

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]: 1 import numpy as np
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 %matplotlib inline
5 from keras.preprocessing.image import ImageDataGenerator
6 from keras.backend import clear_session
7 from keras.optimizers import SGD, Adam
8 from pathlib import Path
9 from keras.applications.mobilenet_v2 import MobileNetV2
10 from keras.models import Sequential, Model, load_model
11 from keras.layers import Dense, Dropout, Flatten, AveragePooling2D, BatchNormalization
12 from keras import initializers, regularizers
13 from pathlib import Path
14 from keras.callbacks import ModelCheckpoint, TensorBoard, ReduceLROnPlateau, History, LearningRateScheduler
15 from datetime import datetime
16 import warnings
17 warnings.filterwarnings("ignore")
18 import os
19 import matplotlib.image as mpimg
```

Using TensorFlow backend.

```
In [2]: 1 !pwd
```

/home/lakshaychhabralc7/nsfw/data

```
In [3]: 1 train_neutral = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/train/neutral')]))
2 test_neutral = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/test/neutral')]))
3 train_porn = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/train/porn')]))
4 train_sexy = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/train/sexy')]))
5 test_porn = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/test/porn')]))
6 test_sexy = (len([iq for iq in os.scandir('/home/lakshaychhabralc7/nsfw/data/test/sexy')]))
```

```
In [4]: 1 train_data = [train_neutral, train_porn, train_sexy]
2 test_data = [test_neutral, test_porn, test_sexy]
```

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

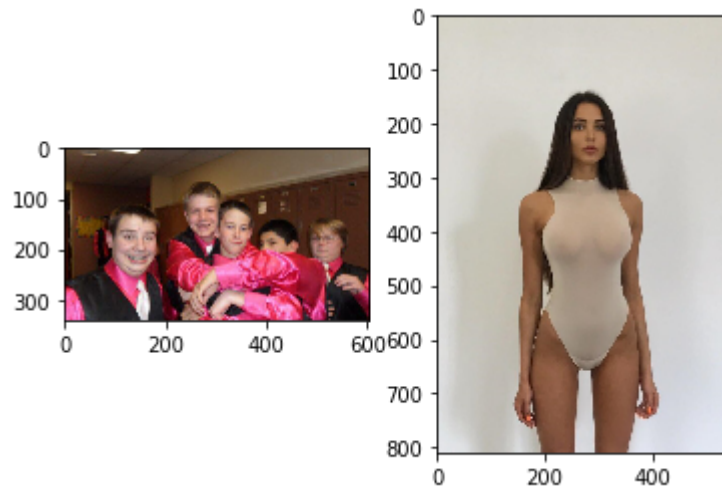
```
In [6]: 1 train_path = r"/home/lakshaychhabralc7/nsfw/data/train"
2 test_path = r"/home/lakshaychhabralc7/nsfw/data/test"
```

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

```
In [7]: 1 print("Example of the data Neutral and Sexy category")
2 f, (ax1, ax2) = plt.subplots(1, 2)
3 img=mpimg.imread(test_path+"/neutral/ffdb5729-8bac-4add-bbc1-41d1e428c842.jpg")
4 ax1.imshow(img)
5 img=mpimg.imread(test_path+"/sexy/ffc15b09-10a0-4753-9adf-d38eb53cf8a1.jpg")
6 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]: 1 # As we know the input size in ImageNet was 224 so we have to resize our images accordingly
2 size = 224
3 epochs = 100
4 steps = 500
5
```

```
In [9]: 1 # We have to take in account different angle of images and to avoid overfit we will use Data Generator,
2 # More the Merrier
3 train_data_generation = ImageDataGenerator(
4     rescale=1./255,
5     rotation_range=30,
6     width_shift_range=0.2,
7     height_shift_range=0.2,
8     shear_range=0.2,
9     zoom_range=0.2,
10    channel_shift_range=20,
11    horizontal_flip=True
12 )
13 validation_data_generation = ImageDataGenerator(
14     rescale=1./255 #need float values
15 )
```

```
In [10]: 1 train_generator = train_data_generation.flow_from_directory(  
2         train_path,  
3         target_size=(size, size),  
4         class_mode='categorical',  
5         batch_size = 64  
6     )  
7  
8     validation_generator = validation_data_generation.flow_from_directory(  
9         test_path,  
10        target_size=(size, size),  
11        class_mode='categorical',  
12        batch_size = 64  
13    )
```

Found 107159 images belonging to 3 classes.

Found 6000 images belonging to 3 classes.

```
In [11]: 1 # from keras.backend import clear_session  
2 # clear_session()  
3 # import tensorflow as tf  
4 # from keras.backend.tensorflow_backend import set_session  
5 # config = tf.ConfigProto()  
6 # config.gpu_options.allow_growth = True # dynamically grow the memory used on the GPU  
7 # sess = tf.Session(config=config)  
8 # set_session(sess) # set this TensorFlow session as the default session for Keras
```

DL Model

```
In [12]: 1 conv_m = MobileNetV2(weights='imagenet', include_top=False, input_shape=(size, size, 3))
2 conv_m.trainable = False
3 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 [ ]: 1
```

```
In [36]: 1 model = Sequential()
2 model.add(conv_m)
3 model.add(AveragePooling2D(pool_size=(7, 7)))
4 model.add(Flatten())
5 model.add(Dense(32, activation = 'relu'))
6 model.add(BatchNormalization())
7 model.add(Dropout(0.5))
8 model.add(Dense(3, activation='softmax'))
9 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 Normalization)	(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]: 1 from time import time
2 filepath = "bestweight.h5"
3 checkpoint = ModelCheckpoint("weights{epoch:05d}.h5", monitor='val_acc', verbose=1, save_best_only=True, mo
4 lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=np.sqrt(0.1), patience=5, verbose=1, cooldown=0, m
5 callbacks = [checkpoint, lr_reduce]
6
7
```

```
In [ ]: 1 model.compile(  
2     loss='categorical_crossentropy',  
3     optimizer=SGD(lr = 0.1, momentum = 0.9),  
4     metrics=['accuracy']  
5 )
```

```
In [15]: 1 model = load_model("nsfwnsfw_mobilenet2_30.h5")
```

```
In [ ]: 1
```



```
In [17]: 1 start = datetime.now()
2 history = model.fit_generator(
3     train_generator,
4     callbacks=callbacks,
5     epochs=100,
6     steps_per_epoch=10,
7     validation_data=validation_generator,
8     validation_steps=10,
9     initial_epoch = 30
10 )
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]: 1 print("time taken : ", datetime.now() - start)

time taken : 2:58:14.032359
```

```
In [19]: 1 history = model.fit_generator(  
2         train_generator,  
3         callbacks=callbacks,  
4         epochs=100,  
5         steps_per_epoch=10,  
6         validation_data=validation_generator,  
7         validation_steps=10,  
8         initial_epoch = 78  
9     )
```

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]: 1 # model.save("nsfwnsfw_mobilenet2_100.h5")
```

```
In [ ]: 1
```

```
In [91]: 1 model = load_model("weights00077.h5")
```

```
In [93]: 1 import coremltools  
2 model.author = "Lakshay Chhabra"  
3 model.short_description = "NSFW Image Classifier"
```

```
In [94]: 1 output_labels = ['Neutral', 'Porn', 'Sexy']
        2 ios = coremltools.converters.keras.convert(model, input_names=['image'], output_names=['output'],
        3                                           class_labels=output_labels, image_input_names='image', i
        4
```

```
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
8933e00>
```

```
In [95]: 1 ios.save('NSFW.mlmodel')
```

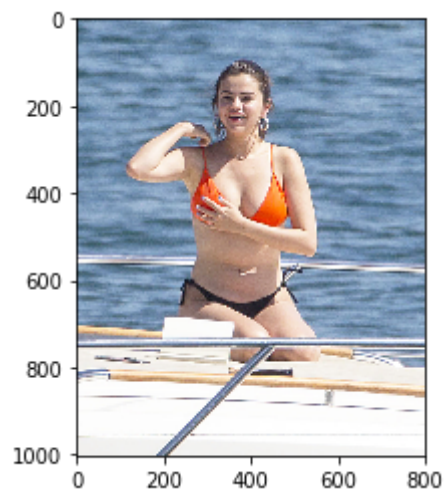
```
In [ ]: 1
```

```
In [69]: 1 test_file = test_path+"/sexy/ffc15b09-10a0-4753-9adf-d38eb53cf8a1.jpg"
```

```
In [105]: 1 # https://stackoverflow.com/a/53403805/7437264
2 from PIL import Image
3 import numpy as np
4 from skimage import transform
5 def load(filename):
6     np_image = Image.open(filename)
7     np_image = np.array(np_image).astype('float32')/255
8     np_image = transform.resize(np_image, (224, 224, 3))
9     np_image = np.expand_dims(np_image, axis=0)
10    img=mpimg.imread(filename)
11    plt.imshow(img)
12    return np_image
13
14 image = load("8.jpg")
15 ans = model.predict(image)
16 mapping = {0 : "Neutral", 1 : "Porn", 2 : "Sexy"}
17 new_ans = np.argmax(ans[0])
18
19 print(mapping[new_ans], np.round(ans,2))
20 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.