Machine Learning Engineer Nanodegree

Capstone Project - Image Recognition to Detect Multiple Attributes

Marcos Alvarado
July 1st, 2018

1. Definition

Project Overview

Image recognition is one of the many applications of Machine Learning, it can solve problems for security purposes, object detection, face detection, healthcare, entertainment, among others. This application has an enormous potential to help our society[1], so it is important to find new uses for this tool, improve the current methods and get more accurate and useful insights from it. A good example of these applications, is the work done by research of The Chinese University of Hong Kong for face detection using deep learning[2]:

"we propose a novel deep convolutional network (DCN) that achieves outstanding performance on FDDB, PASCAL Face, and AFW." (From Facial Parts Responses to Face Detection: A Deep Learning Approach, Department of Information Engineering, The Chinese University of Hong Kong)

In this project, the researcher creates three models to predict three attributes out of the same picture, it uses the Inception-V3 as based model for image recognition[3][4], and the data set uses is the CelebA dataset[5], which contains over 200,000 celebrity images and 40 binary attribute annotations per image.

The topic of this project was chosen by the researcher his interest and new knowledge acquired coursing the MLND Program.

Problem Statement

Many machine learning algorithms used in face recognition or object detection are built to detect a specific attribute, for example, the mood of a person in a picture, the algorithm will predict if the person is happy, sad, neutral, surprised, etc. What if we want to explore many non-exclusive attributes that are of a picture at the same time? As to classify if the person is smiling or not and at the same time know if is wearing eyeglasses or if is wearing a hat. Most probably the same algorithm will not perform well

[1] https://medium.com/the-mission/smart-cities-and-image-recognition-c717bf683fd5
[2] https://arxiv.org/abs/1509.06451
[3] https://www.tensorflow.org/tutorials/image_recognition
[4] https://arxiv.org/abs/1512.00567
[5] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

predicting two or more attributes. This is a classification problem for each of the selected attributes, one model per attribute, but using the same image. The output will be the predicted attribute of each model.

The use of different algorithms, one of them predicting a specific attribute will give us better insights of the picture to be analyzed, results will be measure against the real target and the algorithms will be replicable to pictures outside of the dataset used in this project.

The goal is build three classification models based on image recognition, the attributes to be predicted are:

- If the subject is smiling or not.
- If the subject is female or male
- If the subject is young or not

Metrics

The metrics to measure the performance of the models are:

- **Accuracy**

  Accuracy is a common metric to measure binary classifiers, this adapts well to the problem to be solved, the corresponding formula to this metrics is:

$$accuracy = \frac{True\ Positives + True\ Negatives}{Total\ number\ of\ pictures}$$

  Where:

  - True Positives: Number of times the model predicted to the positive class and it actually was the positive class.
  - True Negatives: Number of times the model predicted to the negative class and it actually was the negative class.
  - Total number of pictures: Total number of pictures included in the test data set.

- **F1 Score**

  F1 score is a metric that consider **precision** and **recall** for test accuracy. This measure gives relevant information about how the positive class is being predicted. The perfect model will have a value of 1 and the worst a value of 0.

$$F1\ score = 2 * \frac{precision * recall}{precision + recall}$$

  Where:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

2. Analysis

Data Exploration

The input data for this project is the CelebA dataset[6], which contains 202,599 number of face images of various celebrities and 40 binary attribute annotations per image, some of the attributes are: male, bald, wearing lipstick, wavy or straight hair, etc. For the purpose of this project, the selected attributes are: **Smiling**, **Male** and **Young**. For each of these attributes a binary classification model will be created.

Images are in format *.jpg with dimension 178x218x3 RGB.

			
File name: 197126.jpg • Not Smiling • Female • Young	File name: 193309.jpg • Smiling • Female • Young	File name: 202506.jpg • Smiling • Male • Young	File name: 195516.jpg • Not Smiling • Male • Not Young

Table 1: Sample images and labels.

In order to train the model, the data set include a recommended partition: Images 1-162770 are training, 162771-182637 are validation, 182638-202599 are testing. This recommendation will be used on this project to select images and create sub sets.

Exploratory Visualization

The plots below show how is the distribution of the selected labels across the data set. In figure 1 the **Smiling** attribute is balanced in the data set, for the **Male** attribute, the chart shows there are more female celebrities than males and for the Young attribute, most of the images are of young celebrities.

This analysis indicates that it will be necessary balance the data set for each model in order to train them. Otherwise, the performance of the model will be impacted[7].

For the information needed for this project, there are no abnormalities, which means that every image has its label.

[6] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
[7] https://www.kth.se/social/files/588617ebf2765401cfcc478c/PHensmanDMasko_dkand15.pdf

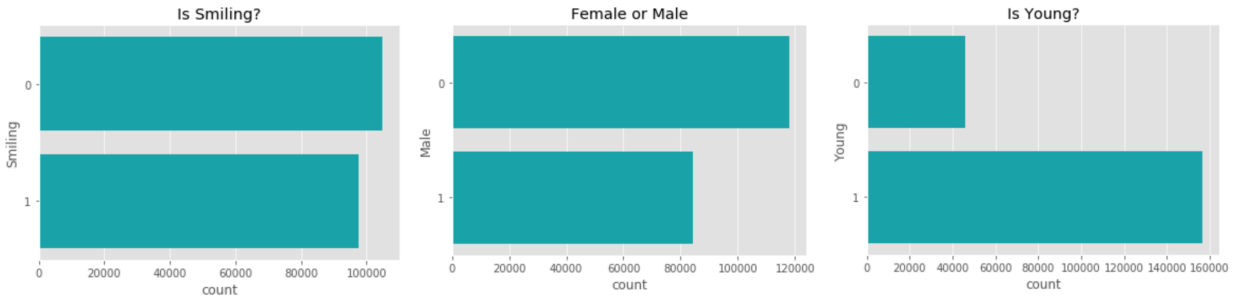

Figure 1: Distribution of labels across the data set.

Algorithms and Techniques

The classifier is a **Convolutional Neural Network** for each model (one for each of the selected attributes: Smiling, Male and Young). The models will be based in the Inception-V3 model trained with the ImageNet dataset[8] and using transfer learning to build the new classifiers. Also, Data Augmentation for image preprocessing will be used.

- **Data Augmentation**

  Data Augmentation allows to generate images with modifications to the original ones, this allows the model to learn from these variations (changing angle, size and position), being able to predict better never seen images that could have the same variations. The parameters to be modified in this projects for data augmentation are:

  - rotation_range: Degree range for random rotations.
  - width_shift_range: Width shift range to move the image.
  - height_shift_range: Height shift range to move the image.
  - shear_range: Shear angle in counter-clockwise direction in degrees.
  - zoom_range: Range for random zoom.
  - horizontal_flip: flip images horizontally.

  For more details about the available parameters: https://keras.io/preprocessing/image/.
  For Preprocessing parameters and example of the results in this project: See "Data Preprocessing" section.

- **Transfer Learning**

  The justification of this technique (transfer learning) is because of the fact that DNNs are computationally expensive to train, in order to get good result is necessary to train using millions of labeled images and machine equipped with GPUs, and this is what I Inception-V3 offers: a pre-trained

[8] http://www.image-net.org

model that has learn from millions of images, being able to identify in lower layers features such as texture, colors, contrast, etc. and is able to transfer this knowledge to new model, where some layers can be removed in other to continue training the network to solve a new problem (classification problem in this case).

The parameters than can be tuned using CNNs transfer learning are:

- Number of layers: Number of layers to be used by the base model.
- Trainable layers: Layers that will be trained, the remaining ones are locked and weights unmodified after new training.
- Type of layers: Fully connected, convolutional or pooling.
- Activation functions: Function uses to calculate the output of a layer.
- Optimizer: Optimizer technique used to train[9].
- Learning rate: Intensity in the model to learn during training.
- Momentum: in SGD (Stochastic gradient descent optimizer), parameter that takes into account previous steps to keep learning.
- Loss: Optimization score function[10].
- Metrics: metrics to be evaluated by the model during training and testing.
- Epochs: Number of iterations over the entire data provided.

More tunable parameters: https://keras.io/models/model/

- **Reduce the number of samples**

Due to computing resource limitations of the researcher to process all the images, the number of the images has been reduced. Basically, using a Mac Book Pro 2017 (2.9 GHz Intel Core i7, 16 GB RAM and 4 GB GPU), it will take days to train each model, stressing the machine and with risk to burn some pieces of hardware, as the CPUs reached over 100º Celsius, even using an external fan; GPU processing is not supported for MacOS by TensorFlow since release 1.2 (https://www.tensorflow.org/install/install_mac). Instead, the researcher is using a Windows 10 Machine with 2.9 GHz Intel Core i7, 16GB RAM, 2GB NVIDIA GeForce GTX 950M), which accelerate the process thanks to the use of GPU.

Also, additional software has been installed in order to run TensorFlow on GPU, such as CUDA[11]. The guide to install it is in TensorFlow website[12].

The number of images to be used by each model is:

- Training: 20000 images
- Validation: 5000 images
- Test: 5000 Images

[9] https://keras.io/optimizers/
[10] https://keras.io/losses/
[11] https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/
[12] https://www.tensorflow.org/install/install_windows

Benchmark

The selected benchmark is the OpenCV[13] library, that has built an algorithm for smile detection based on Haar Cascade[14]. It does not exist a measure of the OpenCV Smile Detection algorithm using the CelebA Data Set, therefore, the researcher will compare the CNN model for smile detection and the OpenCV model using the test data set and compare the accuracy of both models.

3. Methodology

Data Preprocessing

The steps for data preprocessing used on this project are:

- **Images to Arrays**

  Inception-V3 trained model is based on TensorFlow, this requires as input a 4D numpy array. Keras and TensorFlow offer this solution out of the box using the ***preprocess_input*** method[15].

- **Data Augmentation**

  As explained in the Algorithms and Techniques section, Data Augmentation has been used to generate images with modifications to the original ones, this allows the model to learn from these variations (changing angle, size and position), being able to predict better never seen images that could have the same variations.

  Parameters has been tuned as below:

  - rotation_range = 30
  - width_shift_range = 0.2
  - height_shift_range = 0.2
  - shear_range = 0.2
  - zoom_range = 0.2
  - horizontal_flip = True

[13] https://opencv.org
[14] https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html
[15] https://www.tensorflow.org/api_docs/python/tf/keras/applications/inception_v3/preprocess_input

The result is a data generator of modified images that will be used as input to train the model.

Example of the result using Data Augmentation:

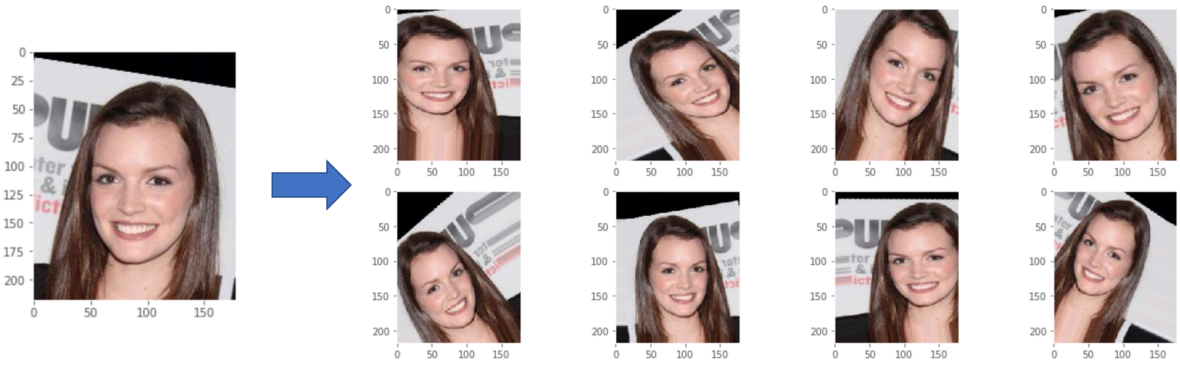

Figure 3: Preprocessed Image, Data Augmentation Process

Implementation

This section will be split into the following steps**:**

- **Files Allocation**

  In order to take advantage of the **keras.preprocessing.image.flow_from_directory**[16] method, which takes images from a specified directory, which have to contain subdirectories, with the name of each label, which contain the corresponding images, the researcher has created some functions to automate this job and make it replicable for all the models.

  As result, the automation will create folders for each model, for each type of data (training, validation and test) and each label; filled with the corresponding images, which also will be balanced. Example for the Smiling Model:

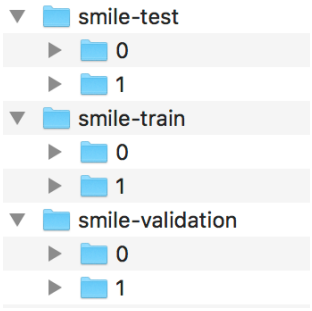

Figure 4: Directory structure for Smiling Model

[16] https://keras.io/preprocessing/image/

- **Data Generators**

  Data Generators has been created using the directories created in the previous step, the method used to create them is **keras.preprocessing.image.flow_from_directory**. Generator will be an input to fir the model.

- **Model Implementation**

  The selected pre-trained model for transfer learning is the Inception-V3. The steps in order to create a new model that classifies the label used are:

  a. Import the Inception-V3 removing the top layers, which are actually the layers that classify the image, these layers need to be replaced by ones that classifies the desired labels

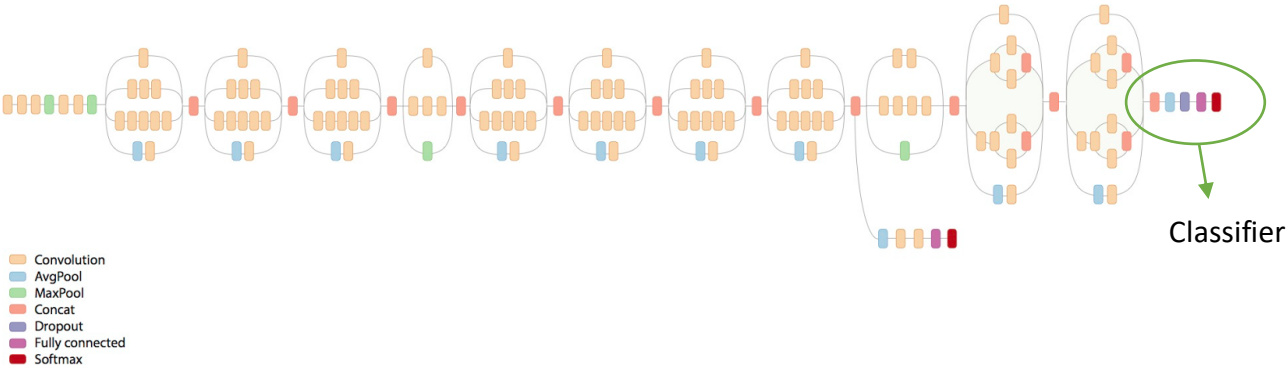

Figure 5: Inception-V3 model structure [(source)]

  b. Add new top layers in order to receive the output of the Inception-V3 Model, train the new layers and classify the images. Details about the structure and parameters of the new top layer are in section "Refinement".

  c. Select the number of layers that will be trainable. As the pre-trained model has learn from the ImageNet data set, lower layers learned to distinct features as texture, edges, contrast, among other, so training can be omitted for those layers, saving processing time, which has been a problem on this project, as explained in section "Algorithms and techniques".

  d. Compile the model, tuning the parameters **optimizer**, **loss** and **metrics**.

  e. Set the model check point in order to save the best model while training all the epochs. The method used is **keras.callbacks.ModelCheckpoint**[17].

  f. Fit the model, the method used is **fit_generator**[18], which receives as input the generators for train and validation data, plus parameters such as steps_per_epoch, class_weight, epochs, callbacks, etc.

[17] https://keras.io/callbacks/
[18] https://keras.io/models/sequential/

Refinement

To resume, on this project the researcher is building three models: Smiling or not, Gender and Young or not. The model structure will be the same for each model.

Initially, using the entire training dataset (162.770 images), the Inception-v3 removing the top layers and adding the structure below[19]:

a. Flatten Layer
b. Dense Layer with output arrays of shape 2048 and ReLU activation function
c. Dense Layer with output arrays of shape 2 and Softmax activation function (prediction)

Optimizer has been set as RMSprop.

Using this layer structure, the model was taking over 2 hours on finish an epoch on a MacBook Pro 2017 running on CPU (this limitation was explained in the "Algorithms and Techniques" section).

After three epochs and with risk of hardware malfunction due the stress to the machine as high temperature reached on the CPUs, the accuracy over the validation data was not higher than 70%.

As measure to improve the performance on timing and accuracy, the researcher got access to a windows machine with better characteristic and GPU support from TensorFlow. Also, the data set of images was reduced for each model to:

- Training: 20000 images
- Validation: 5000 images
- Test: 5000 Images

And the structure of the top layers of the model has been changes to:

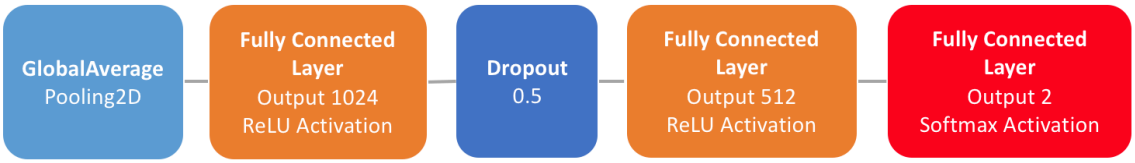

Figure 6: New top layers for custom models.

The first 52 layers of the Inception-V3 model has been set to not trainable and the optimizer has been set to Stochastic Gradient Descent with learning rate = 0.001 and momentum = 0.9.

Using the new set up the timing and performance of the model improved considerably.

[19] https://keras.io/layers/core/

4. Results

Model Evaluation and Validation

The test data set consist in 5000 images randomly selected from the recommended partition for testing. Using the model described in the previous section, the performance metrics for every model are:

Model	Accuracy	F1 Score
Transfer Learning – Smiling or Not	91.38%	0.9137
Transfer Learning – Gender	95.52%	0.9535
Transfer Learning – Young or Not	82.06%	0.8182

Table 2: Models performance metrics.

- **Transfer Learning – Smiling or Not**

  The model has a high accuracy and F1 Score predicting if the subject in the image is smiling or not, the selected parameters explained in the "Refinement" section have a high impact in the performance of the model. In order to test the robustness of the model, images outside of the CelabA and with different dimension have tested in the developed model; the results are very accurate and can be found in the "3. Predict" Jupyter Notebook and some examples can be found in the "Free-Form Visualization" section of this document.

- **Transfer Learning – Gender - Female or Male**

  The model has a high accuracy and F1 Score predicting if the subject in the image is female or male, the selected parameters explained in the "Refinement" section have a high impact in the performance of the model. In order to test the robustness of the model, images outside of the CelabA and with different dimension have tested in the developed model; the results are very accurate and can be found in the "3. Predict" Jupyter Notebook and some examples can be found in the "Free-Form Visualization" section of this document.

- **Transfer Learning – Young or Not**

  This is the model with lower accuracy of the three models (82.06 %). In principle, features that can make a subject look older than other can be difficult to find for an algorithm and the is also a subjective factor on how to label a person as young or older. Also, even for humans in some cases is difficult to know the age of a person based on appearances. Some cases available in the data set are:

Figure 7: Sample of celebrities labeled as "Not Young"

In Figure 7 all the celebrities are labeled as "Not Young". The researcher consider that the first image has more features of an older person, such as white hair, baldness and wrinkles. Meanwhile the other two images do not have these features, or at least there are not that string.

Nevertheless, considering the subjective of the attribute, the model is doing a good job reaching a 82% of accuracy over the test data set.

Justification

The selected benchmark is the Smile Detector based on Haar Cascade from OpenCV[2021], which will be compared against the developed model based on transfer learning for smile recognition using the same test images, the comparison will be using the accuracy performance metric.

During the testing the researcher noted that the Haar Cascade Smile Detector accuracy is very sensitive to the parameter **minNeighbors** in the **detectMultiScale** method. An example of the impact is:

Figure 8: Comparison of result of Haar Cascade Smile Detector using different minNeighbors parameter.

Figure 8 shows that changing the ***minNeighbors*** parameter in one integer, the prediction will detect (or not) a smile in the picture.

[20] https://opencv.org
[21] https://github.com/opencv/opencv/tree/master/data/haarcascades

In order to get the best result out of the Haar Cascade Smile Detector Model, the researcher has iterated over the **minNeighbors** parameter, and calculated the accuracy on each iteration, the best result will be the accuracy benchmark. All the steps are coded in the "2. Benchmark Comparison" Jupyter Notebook.

After the iteration process, the best result of for the Haar Cascade Model is an accuracy of **81.98%**. As conclusion, it can be said that OpenCV has a very good implementation for Face Detection, but this is not necessarily the case for Smile Detection, where the patrons to find a smile in an image using Haar Cascade can be difficult to find.

The developed model based on transfer learning is better than the benchmark. This solves the problem to predict accurately the selected attributes.

5. Conclusion

Free-Form Visualization

A good way to measure how good the developed models are, is testing them with never seen data and images with different dimensions at the ones found in the CelebA data set, which are 178x218. Below there are some examples of the results on never seen images, the Jupyter Notebook "3. Predict" has implemented a code that load the trained models and takes a picture from a specified directory and predict the selected attributes (Smiling or Not, Gender and Young or Not).

Some examples of pictures inside and outside of the CelebA Data set:

- In Celeba Data Set

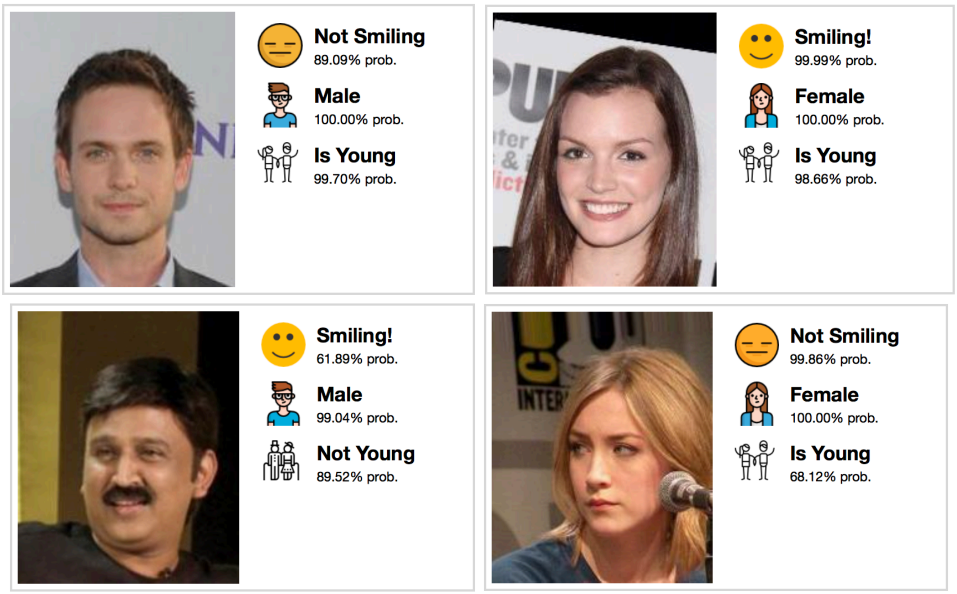

Figure 9: Plotting results of the three classifications models using Jupyter Notebook "3. Predict" over picture of the CelebA Data Set.

- Not in CelebA Data Set

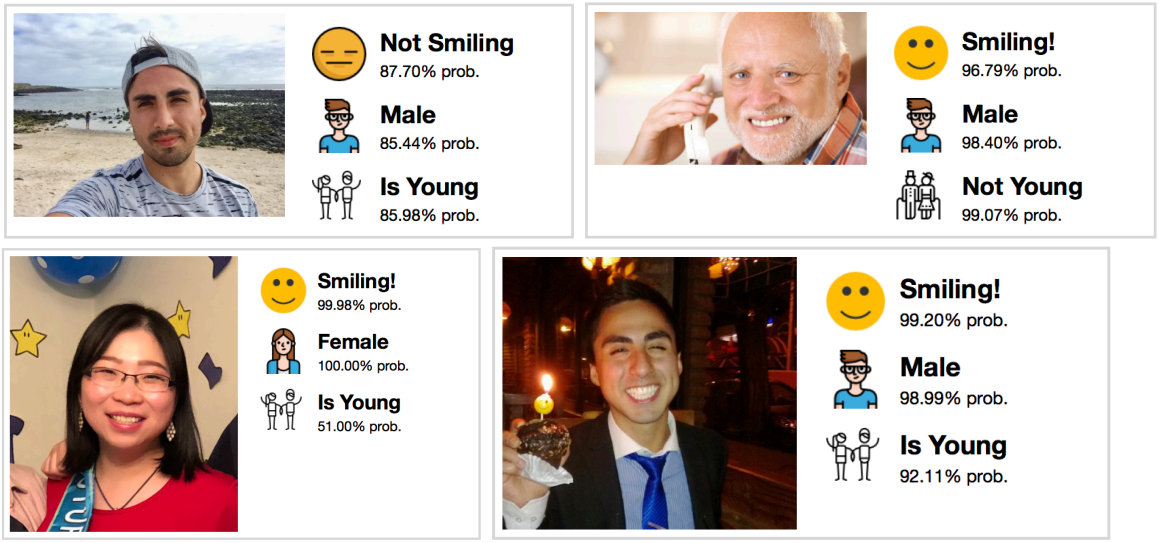

Figure 9: Plotting results of the three classifications models using Jupyter Notebook "3. Predict" over picture that are not part of the CelebA Data Set.

Reflection

The project can be summarized in the following steps:

a. Data Collection
b. Data Analysis
c. Data Preparation and Preprocessing
d. Load the base model (Inception-V3 in this case)
e. Create the New classification layers
f. Create Data Generators to fit the model
g. Train the Model
h. Evaluate the Model
i. Perform Corrections

The project was particularly interested as it involves build three models based on transfer learning and adapt them to predict three different attributes from images.

One of the most difficult part during the project was to deal with resources limitation regarding computational power, as TensorFlow no longer supports its models to run on GPU for MacOS, and running the models on CPU was not a reliable option due timing and risk to damage hardware. AWS could be a good option too, but thanks to the access to a good machine with TensorFlow GPU support to run the models, the researcher was able to train the classification models.

Other difficult part was to choose the final CNNs structure to run the new classifiers, there are many options that can lead to get certain accuracy. The selected structure for this project performs well to solve the specified problem, getting a good accuracy and robustness for never seen images.

Improvement

The main improvements that could be done for this project are:

- Train the algorithms with the entire data set of images. Due computational resource limitation, the model was train with a subset of images. Having an appropriate machine, the model can be trained including all the images. This will make the algorithm to learn from different context of the picture giving it more experience in order to predict better never seen images.

- Use difference structures for the CNNs. This approach could give better performance to the model, is an expensive task anyway, as the model can be measure on the test data set after is trained, and this takes time and computational resources.

- Watching the pictures of the CelebA Data Set, most of the pictures are almost a close-up to the face of the subject, this leads to the model to learn from this type of pictures, and in situation where the subjects is just a small portion of a picture, the model could not perform well. To deal with this, more sophisticated preprocessing data can be added or complement the data set with pictures that are not entirely based in close-up to the face of the subject.

- Environments where there are more than one subject in the picture was not part of the scope of this project, but it is a good improvement in order to develop a better application. OpenCV is a good candidate to help with this development, as it very accurate to detect feces and its position in the pictures, then that portion of the picture (the faces) can be classified separately using the developed models on this project.

- Integrate the solution with a user interface will be totally a plus for the development. In order to give more usability to the users, the solution can be integrated an API, and be consumed by Web and/or Mobile Applications.