NSFW CLASSIFIER

The NSFW Classifier is used to Classify Not safe for work images vs Safe images.
NSFW images includes Porn and Sexy images.
Rest are classified as Safe Images.

This model is hugely inspired by https://www.freecodecamp.org/news/how-to-set-up-nsfw-content-detection-with-machine-learning-229a9725829c/ (https://www.freecodecamp.org/news/how-to-set-up-nsfw-content-detection-with-machine-learning-229a9725829c/) , https://github.com/GantMan/nsfw_model (https://github.com/GantMan/nsfw_model) and his work. Some modification is made and model architecture is changed.

The Data for this project is collected with help of scripts at https://github.com/alex000kim/nsfw_data_scraper (https://github.com/alex000kim/nsfw_data_scraper)

Mobile Net Architecture is used for the classification as it is very fast and has less params to train.

An ios app is also made and its screen recording is uploaded at https://github.com/lakshaychhabra/NSFW-ios-ML (https://github.com/lakshaychhabra/NSFW-ios-ML)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from keras.preprocessing.image import ImageDataGenerator
from keras.backend import clear_session
from keras.optimizers import SGD, Adam
from pathlib import Path
from keras.applications.mobilenet_v2 import MobileNetV2
from keras.models import Sequential, Model, load_model
from keras.layers import Dense, Dropout, Flatten, AveragePooling2D, BatchNormalization
from keras import initializers, regularizers
from pathlib import Path
from keras.callbacks import ModelCheckpoint, TensorBoard, ReduceLROnPlateau, History, LearningRateScheduler
from datetime import datetime
import warnings
warnings.filterwarnings("ignore")
import os
import matplotlib.image as mpimg
```

```
Using TensorFlow backend.
```

In [2]:
```
!pwd
```

```
/home/lakshaychhabralc7/nsfw/data
```

In [3]:
```
train_neutral = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/train/neutral')]))
test_neutral = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/test/neutral')]))
train_porn = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/train/porn')]))
train_sexy = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/train/sexy')]))
test_porn = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/test/porn')]))
test_sexy = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/test/sexy')]))
```

In [4]:
```
train_data = [train_neutral, train_porn, train_sexy]
test_data = [test_neutral, test_porn, test_sexy]
```

In [5]:
```
print("Total number of train data is : ", train_data[0], "+", train_data[1], "+", train_data[2],"=", sum(tr
print("Total number of test data is : ", test_data[0], "+", test_data[1], "+", test_data[2],"=", sum(test_d
```

```
Total number of train data is :  34387 + 55581 + 17191 = 107159
Total number of test data is :  2000 + 2000 + 2000 = 6000
```

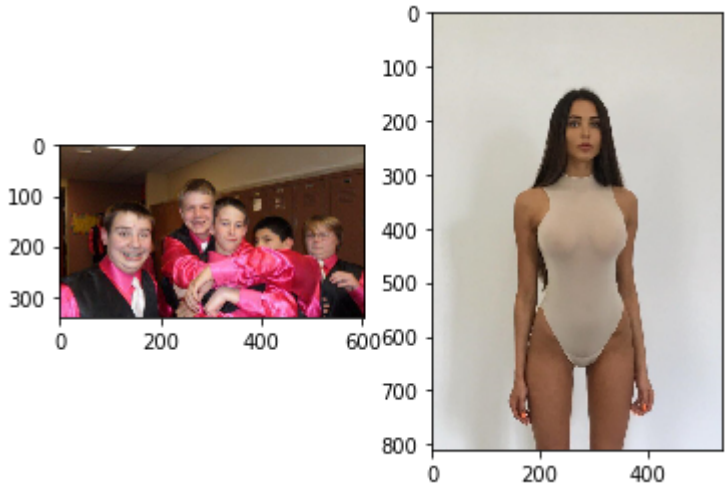
In [6]:
```
train_path =r"/home/lakshaychhabralc7/nsfw/data/train"
test_path = r"/home/lakshaychhabralc7/nsfw/data/test"
```

So we Have 107k images to train and have 6k images to test

In [7]:
```
print("Example of the data Neutral and Sexy category")
f, (ax1, ax2) = plt.subplots(1, 2)
img=mpimg.imread(test_path+"/neutral/ffdb5729-8bac-4add-bbc1-41d1e428c842.jpg")
ax1.imshow(img)
img=mpimg.imread(test_path+"/sexy/ffc15b09-10a0-4753-9adf-d38eb53cf8a1.jpg")
ax2.imshow(img)
```

```
Example of the data Neutral and Sexy category
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x7f79ec883588>
```



We need a fast model which gives high accuracy and also have less parsms to train.
We are choosing MobileNet for it.

We will use Transfer Learning and choose weights which were trained for ImageNet.

In [8]:
```
# As we know the input size in ImageNet was 224 so we have to resize our images accordingly
size = 224
epochs = 100
steps = 500

```

In [9]:
```
# We have to take in account different angle of images and to avoid overfit we will use Data Generator,
# More the Merrier
train_data_generation = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    channel_shift_range=20,
    horizontal_flip=True
)
validation_data_generation = ImageDataGenerator(
    rescale=1./255 #need float values
)
```

In [10]:
```
train_generator = train_data_generation.flow_from_directory(
        train_path,
        target_size=(size, size),
        class_mode='categorical',
        batch_size = 64
    )

validation_generator = validation_data_generation.flow_from_directory(
    test_path,
    target_size=(size, size),
    class_mode='categorical',
    batch_size = 64
)
```

```
Found 107159 images belonging to 3 classes.
Found 6000 images belonging to 3 classes.
```

In [11]:
```
# from keras.backend import clear_session
# clear_session()
# import tensorflow as tf
# from keras.backend.tensorflow_backend import set_session
# config = tf.ConfigProto()
# config.gpu_options.allow_growth = True  # dynamically grow the memory used on the GPU
# sess = tf.Session(config=config)
# set_session(sess)  # set this TensorFlow session as the default session for Keras
```

DL Model

In [12]:
```
conv_m = MobileNetV2(weights='imagenet', include_top=False, input_shape=(size, size, 3))
conv_m.trainable = False
conv_m.summary()
```

```
block_15_add (Add)              (None, 7, 7, 160)    0           block_14_add[0][0]
                                                                 block_15_project_BN[0][0]

__________________________________________________________________________________________
block_16_expand (Conv2D)        (None, 7, 7, 960)    153600      block_15_add[0][0]

__________________________________________________________________________________________
block_16_expand_BN (BatchNormal (None, 7, 7, 960)    3840        block_16_expand[0][0]

__________________________________________________________________________________________
block_16_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_16_expand_BN[0][0]

__________________________________________________________________________________________
block_16_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_16_expand_relu[0][0]

__________________________________________________________________________________________
block_16_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_16_depthwise[0][0]

__________________________________________________________________________________________
block_16_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_16_depthwise_BN[0][0]

__________________________________________________________________________________________
block_16_project (Conv2D)       (None, 7, 7, 320)    307200      block_16_depthwise_relu[0][0]

__________________________________________________________________________________________
block_16_project_BN (BatchNorma (None, 7, 7, 320)    1280        block_16_project[0][0]

__________________________________________________________________________________________
Conv_1 (Conv2D)                 (None, 7, 7, 1280)   409600      block_16_project_BN[0][0]
```

In [ ]:
```

```

In [36]:
```
model = Sequential()
model.add(conv_m)
model.add(AveragePooling2D(pool_size=(7, 7)))
model.add(Flatten())
model.add(Dense(32, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
mobilenetv2_1.00_224 (Model) (None, 7, 7, 1280)        2257984   
_________________________________________________________________
average_pooling2d_5 (Average (None, 1, 1, 1280)        0         
_________________________________________________________________
flatten_7 (Flatten)          (None, 1280)              0         
_________________________________________________________________
dense_19 (Dense)             (None, 32)                40992     
_________________________________________________________________
batch_normalization_7 (Batch (None, 32)                128       
_________________________________________________________________
dropout_11 (Dropout)         (None, 32)                0         
_________________________________________________________________
dense_20 (Dense)             (None, 3)                 99        
=================================================================
Total params: 2,299,203
Trainable params: 41,155
Non-trainable params: 2,258,048
_________________________________________________________________
```

In [11]:
```
from time import time
filepath = "bestweight.h5"
checkpoint = ModelCheckpoint("weights{epoch:05d}.h5", monitor='val_acc', verbose=1, save_best_only=True, mo
lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=np.sqrt(0.1), patience=5, verbose=1, cooldown=0, m
callbacks = [checkpoint, lr_reduce]


```

In [ ]:
```
model.compile(
    loss='categorical_crossentropy',
    optimizer=SGD(lr = 0.1, momentum = 0.9),
    metrics=['accuracy']
)
```

In [15]:
```
model = load_model("nsfwnsfw_mobilenet2_30.h5")
```

In [ ]:
```

```

In [17]:
```
start = datetime.now()
history = model.fit_generator(
    train_generator,
    callbacks=callbacks,
    epochs=100,
    steps_per_epoch=10,
    validation_data=validation_generator,
    validation_steps=10,
    initial_epoch = 30
)
```

```
10/10 [==============================] - 150s 15s/step - loss: 0.2795 - acc: 0.8984 - val_loss: 0.2500 - val_
acc: 0.9047

Epoch 00073: val_acc did not improve from 0.92344
Epoch 74/100
10/10 [==============================] - 152s 15s/step - loss: 0.2443 - acc: 0.9016 - val_loss: 0.2477 - val_
acc: 0.9000

Epoch 00074: val_acc did not improve from 0.92344
Epoch 75/100
10/10 [==============================] - 152s 15s/step - loss: 0.2644 - acc: 0.9047 - val_loss: 0.2303 - val_
acc: 0.9172

Epoch 00075: val_acc did not improve from 0.92344
Epoch 76/100
10/10 [==============================] - 153s 15s/step - loss: 0.2689 - acc: 0.9219 - val_loss: 0.2325 - val_
acc: 0.9125

Epoch 00076: val_acc did not improve from 0.92344
```

In [18]:
```
print("time taken : ", datetime.now() - start)
```

```
time taken :  2:58:14.032359
```

In [19]:
```
history = model.fit_generator(
    train_generator,
    callbacks=callbacks,
    epochs=100,
    steps_per_epoch=10,
    validation_data=validation_generator,
    validation_steps=10,
    initial_epoch = 78
)
```

```
Epoch 00093: val_acc did not improve from 0.93590
Epoch 94/100
10/10 [==============================] - 153s 15s/step - loss: 0.2603 - acc: 0.9062 - val_loss: 0.2282 - val_
acc: 0.9187

Epoch 00094: val_acc did not improve from 0.93590
Epoch 95/100
10/10 [==============================] - 152s 15s/step - loss: 0.2352 - acc: 0.9172 - val_loss: 0.2366 - val_
acc: 0.9094

Epoch 00095: val_acc did not improve from 0.93590
Epoch 96/100
10/10 [==============================] - 154s 15s/step - loss: 0.2919 - acc: 0.8984 - val_loss: 0.2249 - val_
acc: 0.9215

Epoch 00096: val_acc did not improve from 0.93590
Epoch 97/100
10/10 [==============================] - 156s 16s/step - loss: 0.2769 - acc: 0.9047 - val_loss: 0.2309 - val_
acc: 0.9141
```

In [43]:
```
# model.save("nsfwnsfw_mobilenet2_100.h5")
```

In [ ]:
```

```

In [91]:
```
model = load_model("weights00077.h5")
```

In [93]:
```
import coremltools
model.author = "Lakshay Chhabra"
model.short_description = "NSFW Image Classifier"
```

In [94]:
```
output_labels = ['Neutral', 'Porn', 'Sexy']
ios = coremltools.converters.keras.convert(model, input_names=['image'], output_names=['output'],
                                           class_labels=output_labels, image_input_names='image', i
```

```
48933320>
116 : mobilenetv2_1.00_224_block_12_add, <keras.layers.merge.Add object at 0x1c48933438>
117 : mobilenetv2_1.00_224_block_13_expand, <keras.layers.convolutional.Conv2D object at 0x1c48933470>
118 : mobilenetv2_1.00_224_block_13_expand_BN, <keras.layers.normalization.BatchNormalization object at 0x1c4
89335f8>
119 : mobilenetv2_1.00_224_block_13_expand_relu, <keras.layers.advanced_activations.ReLU object at 0x1c489337
10>
120 : mobilenetv2_1.00_224_block_13_pad, <keras.layers.convolutional.ZeroPadding2D object at 0x1c48933748>
121 : mobilenetv2_1.00_224_block_13_depthwise, <keras.layers.convolutional.DepthwiseConv2D object at 0x1c4893
3780>
122 : mobilenetv2_1.00_224_block_13_depthwise_BN, <keras.layers.normalization.BatchNormalization object at 0x
1c489337f0>
123 : mobilenetv2_1.00_224_block_13_depthwise_relu, <keras.layers.advanced_activations.ReLU object at 0x1c489
33ac8>
124 : mobilenetv2_1.00_224_block_13_project, <keras.layers.convolutional.Conv2D object at 0x1c48933b00>
125 : mobilenetv2_1.00_224_block_13_project_BN, <keras.layers.normalization.BatchNormalization object at 0x1c
48933c88>
126 : mobilenetv2_1.00_224_block_14_expand, <keras.layers.convolutional.Conv2D object at 0x1c48933da0>
127 : mobilenetv2_1.00_224_block_14_expand_BN, <keras.layers.normalization.BatchNormalization object at 0x1c4
```

In [95]:
```
ios.save('NSFW.mlmodel')
```

In [ ]:
```

```

In [69]:
```
test_file = test_path+"/sexy/ffc15b09-10a0-4753-9adf-d38eb53cf8a1.jpg"
```

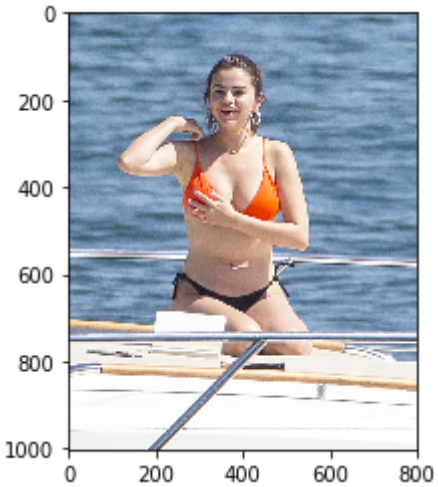
In [105]:

```
# https://stackoverflow.com/a/53403805/7437264
from PIL import Image
import numpy as np
from skimage import transform
def load(filename):
    np_image = Image.open(filename)
    np_image = np.array(np_image).astype('float32')/255
    np_image = transform.resize(np_image, (224, 224, 3))
    np_image = np.expand_dims(np_image, axis=0)
    img=mpimg.imread(filename)
    plt.imshow(img)
    return np_image

image = load("8.jpg")
ans = model.predict(image)
maping = {0 : "Neutral", 1 : "Porn", 2 : "Sexy"}
new_ans = np.argmax(ans[0])

print(maping[new_ans], np.round(ans,2))
print("With {} probability".format(ans[0][new_ans]))
```

```
Sexy [[0.02 0.02 0.96]]
With 0.9588373303413391 probability
```



Summary

1. This Model unable to classify drawings and Anime as it is not trained for them.
2. It fails to classify Male genitalia because images in train data are mostly of females.
3. The accuracy can further be improved as I was limited by resources and don't have a GPU, so further training can increase accuracy.
4. Future Goals : To add bounded box with help or YOLO or Sliding window or any other object detection algo and classify video data live.