/step - loss: 0.4253 - accuracy: 0.8600 - val_loss:
0.1131 - val_accuracy: 0.9500
Epoch 8/10
10/10 [==============================] - 12s 1s/step - loss: 0.1282 - accuracy: 0.9521 - val_loss:
0.1116 - val_accuracy: 0.9833
Epoch 9/10
10/10 [==============================] - 11s 1s/step - loss: 0.1480 - accuracy: 0.9521 - val_loss:
0.0593 - val_accuracy: 0.9667
Epoch 10/10
10/10 [==============================] - 12s 1s/step - loss: 0.1257 - accuracy: 0.9681 - val_loss:
0.0055 - val_accuracy: 0.9833
```

Out[57]:

```
<keras.callbacks.callbacks.History at 0xc922deff48>
```

In [58]:

```
model.save_weights("first_model.h5")
```

In [59]:

```
pred=image.load_img("Dataset/Val/Normal/NORMAL2-IM-0462-0001.jpeg",target_size=(150,150))
```

In [60]:

```
pred
```

Out[60]:

In [61]:

```
pred=image.img_to_array(pred)
```

In [62]:

```
pred=np.expand_dims(pred,axis=0)
```

In [63]:

```
result=model.predict(pred)
```

In [64]:

```
print(result)
```

```
[[1.]]
```

In [ ]:

```
```