Deblurring

June 14, 2018

```
In [ ]: %reset
```

** Importing Necessary Packages **

```
In [1]: import numpy as np
        from numpy import random
        import matplotlib.pyplot as plt
        import scipy.misc
        from skimage.io import imread
        from glob import glob
```

```
In [2]: from keras.layers import Conv2D, BatchNormalization, Activation,Merge, merge
        from keras.models import Model, Input
        from keras.optimizers import Adam
        import keras.backend as K
        import tensorflow as tf
```

```
/opt/conda/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the sec
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
/opt/conda/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: compiletime version 3.5 of
  return f(*args, **kwds)
```

```
In [3]: config = tf.ConfigProto()
        config.gpu_options.per_process_gpu_memory_fraction = 0.9
        K.set_session(tf.Session(config=config))
```

```
In [4]: from dataIO import pk
        import cv2 as cv
        import os
        import matplotlib.pyplot as plt
```

** Loading Images **

Only showing a small set of images from the local test set we generated.

```
In [ ]: '''
        clean_images_path = glob('./CelebA Images/Clean Images/*.png')
```

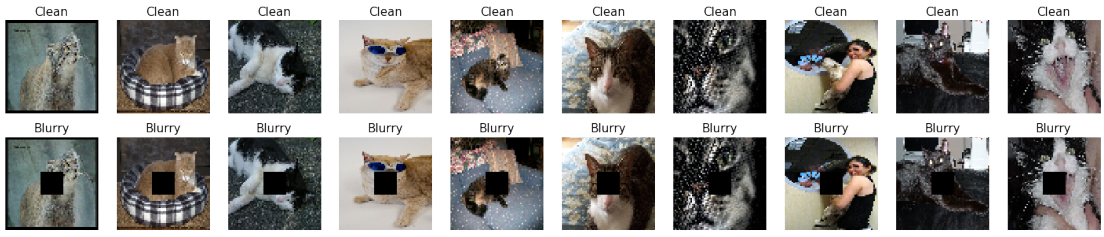
```
        blurry_images_path = glob('./CelebA Images/Blurry Images/*.png')
        Images = []; Blurry = []
        for image, blurry in zip(clean_images_path, blurry_images_path):
            Images.append(imread(image))
            Blurry.append(imread(blurry))
        Images = np.array(Images).astype('float32')
        Blurry = np.array(Blurry).astype('float32')
        '''
```

```
In [8]: X_train = pk.load("train.gz")
        X_validation = pk.load("validation.gz")
        X_test = pk.load("test.gz")
```

```
Load from 'train.gz' ...
    Complete! Elapse 1.201928 sec.

Load from 'validation.gz' ...
    Complete! Elapse 0.123054 sec.

Load from 'test.gz' ...
    Complete! Elapse 0.123529 sec.
```

```
In [9]: X_train['blur_data']=X_train['blur_data'].astype(float)
        X_train['data']=X_train['data'].astype(float)
        X_validation ['blur_data']=X_validation['blur_data'].astype(float)
        X_validation ['data']=X_validation['data'].astype(float)
        X_test ['blur_data']=X_test['blur_data'].astype(float)
        X_test ['data']=X_test['data'].astype(float)
```

```
In [10]: f, ax = plt.subplots(2,10,figsize=(25,5))
         for i in range(10):
             ax[0,i].imshow(cv.cvtColor(X_train['data'][i].astype('uint8'),cv.COLOR_BGR2RGB));
             ax[1,i].imshow(cv.cvtColor(X_train['blur_data'][i].astype('uint8'),cv.COLOR_BGR2RGB
         plt.show()
```



** Defining CNN Model for Training Model **
The model has been trained on a much larger dataset of CelebA images.

In [11]:
```
deblur_CNN_input = Input(shape=(64,64,3))

#HIDDEN LAYERS
deblur_CNN_layer1 = Conv2D(filters=128, kernel_size=10, strides = 1, padding='same')(de
deblur_CNN_layer1 = BatchNormalization()(deblur_CNN_layer1)
deblur_CNN_layer1 = Activation('relu')(deblur_CNN_layer1)

deblur_CNN_layer2 = Conv2D(filters=320, kernel_size=1, strides = 1, padding='same')(deb
deblur_CNN_layer2 = BatchNormalization()(deblur_CNN_layer2)
deblur_CNN_layer2 = Activation('relu')(deblur_CNN_layer2)

deblur_CNN_layer3 = Conv2D(filters=320, kernel_size=1, strides = 1, padding='same')(deb
deblur_CNN_layer3= BatchNormalization()(deblur_CNN_layer3)
deblur_CNN_layer3 = Activation('relu')(deblur_CNN_layer3)

deblur_CNN_layer4 = Conv2D(filters=320, kernel_size=1, strides = 1, padding='same')(deb
deblur_CNN_layer4 = BatchNormalization()(deblur_CNN_layer4)
deblur_CNN_layer4 = Activation('relu')(deblur_CNN_layer4)

deblur_CNN_layer5 = Conv2D(filters=128, kernel_size=1, strides = 1, padding='same')(deb
deblur_CNN_layer5 = BatchNormalization()(deblur_CNN_layer5)
deblur_CNN_layer5 = Activation('relu')(deblur_CNN_layer5)

deblur_CNN_layer6 = Conv2D(filters=128, kernel_size=3, strides = 1, padding='same')(deb
deblur_CNN_layer6 = BatchNormalization()(deblur_CNN_layer6)
deblur_CNN_layer6 = Activation('relu')(deblur_CNN_layer6)

deblur_CNN_layer7 = Conv2D(filters=512, kernel_size=1, strides = 1, padding='same')(deb
deblur_CNN_layer7 = BatchNormalization()(deblur_CNN_layer7)
deblur_CNN_layer7 = Activation('relu')(deblur_CNN_layer7)

deblur_CNN_layer8 = Conv2D(filters=128, kernel_size=5, strides = 1, padding='same')(deb
deblur_CNN_layer8 = BatchNormalization()(deblur_CNN_layer8)
deblur_CNN_layer8 = Activation('relu')(deblur_CNN_layer8)

deblur_CNN_layer9 = Conv2D(filters=128, kernel_size=5, strides = 1, padding='same')(deb
deblur_CNN_layer9 = BatchNormalization()(deblur_CNN_layer9)
deblur_CNN_layer9 = Activation('relu')(deblur_CNN_layer9)

deblur_CNN_layer10 = Conv2D(filters=128, kernel_size=3, strides = 1, padding='same')(de
deblur_CNN_layer10 = BatchNormalization()(deblur_CNN_layer10)
deblur_CNN_layer10 = Activation('relu')(deblur_CNN_layer10)

deblur_CNN_layer11 = Conv2D(filters=128, kernel_size=5, strides = 1, padding='same')(de
deblur_CNN_layer11 = BatchNormalization()(deblur_CNN_layer11)
deblur_CNN_layer11 = Activation('relu')(deblur_CNN_layer11)

deblur_CNN_layer12 = Conv2D(filters=128, kernel_size=5, strides = 1, padding='same')(de
```

```
        deblur_CNN_layer12 = BatchNormalization()(deblur_CNN_layer12)
        deblur_CNN_layer12 = Activation('relu')(deblur_CNN_layer12)

        deblur_CNN_layer13 = Conv2D(filters=256, kernel_size=1, strides = 1, padding='same')(de
        deblur_CNN_layer13 = BatchNormalization()(deblur_CNN_layer13)
        deblur_CNN_layer13 = Activation('relu')(deblur_CNN_layer13)

        deblur_CNN_layer14 = Conv2D(filters=64, kernel_size=7, strides = 1, padding='same')(deb
        deblur_CNN_layer14 = BatchNormalization()(deblur_CNN_layer14)
        deblur_CNN_layer14 = Activation('relu')(deblur_CNN_layer14)

        deblur_CNN_output = Conv2D(filters=3, kernel_size=7, strides = 1, padding='same', activ

        deblur_CNN = Model(inputs= deblur_CNN_input, outputs=deblur_CNN_output )
```

In [12]:
```
deblur_CNN.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 64, 64, 3)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 128)       38528
_________________________________________________________________
batch_normalization_1 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_1 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 320)       41280
_________________________________________________________________
batch_normalization_2 (Batch (None, 64, 64, 320)       1280
_________________________________________________________________
activation_2 (Activation)    (None, 64, 64, 320)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 320)       102720
_________________________________________________________________
batch_normalization_3 (Batch (None, 64, 64, 320)       1280
_________________________________________________________________
activation_3 (Activation)    (None, 64, 64, 320)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 64, 64, 320)       102720
_________________________________________________________________
batch_normalization_4 (Batch (None, 64, 64, 320)       1280
_________________________________________________________________
activation_4 (Activation)    (None, 64, 64, 320)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 64, 64, 128)       41088
_________________________________________________________________
```

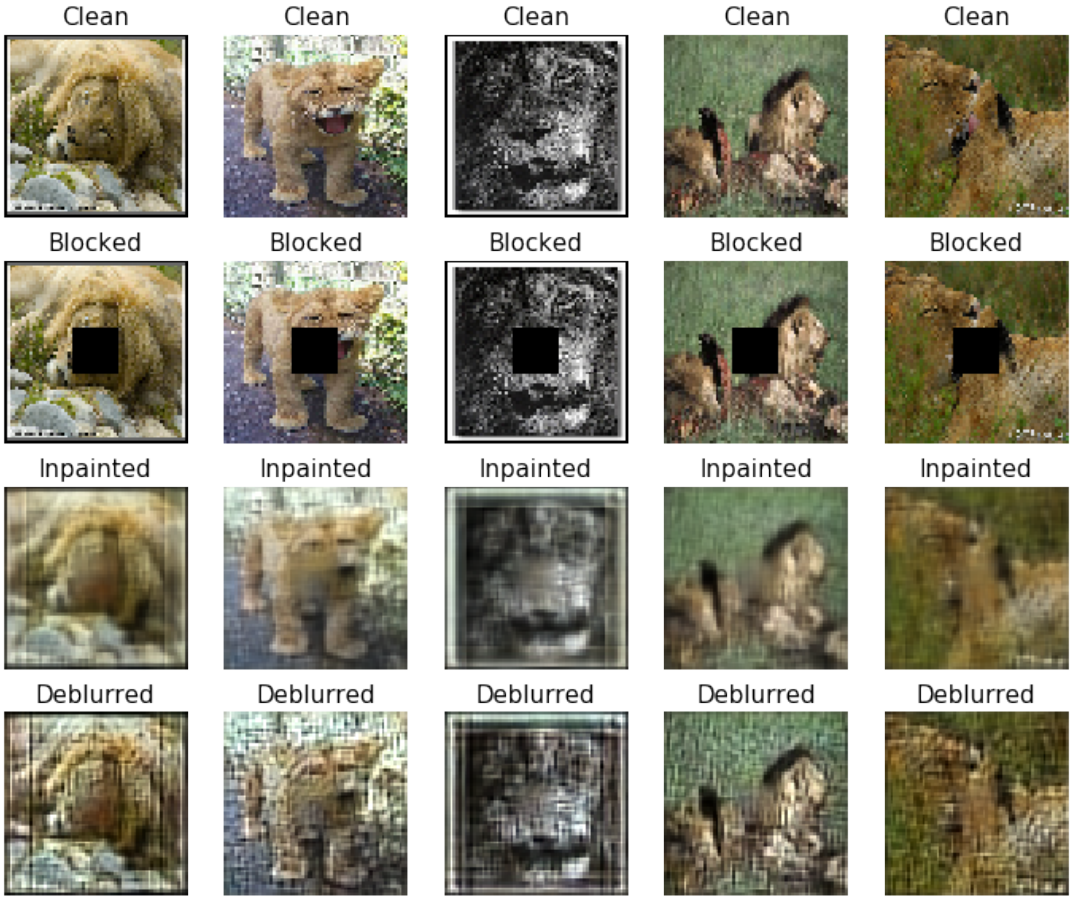
```
batch_normalization_5 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_5 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 64, 64, 128)       147584
_________________________________________________________________
batch_normalization_6 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_6 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 64, 64, 512)       66048
_________________________________________________________________
batch_normalization_7 (Batch (None, 64, 64, 512)       2048
_________________________________________________________________
activation_7 (Activation)    (None, 64, 64, 512)       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 64, 64, 128)       1638528
_________________________________________________________________
batch_normalization_8 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_8 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 64, 64, 128)       409728
_________________________________________________________________
batch_normalization_9 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_9 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 64, 64, 128)       147584
_________________________________________________________________
batch_normalization_10 (Batc (None, 64, 64, 128)       512
_________________________________________________________________
activation_10 (Activation)   (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 64, 64, 128)       409728
_________________________________________________________________
batch_normalization_11 (Batc (None, 64, 64, 128)       512
_________________________________________________________________
activation_11 (Activation)   (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 64, 64, 128)       409728
_________________________________________________________________
batch_normalization_12 (Batc (None, 64, 64, 128)       512
_________________________________________________________________
activation_12 (Activation)   (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 64, 64, 256)       33024
_________________________________________________________________
```

```
batch_normalization_13 (Batc (None, 64, 64, 256)       1024
_________________________________________________________________
activation_13 (Activation)   (None, 64, 64, 256)       0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 64, 64, 64)        802880
_________________________________________________________________
batch_normalization_14 (Batc (None, 64, 64, 64)        256
_________________________________________________________________
activation_14 (Activation)   (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 64, 64, 3)         9411
=================================================================
Total params: 4,411,843
Trainable params: 4,406,211
Non-trainable params: 5,632
_________________________________________________________________
```

```
In [13]: def plot_loss(train,valid):
             fig,ax = plt.subplots()
             plt.xlabel('Training epoches')
             plt.ylabel('Loss values')
             x = range(len(train))

             ax.plot(x,train,'g',label = 'train set')
             ax.plot(x,valid,'r',label = 'validation set')
             plt.grid(True)
             plt.legend(bbox_to_anchor=(1.0, 1), loc=1, borderaxespad=0.)
             plt.show()
             plt.pause(0.001)

In [14]: adam = Adam(lr= 0.00001)
         deblur_CNN.compile(optimizer= adam, loss= 'mean_squared_error')

In [15]: deblur_CNN.load_weights('weights_centre_merge_10.h5')

In [ ]: for i in range(50):
        #deblur_CNN.load_weights('original_weights.h5')
            hist = deblur_CNN.fit( X_train['blur_data'],X_train['data'],batch_size=16, validatio
            f = open('merge_val_loss.txt','a')
            for i in hist.history['val_loss']:
                f.write(str(i)+'\n')
            f.close()
            f = open('merge_train_loss.txt','a')
            for i in hist.history['loss']:
                f.write(str(i)+'\n')
            f.close()

In [ ]: print(hist.history)
```

```
In [ ]: f = open('merge_val_loss.txt','a')
        for i in hist.history['val_loss']:
            f.write(str(i)+'\n')
        f.close()
        f = open('merge_train_loss.txt','a')
        for i in hist.history['loss']:
            f.write(str(i)+'\n')
        f.close()

In [ ]: with open('merge_val_loss.txt', 'r') as f:
            data = f.readlines()
            val = [float(i) for i in data]
        with open('merge_train_loss.txt', 'r') as f:
            data = f.readlines()
            train = [float(i) for i in data]
        plot_loss(train,val)

In [ ]: '''
        deblur_CNN.load_weights('train7-11-21/weights_7_80_11_20_21_20.h5')
        X_test = pk.load("data64_blur21\\shf_test.gz")
        '''

In [ ]: deblur_CNN.load_weights('weights_center_merge_10.h5')

In [19]: deblur_CNN.load_weights('weights_de_merge_10.h5')

In [ ]: deblur_CNN.load_weights('weights21_merge_10.h5')

In [16]: Deblurred = deblur_CNN.predict(X_test['blur_data'])
         Deblurred = np.clip(Deblurred, 0, 255)
         Deblurred=Deblurred.astype(np.uint8)
         f, ax = plt.subplots(3,10, figsize=(30,10))
         for i in range(10):
             ax[0,i].imshow(cv.cvtColor(X_test['data'][i+100].astype('uint8'),cv.COLOR_BGR2RGB))
             ax[1,i].imshow(cv.cvtColor(X_test['blur_data'][i+100].astype('uint8'),cv.COLOR_BGR2
             ax[2,i].imshow(cv.cvtColor(Deblurred[i+100].astype('uint8'),cv.COLOR_BGR2RGB));  ax
         plt.show()
```

```
In [20]: Deblurred = deblur_CNN.predict(X_test['inpaint_data'])
         Deblurred = np.clip(Deblurred, 0, 255)
         Deblurred=Deblurred.astype(np.uint8)
         f, ax = plt.subplots(4,5, figsize=(12,10))
         for i in range(5):
             ax[0,i].imshow(cv.cvtColor(X_test['data'][i+208].astype('uint8'),cv.COLOR_BGR2RGB))
             ax[1,i].imshow(cv.cvtColor(X_test['blur_data'][i+208].astype('uint8'),cv.COLOR_BGR2
             ax[2,i].imshow(cv.cvtColor(X_test['inpaint_data'][i+208].astype('uint8'),cv.COLOR_B
             ax[3,i].imshow(cv.cvtColor(Deblurred[i+208].astype('uint8'),cv.COLOR_BGR2RGB));  ax
         plt.show()
```




```
In [ ]: deblur_CNN.save_weights('weights_centre_merge_10.h5')

In [17]: X_test['inpaint_data'] = Deblurred

In [21]: X_test['deblur_data'] = Deblurred
```

```
In [ ]: pk.dump(X_test, "test.gz", overwrite=True)

In [ ]: X_test = pk.load("test.gz")
        X_test['data'].shape

In [ ]: pk.dump(X_deblur, "train7-11-21/Deblurred_step_for21.gz", overwrite=True)

In [28]: X_test['mask_data']= np.copy(X_test['blur_data'])
         X_test['mask_data'][:,24:40,24:40,:] = np.copy(X_test['deblur_data'][:,24:40,24:40,:])
         f, ax = plt.subplots(5,5, figsize=(20,20))
         for i in range(5):
             ax[0,i].imshow(cv.cvtColor(X_test['data'][i+108].astype('uint8'),cv.COLOR_BGR2RGB))
             ax[1,i].imshow(cv.cvtColor(X_test['blur_data'][i+108].astype('uint8'),cv.COLOR_BGR2
             ax[2,i].imshow(cv.cvtColor(X_test['inpaint_data'][i+108].astype('uint8'),cv.COLOR_B
             ax[3,i].imshow(cv.cvtColor(X_test['deblur_data'][i+108].astype('uint8'),cv.COLOR_BG
             ax[4,i].imshow(cv.cvtColor(X_test['mask_data'][i+108].astype('uint8'),cv.COLOR_BGR2
         plt.show()
```