Import Libraries

```
%matplotlib inline

import numpy as np
import pandas as pd

import keras

from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

from keras import models
from keras.models import Sequential
from keras.models import Model
from keras.models import load_model

from keras.layers import Input, Dropout, Flatten, Conv2D, MaxPooling2D, BatchNormalization, GlobalAveragePooling2D, Dense
from keras.callbacks import Callback, ModelCheckpoint

from keras.applications.vgg16 import preprocess_input



import matplotlib.pyplot as plt
import cv2

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

Training parameters

```
img_width, img_height = 150, 150
input_shape = (img_height, img_width, 3)
nb_train_samples = 4
nb_val_samples = 4
nb_epochs = 200
num_classes=2


train_data_dir = '/content/drive/MyDrive/train'
val_data_dir = '/content/drive/MyDrive/validation'

train_datagen = ImageDataGenerator(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=1, class_mode='categorical')
validation_generator = test_datagen.flow_from_directory(val_data_dir, target_size=(img_width, img_height),
                                                        batch_size=1,class_mode='categorical')
```

```
Found 4 images belonging to 2 classes.
Found 4 images belonging to 2 classes.
```

Build a classification model on top of Base Network

```
model = Sequential()
model.add(Conv2D(8, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
```

```
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 148, 148, 8)       224
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 74, 74, 8)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 72, 72, 16)        1168
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 36, 36, 16)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 36, 36, 16)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 20736)             0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                663584
_________________________________________________________________
batch_normalization_2 (Batch (None, 32)                128
_________________________________________________________________
dropout_6 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 66
=================================================================
Total params: 665,170
Trainable params: 665,106
Non-trainable params: 64
_________________________________________________________________
```

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

```
model_checkpoint_callback = keras.callbacks.ModelCheckpoint("best_Model.h5",save_best_only=True)
```

```
history = model.fit_generator( train_generator, callbacks = model_checkpoint_callback, steps_per_epoch=nb_train_samples,epoc
print('Training Completed!')
```

```
print('Training Completed!')
print(history.history.keys())
```

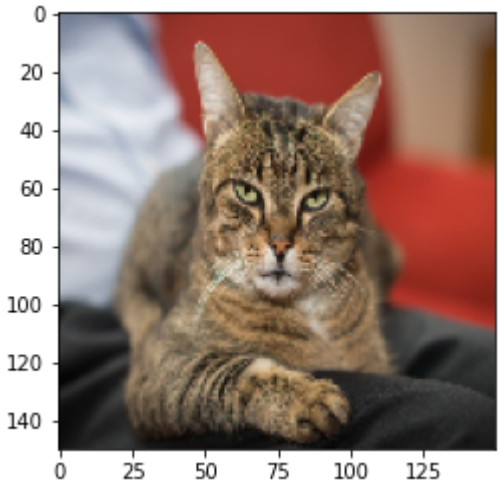
```
Training Completed!
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
img_path = '/content/drive/MyDrive/cat.jpg'
label = ['Cat','Dog']
img = image.load_img(img_path, target_size=(150, 150))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

features = model.predict(x)
thresholded = (features>0.5)*1
ind = np.argmax(thresholded)
print('Predicted Array:',thresholded)
print('Predicted Label:',label[ind])
```

```
Predicted Array: [[1 0]]
Predicted Label: Cat
```

```
imgplot = plt.imshow(img)
plt.show()
```



```
model = load_model('/content/best_Model.h5')
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 148, 148, 8)       224
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 74, 74, 8)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 72, 72, 16)        1168
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 36, 36, 16)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 36, 36, 16)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 20736)             0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                663584
_________________________________________________________________
batch_normalization_2 (Batch (None, 32)                128
_________________________________________________________________
dropout_6 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 66
=================================================================
Total params: 665,170
Trainable params: 665,106
Non-trainable params: 64
_________________________________________________________________
```

```
img_path = '/content/drive/MyDrive/cat.jpg'
label = ['Cat','Dog']
img = image.load_img(img_path, target_size=(150, 150))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

features = model.predict(x)
```

```
thresholded = (features>0.5)*1
ind = np.argmax(thresholded)
print('Predicted Array:',thresholded)
print('Predicted Label:',label[ind])
```

```
Predicted Array: [[1 0]]
Predicted Label: Cat
```

```
imgplot = plt.imshow(img)
plt.show()
```