Gender image categorization using neural networks

Angel Lopez Manriquez

Contents

1 Description of the problem **1**

2 Description of our approach **2**

2.1 Preprocessing . . . 2

2.2 Architecture of the model . . . 3

2.3 Validation . . . 3

3 Implementation **3**

4 Results **4**

5 Conclusions **4**

Abstract

In this document we present our approach to the problem of, given an image, detect human faces and classify them as male or female by using only neural networks, no deep neural layers were are allowed in this project, tough generalization could have been improved by doing so.

We used numpy, matploblib, tqdm, cv2 (computer vision library), keras and scikit learn libraries to solve the task.

1 Description of the problem

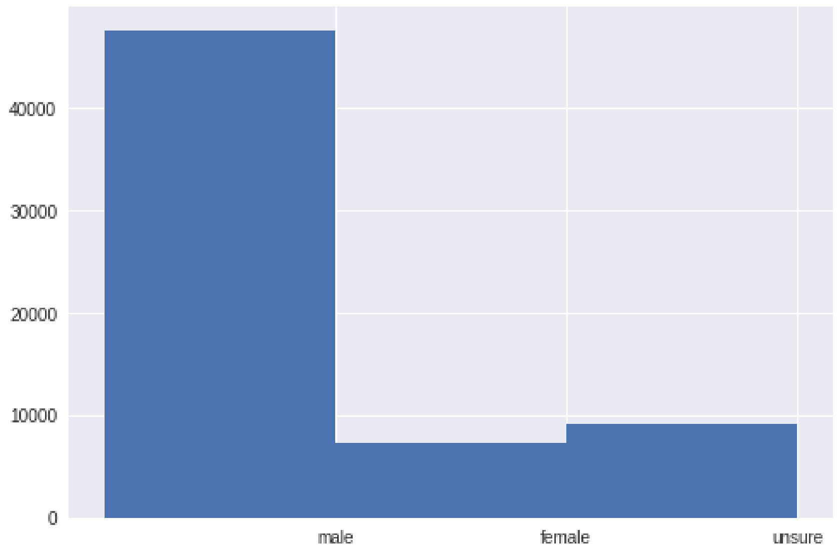
Object recognition is the task of automatically finding and identifying objects in an image Humans detect and recognize objects in images with extremes ease, even if objects vary in shape, size, location, color, texture, shine or are partially clogged.

The task we have to solve is the classification of the faces found in an image, introduced in [1] [2] [3] [4].

The dataset is available at: https://www.figure-eight.com/data-for-everyone/ named as *gender image categorization of images* which is only a csv file, containing the links for images and their corresponding label. There are more columns, but we only care for:

- `please_select_the_gender_of_the_person_in_the_picture` which can be *male, female or unsure*.
- `image_url`, which contains the url where the image is located. The link might be forbidden.

Figure 1: Number of instances



As we can appreciate in the histogram, there many many instances of male than female or unsure. We try to train the model with the same number of samples.

2 Description of our approach

We organized the implementation of the project according to the tasks:

1. Preprocessing the dataset
   (a) Get images tagged as *male, female*.
   (b) Balance the amount of samples for each class.
   (c) Detect the faces in the image in order to reduce the dimensionality of the problem, obtaining only what we care about, the faces
2. Define the architecture of the model Since this problem needs to be fast (for example, face detection when taking a selfie) we defined a simple architecture. The layers are
   (a) 96×96 nodes. Input layer, we flat the image an pass it.
   (b) 128 nodes. Fully connected layer, with the rectifed linear unit function ($relu(x) = \max(0, x)$)
   (c) 2 nodes (since we have two classes) with the softmax function.
3. Train the model
4. Evaluate the model, modify the hyper-parameters and repeat until obtain suitable results.

2.1 Preprocessing

First, we drop all rows which are tagged as ”unsure”, they are irrelevant, it may be an animal, a flower of an text image.

Next, something we desire to train the model with almost the same amount of images per class, we look if we have at least n images, if we have them, we just load them into an `numpy array`, else we download at least n samples per class into an load them.

We loop over each image that we have downloaded, it may occur that:

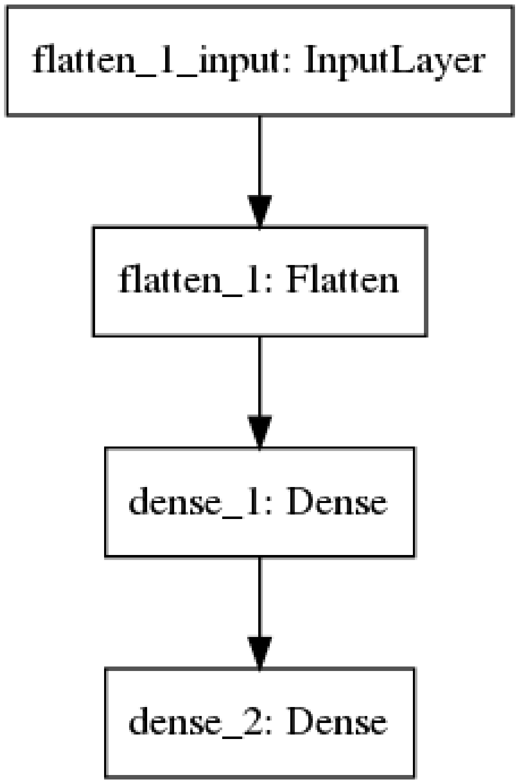
1. The image, in fact, does not contain a person.

2. We could not extract a face from the image.
3. The quality of the face is poor, implying we could not resize the image.

For any of the cases described above, we dropped the rows, again. Finally, we extract only one face per image for training the model.

2.2 Architecture of the model

We define the model with few layers and few neurons to increase speed. Of course, we do not want to lose accuracy, as we have discussed in lectures, there is no perfect way to decide the architecture for a model. We proposed this model, and it gave good results.



Some of the hyperparameters we must define are the input shape and the number of outputs [1]

2.3 Validation

To validate our results we compute the cross-validation in the complete dataset.

As an additional validation step we computed the confusion matrices for the classifiers.

3 Implementation

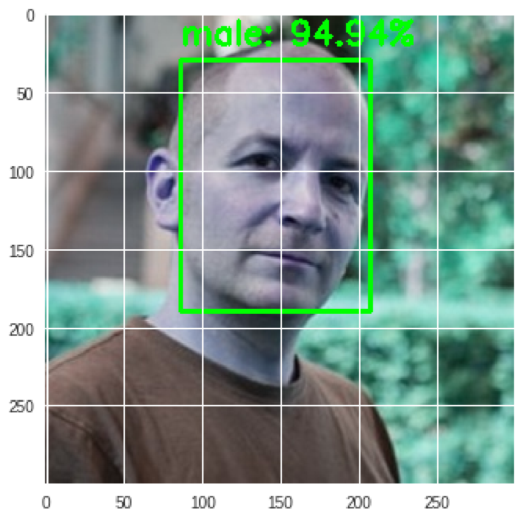
All the project steps were implemented in Python. We used the `urlopen` function from the urllib module to download the image. For reading the dataset we used pandas, then select the relevant rows into a list of numpy arrays, to preprocess the data we did it with the face detection from the `cvlib` library. We use keras to define the architecture and training the model.

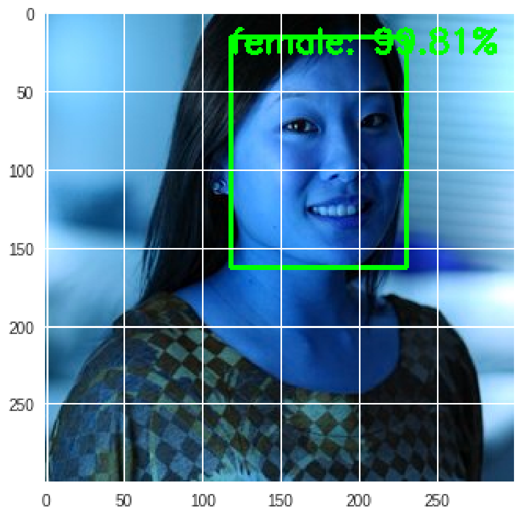
We illustrate how the implementation works in the Python notebook Gender image categorization using neural networks

[1]keras

4 Results

The accuracies are shown in the notebook. By increasing the amount of samples for training the model, the number of epochs we obtain an accurracy about 75% .





5 Conclusions

It's very impressive how mathematics help us in our job, by defining a graph with weights associated for each one, and the number of possibilites we have to improve the results.

Overall, I think the accuracy of the models is decent but can be improved further by using more data, data augmentation and better network architectures and deep neural networks.

One can also try to use a regression model instead of classification for Age Prediction if enough data is available.

References

[1] Krutika Bapat Satya Mallick. Shape matching using hu moments. https://www.learnopencv.com/shape-matching-using-hu-moments-c-python/, 2007.

[2] Irfan Essan Dr. Aaron Bobick. Hu moments. https://classroom.udacity.com/courses/ud810/lessons/3513198914/concepts/34988000950923.

[3] Gary Bradski and Adrian Kaehler. Learning opencv. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):141–144, 2003.

[4] Vikas Gupta. Gender and age classification using opencv deep learning (c++/python). https://www.learnopencv.com/age-gender-classification-using-opencv-deep-learning-c-python/, 2015.