VGG-16_Keras_HappyHouse

April 9, 2019

1 Keras - the Happy House

In [1]:
```
import numpy as np
from keras import layers
from keras.layers import Input, Dense, Activation, ZeroPadding2D, BatchNorm
from keras.layers import AveragePooling2D, MaxPooling2D, Dropout, GlobalMax
from keras.models import Model
from keras.preprocessing import image
from keras.utils import layer_utils
from keras.utils.data_utils import get_file
from keras.applications.imagenet_utils import preprocess_input
import pydot
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot
from keras.utils import plot_model
from kt_utils import *

import keras.backend as K
K.set_image_data_format('channels_last')
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow

%matplotlib inline
```

```
Using TensorFlow backend.
```

In [2]:
```
X_train_orig, Y_train_orig, X_test_orig, Y_test_orig, classes = load_datase

# Normalize image vectors
X_train = X_train_orig/255.
X_test = X_test_orig/255.

# Reshape
Y_train = Y_train_orig.T
Y_test = Y_test_orig.T
```

```
print ("number of training examples = " + str(X_train.shape[0]))
print ("number of test examples = " + str(X_test.shape[0]))
print ("X_train shape: " + str(X_train.shape))
print ("Y_train shape: " + str(Y_train.shape))
print ("X_test shape: " + str(X_test.shape))
print ("Y_test shape: " + str(Y_test.shape))
```

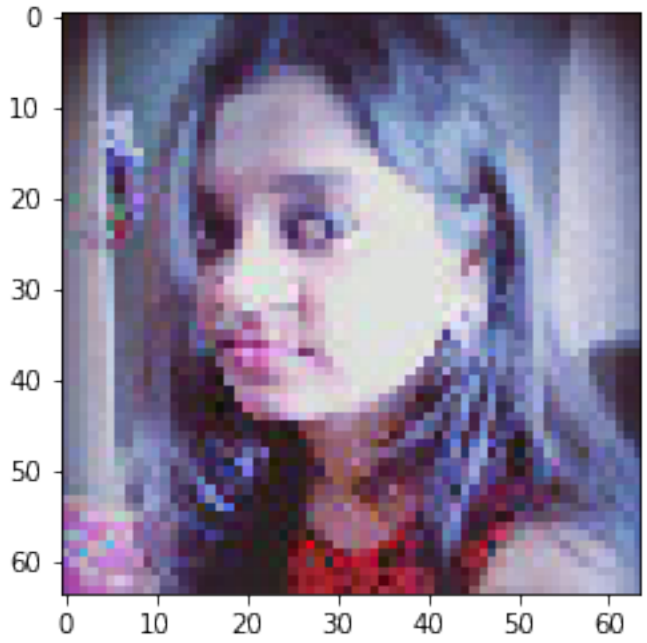
```
number of training examples = 600
number of test examples = 150
X_train shape: (600, 64, 64, 3)
Y_train shape: (600, 1)
X_test shape: (150, 64, 64, 3)
Y_test shape: (150, 1)
```

In [15]:
```
def HappyModel(input_shape):
    """
    Implementation of the HappyModel.

    Arguments:
    input_shape -- shape of the images of the dataset

    Returns:
    model -- a Model() instance in Keras
    """

    X_input = Input(input_shape)
    #X = ZeroPadding2D((3,3))(X_input)
    X = Conv2D(64, (3, 3), strides = (1, 1), padding='same', name = 'z1')
    X = BatchNormalization(axis = 3 , name = 'bn1')(X)
    X = Activation('relu')(X)
    X = Conv2D(64, (3, 3), strides = (1, 1), padding='same', name = 'z2')
    X = BatchNormalization(axis = 3 , name = 'bn2')(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((2, 2), name = 'max_pool1')(X)
    X = Conv2D(128, (3, 3), strides = (1, 1), padding='same', name = 'z3')
    X = BatchNormalization(axis = 3 , name = 'bn3')(X)
    X = Activation('relu')(X)
    X = Conv2D(128, (3, 3), strides = (1, 1), padding='same', name = 'z4')
    X = BatchNormalization(axis = 3 , name = 'bn4')(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((2, 2), name = 'max_pool2')(X)
    X = Conv2D(256, (3, 3), strides = (1, 1), padding='same', name = 'z5')
    X = BatchNormalization(axis = 3 , name = 'bn5')(X)
    X = Activation('relu')(X)
    X = Conv2D(256, (3, 3), strides = (1, 1), padding='same', name = 'z6')
    X = BatchNormalization(axis = 3 , name = 'bn6')(X)
    X = Activation('relu')(X)
```

```
            X = Conv2D(256, (3, 3), strides = (1, 1), padding='same', name = 'z7')
            X = BatchNormalization(axis = 3 , name = 'bn7')(X)
            X = Activation('relu')(X)
            X = MaxPooling2D((2, 2), name = 'max_pool3')(X)
            X = Conv2D(512, (3, 3), strides = (1, 1), padding='same', name = 'z8')
            X = BatchNormalization(axis = 3 , name = 'bn8')(X)
            X = Activation('relu')(X)
            X = Conv2D(512, (3, 3), strides = (1, 1), padding='same', name = 'z9')
            X = BatchNormalization(axis = 3 , name = 'bn9')(X)
            X = Activation('relu')(X)
            X = Conv2D(512, (3, 3), strides = (1, 1), padding='same', name = 'z10'
            X = BatchNormalization(axis = 3 , name = 'bn10')(X)
            X = Activation('relu')(X)
            X = MaxPooling2D((2, 2), name = 'max_pool4')(X)
            X = Conv2D(512, (3, 3), strides = (1, 1), padding='same', name = 'z11'
            X = BatchNormalization(axis = 3 , name = 'bn11')(X)
            X = Activation('relu')(X)
            X = Conv2D(512, (3, 3), strides = (1, 1), padding='same', name = 'z12'
            X = BatchNormalization(axis = 3 , name = 'bn12')(X)
            X = Activation('relu')(X)
            X = Conv2D(512, (3, 3), strides = (1, 1), padding='same', name = 'z13'
            X = BatchNormalization(axis = 3 , name = 'bn13')(X)
            X = Activation('relu')(X)
            X = MaxPooling2D((2, 2), name = 'max_pool5')(X)
            X = Flatten()(X)
            X = Dense(4096, activation='sigmoid', name = 'fc1')(X)
            X = Dense(4096, activation='sigmoid', name = 'fc2')(X)
            X = Dense(1, activation='sigmoid', name = 'fc3')(X)

            model = Model(inputs = X_input, outputs = X, name = 'HappyModel')


            return model
```

```
In [18]: happyModel = HappyModel(X_train.shape[1:])
         #print(X_train.shape[1:])

In [19]: happyModel.compile(optimizer = "adam", loss='binary_crossentropy', metrics

In [20]: happyModel.fit(x = X_train , y = Y_train, epochs = 2, batch_size = 16)

Epoch 1/2
600/600 [==============================] - 370s - loss: 7.9502 - acc: 0.4900
Epoch 2/2
600/600 [==============================] - 365s - loss: 8.0590 - acc: 0.5000


Out[20]: <keras.callbacks.History at 0x7fc13b9127b8>
```

```
In [21]: preds = happyModel.evaluate(x=X_test, y=Y_test)

         print()
         print ("Loss = " + str(preds[0]))
         print ("Test Accuracy = " + str(preds[1]))

150/150 [==============================] - 29s

Loss = 9.02613381704
Test Accuracy = 0.440000001589


In [22]: img_path = 'images/my_image.jpg'
         img = image.load_img(img_path, target_size=(64, 64))
         imshow(img)

         x = image.img_to_array(img)
         x = np.expand_dims(x, axis=0)
         x = preprocess_input(x)

         print(happyModel.predict(x))

[[  1.57023533e-22]]
```




```
In [23]: happyModel.summary()
```

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	(None, 64, 64, 3)	0
z1 (Conv2D)	(None, 64, 64, 64)	1792
bn1 (BatchNormalization)	(None, 64, 64, 64)	256
activation_42 (Activation)	(None, 64, 64, 64)	0
z2 (Conv2D)	(None, 64, 64, 64)	36928
bn2 (BatchNormalization)	(None, 64, 64, 64)	256
activation_43 (Activation)	(None, 64, 64, 64)	0
max_pool1 (MaxPooling2D)	(None, 32, 32, 64)	0
z3 (Conv2D)	(None, 32, 32, 128)	73856
bn3 (BatchNormalization)	(None, 32, 32, 128)	512
activation_44 (Activation)	(None, 32, 32, 128)	0
z4 (Conv2D)	(None, 32, 32, 128)	147584
bn4 (BatchNormalization)	(None, 32, 32, 128)	512
activation_45 (Activation)	(None, 32, 32, 128)	0
max_pool2 (MaxPooling2D)	(None, 16, 16, 128)	0
z5 (Conv2D)	(None, 16, 16, 256)	295168
bn5 (BatchNormalization)	(None, 16, 16, 256)	1024
activation_46 (Activation)	(None, 16, 16, 256)	0
z6 (Conv2D)	(None, 16, 16, 256)	590080
bn6 (BatchNormalization)	(None, 16, 16, 256)	1024
activation_47 (Activation)	(None, 16, 16, 256)	0
z7 (Conv2D)	(None, 16, 16, 256)	590080
bn7 (BatchNormalization)	(None, 16, 16, 256)	1024

activation_48 (Activation)	(None, 16, 16, 256)	0
max_pool3 (MaxPooling2D)	(None, 8, 8, 256)	0
z8 (Conv2D)	(None, 8, 8, 512)	1180160
bn8 (BatchNormalization)	(None, 8, 8, 512)	2048
activation_49 (Activation)	(None, 8, 8, 512)	0
z9 (Conv2D)	(None, 8, 8, 512)	2359808
bn9 (BatchNormalization)	(None, 8, 8, 512)	2048
activation_50 (Activation)	(None, 8, 8, 512)	0
z10 (Conv2D)	(None, 8, 8, 512)	2359808
bn10 (BatchNormalization)	(None, 8, 8, 512)	2048
activation_51 (Activation)	(None, 8, 8, 512)	0
max_pool4 (MaxPooling2D)	(None, 4, 4, 512)	0
z11 (Conv2D)	(None, 4, 4, 512)	2359808
bn11 (BatchNormalization)	(None, 4, 4, 512)	2048
activation_52 (Activation)	(None, 4, 4, 512)	0
z12 (Conv2D)	(None, 4, 4, 512)	2359808
bn12 (BatchNormalization)	(None, 4, 4, 512)	2048
activation_53 (Activation)	(None, 4, 4, 512)	0
z13 (Conv2D)	(None, 4, 4, 512)	2359808
bn13 (BatchNormalization)	(None, 4, 4, 512)	2048
activation_54 (Activation)	(None, 4, 4, 512)	0
max_pool5 (MaxPooling2D)	(None, 2, 2, 512)	0
flatten_5 (Flatten)	(None, 2048)	0
fc1 (Dense)	(None, 4096)	8392704

```
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
fc3 (Dense)                  (None, 1)                 4097
=================================================================
Total params: 39,909,697
Trainable params: 39,901,249
Non-trainable params: 8,448
_________________________________________________________________
```

In [24]:
```
plot_model(happyModel, to_file='HappyModel.png')
SVG(model_to_dot(happyModel).create(prog='dot', format='svg'))
```

Out[24]:


input_5: InputLayer
z1: Conv2D
bn1: BatchNormalization
activation_42: Activation
z2: Conv2D
bn2: BatchNormalization
activation_43: Activation
max_pool1: MaxPooling2D
z3: Conv2D
bn3: BatchNormalization
activation_44: Activation
z4: Conv2D
bn4: BatchNormalization
activation_45: Activation
max_pool2: MaxPooling2D
z5: Conv2D
bn5: BatchNormalization
activation_46: Activation
z6: Conv2D
bn6: BatchNormalization
activation_47: Activation
z7: Conv2D
bn7: BatchNormalization
activation_48: Activation
max_pool3: MaxPooling2D
z8: Conv2D
bn8: BatchNormalization
activation_49: Activation
z9: Conv2D
bn9: BatchNormalization
activation_50: Activation
z10: Conv2D
bn10: BatchNormalization
activation_51: Activation
max_pool4: MaxPooling2D
z11: Conv2D
bn11: BatchNormalization
activation_52: Activation
z12: Conv2D
bn12: BatchNormalization
activation_53: Activation
z13: Conv2D
bn13: BatchNormalization
activation_54: Activation
max_pool5: MaxPooling2D
flatten_5: Flatten
fc1: Dense
fc2: Dense
fc3: Dense

In [ ]: