Age, Gender and Ethnicity prediction from face detection using Convolutional Neural Network

Gianluca Maselli, *maselli.1892187@studenti.uniroma1.it*

***Abstract*—This paper present and automatic approach for face detection and face classification. By using the UTKFace dataset a CNN is trained in a multi-label classification task in order to predict the age, gender and ethnicity of a given face in input. After training the network we perform the face detection task by using OpenCV on images external to the dataset and taken from the web. The detected faces are then used as input for the pre-trained network which performs the age regression task and gender, ethnicity classifications. By contrast, the same experiment is repeated by substituting the age regression task with an age classification task considering age ranges as classes.**
At the end we show the differences of both the approaches in terms of predictions on external samples as well in order to profile unknown subjects in the best way possible.

I. Introduction

Nowadays, face information of human has been widely explored. As a matter of fact many related applications have also been well developed by companies and government agencies. Among them, estimate the age through a regression task and gender, ethnicity classifications has increasingly gained attention becoming one of the top surveyed topics in the academic field of computer vision. The latter is fundamental when dealing to various kind of systems such as surveillance and authorization systems in several public places.
Age, gender and ethnicity estimation from a detected face can be a crucial aspect in cases where human operators are not fast enough to intervene and to prevent crimes carried out by non profiled people. In fact, by capturing and processing biometric data used by system in subsequent authentication operations is possible to see if the subject has been already enrolled in the system or not. If not, it is possible to create a new gallery of templates regarding the new subject.
The idea for this application is then to try to estimate the age, gender and ethnicity from extracted faces coming from images of group of people by using a deep learning approach. In the past few years deep learning has attracted tremendous attention from researchers in many fields and in particular in computer vision, indeed by relying on Convolutional Neural Networks (CNN), fast and accurate results can be achieved when facing image classification problems.
Three major problems has been faced, the first one was to find a suitable dataset in order to train the network and, the second was to build a balanced model constituted by three independent classifiers, while, the last one was to rely on the Google Colab platform for the experiments. Indeed using this platform implies a limited amount of RAM and hard disk memories. The latter is one of the main drawback since we are not allowed to use a very big dataset to train our network due to the fact that by loading larger dataset could result into a crash. To solve the first and the third problem, the only choice, after exploring several possibilities, was the UTKFace dataset [1] consisting of over 20,000 face images with annotations of age, gender, and ethnicity. Despite the fact that the dataset is not very large, inside some bad samples were found. After deleting those ones, by using PIL and Pandas Libraries, a csv file was created containing for each line a dictionary regarding the age, gender, ethnicity, image name and pixels of each cropped face contained in the dataset. Even if at the first glance this could seem to be useless, at the moment of the dataset loading in Google Colab platform, this strategy improves the loading speed and does not cause a timeout of the platform.
From the time being, in the following sections, with ”dataset” we will indicate the one written in the csv file.
Before entering in the experiments part, some important theoretical outlines will be explained aiming to give an overall idea of what we have implemented. Later on, the experiments will be explained step by step along with the libraries, methods and model adopted. Finally the result will be shown and commented by also underlying where the model could be improved by introducing some adjustments.

II. Background

In this section, theoretical outlines of what has been used will be introduced. This section is fundamental to understand the considerations on which the experiments have been performed.

A. Face Detection

Largely implies in various field, face detection is used to find and identify human faces in digital images. Nowadays, it plays an important role as the first step in many key applications including face tracking, face analysis and facial recognition. Furthermore, face detection has a significant effect on how sequential operations will perform in the application. In particular in face analysis, face detection helps to identify which parts of an image should be focused on to determine age, gender, ethnicity and also emotions in cases where facial expressions are considered as well.
Basically, to find human faces from different images where other non-face object are present (e.g, animals, landscapes, inanimate objects and so on) face detection application use machine learning algorithms which typically start by searching

[1] https://susanqq.github.io/UTKFace/

for human eyes, since they are the easiest feature to detect, and then it attempts to detect eyebrows, the mouth, nose, nostrils and the iris. At the end the algorithm is able to detect the entire facial region, moreover, to confirm this assumption, it perform additional tests.
In general, detecting face in picture is not an easy task since a variability of factors can be encountered. Among them we can consider the face pose, expression, position, orientation, skin color, the presence of glasses of facial hair, the differences in camera gain, lighting conditions and image resolution. To improve the quality of face detection, in recent years, studies have demonstrated that by applying a deep learning methods can significantly outperforming traditional computer vision methods.

B. Convolutional Neural Network

Widely adopted in computer vision, face recognition, image classification etc., the convolutional neural network (CNN) is a specialized type of neural network model designed not only to work with two-dimensional image data, but also with one-dimensional and three-dimensional data.
The name of this network comes from the fact that it uses a specific convolutional layers which performs an operation called convolution.
Considering an input, a convolution is the application of a filter to the latter that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image. Mathematically speaking, the convolution is a linear operation involving the multiplication (dot product) between a set of weights and the input. Since this operation was designed for two-dimensional input, the multiplication is performed between an array of input data and two-dimensional array of weights(called filters or a kernel).The filter (smaller that the input) is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom. After applying the filter multiple times to the input array what is obtained is a two-dimensional array of output values which represent a filtering of the input. This two dimensional out array is the so called feature map as shown in Fig 1.
After that the feature map is created each value that it

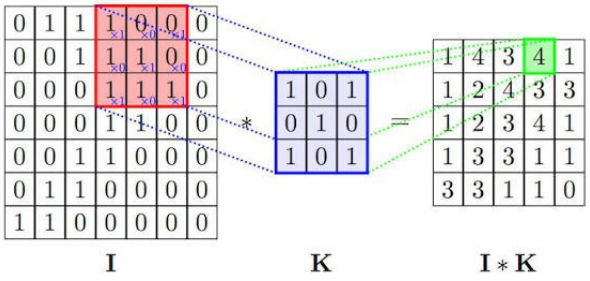

Fig. 1. Example of the convolution operation where I is the input and K is the filter(or kernel) applied. The result of this operation is the Feature Map

contains is passed through a nonlinear function(e.g ReLU).

The innovation of using the convolution operation in a neural network is that the values of the filter are weights to be learned during the training of the network. This is such a powerful idea since the network will learn what types of features to extract from the input. Specifically, training the network under stochastic gradient descent leads it to learn to extract features from the image that minimize the loss (in our case we have three of them, one for each classifier) for the specific task that the network is being trained to solve [1]. In our case we would like then to extract features that are the most useful to classifying faces.

A part from the convolutional layers, many other hidden layers are present in ConvNets. An example could be pooling layers. They are used to reduce the spatial size of the convolved feature. Throught dimensionality reduction the computational power required to process data is decreased. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.
Often fully-connected layers are added to perform a classification of features after flatting them into a single 'Flatten' layer (the feature map coming from the last convulutional layer is flatten into a column vector). Adding a fully-connected layer is usually a cheap way of learning non-linear combinations of high-level features as represented by the output of the convolutional layer. In fact, the fully-connected layer learns a possibly non-linear function in that space. An example of a standard Convolutional Neural Network is reported in Fig 2.

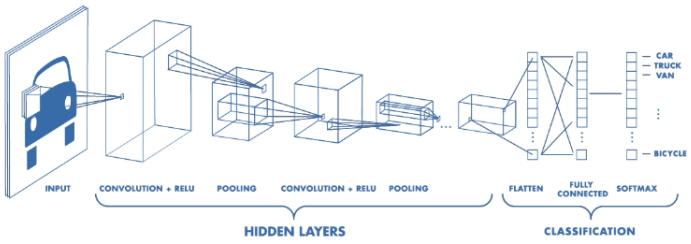

Fig. 2. Example of a classical convolutional neural network considering also pooling layers, ReLU activation functions and flatten layer.

III. Proposed Model

In this research for each image the model must be able to predict three different outcomes which are age, gender and ethnicity respectively. This problem can be solved mainly in two different ways:

1) build three different CNNs and then train them separately to estimate age,gender and ethnicity individually. Each network perform a single label classification for the detected face given as an input.
2) build a single CNN which trained to predict three different outcomes at the same time. Broadly speaking, this is a problem in which zero or more labels are required as output for each input sample, and the outputs

are required simultaneously. The assumption is that the output labels are a function of the inputs. Basically, for each given detected face as input we would like to predict age,gender and ethnicity at the same time.

In many situations requiring a person profiling, the model should be able to be as fast and accurate as possible. For this and other reasons, the chosen approach was the second one.

Even if the model could be intuitive, many problems has been encountered. Fist of all, having a model which performs two classifications and one regression task (or three classifications in the case of classify samples in age ranges classes) at the same time means having three different activation functions in the last layers. This result in three different loss functions. Basically, we must assure that the model does not overfits due to the fact that one classifier could be faster at learning with respect to the others. On this assumption, even if the three classifiers shares the same base, they must be balanced with respect to the others. The solution is then to use more or less the same number of layers with the same amount of filters for each classifier in order not to create an unbalanced architecture. Despite on the architecture, the curse of overfitting is always present. Therefore to avoid this well-known problem, dropout layers are inserted in the architecture. Substantially, the latter is a regularization technique which implying that some number of layer outputs are randomly ignored (”dropped out”). This has the effect of making the layer to be treated like a layer with a different number of nodes and connectivity to the prior layer. In this way, this techniques makes the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs.
In Fig 3 the model is shown. In the experiment section the choices made will be further explained giving the motivation behind them as well.

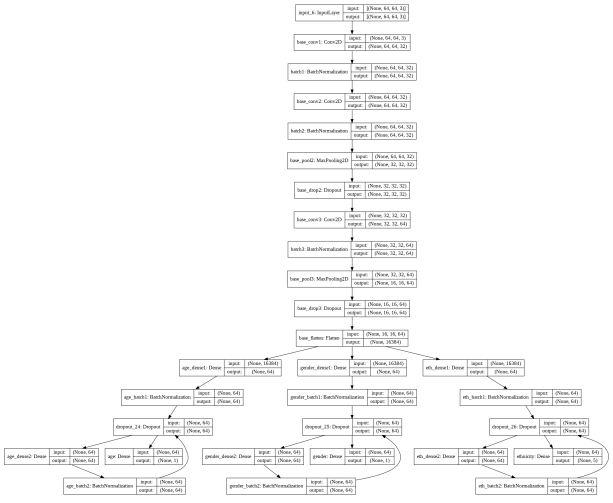

Fig. 3. The full proposed model of the pre-trained network used for age,gender and ethnicity classification of the detected faces.

IV. Experimentation

After explained all the theoretical outlines on which the proposed model has been built, the most important steps leading to our results along with the reasons behind the choices made are introduced.

A. Development platform

Experiments are performed using Google Colab. This is a straightforward solution when someone does not have the sufficient computational power on its local device. Google Colab allows the use of a GPU for a limited amount of time. This can be very useful when training a Convolutional Neural Network by increasing the training speed a lot with respect a normal training without the use of GPU.
The major disadvantages of using Google Colab is that, since we mount our personal Google Drive, it is not able to load huge amount of data in its local memory disk. In fact when a large dataset is loaded, it can cause a runtime error leading to a crash of the system. At this point the program stops and restarts.
In facial classification, using Google Colab could be a problem due to the fact that the network must be trained on a large dataset. Despite on this, we tried to solve this problem with other solutions we are going to explain in the following sections.

B. Libraries

The application has been developed by using six main libraries:

1) TensorFlow: Even if machine learning models is a difficult discipline, the related algorithms are make easier to implement by relying on this open source library. Thanks to this machine learning framework, it is easier to acquire data, train models, serve predictions and refine future results. It is also easy to use since it uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++. For our purposes this is a good library since allows to train and run deep neural networks to solve different tasks. The used version in the application is the 2.4.2.
2) Keras: Powerful and easy-to-use free open source Python library to develop and evaluate deep learning models. Basically this library wraps the efficient computation TensorFlow library and allows to define and train neural network models in few lines of codes. All the function used, along with the layers to build the CNN, comes from this library.
3) OpenCV: Widely adopted in computer vision, machine learning and image processing, it is a huge open-source library that allows to process images and videos to identify objects, faces, or even handwriting of a human. Python is capable of processing OpenCV array structure for analysis when this library is integrated with other libraries as Numpy. Image patterns and its various features are indetified by using a vector space and perform mathematical operations on these features.

4) Numpy: It is a Python library useful when dealing with multidimensional array objects, various derived objects(matrices and masked arrays) and an assortment of routines for fast operations on arrays(mathematical, logical, shape manipulation, sorting, and much more).
5) Pandas: Another open-source library built on top of Numpy package. Widely used in machine learnin tasks, it provides support for multi-dimensional arrays. In particular, it offers data structures and operations for manipulating numerical tables and time series. In the application, it is used to storage the labels of each image along with its pixels (coming from the csv previously written containing the images of the UTKFace dataset with relative labels) in a Dataframe used to train the Model. The same reasoning is applied on the face detection performed on images outside the dataset, which pixels are stored in another Dataframe and then extracted and reshaped to be given as an input for the age, gender and ethnicity predictions performed by the previously trained model.
6) Matplotlib: It is a visualization library in Python for 2D plots of arrays. It is built on Numpy arrays and designed to work with the broader SciPy stack. One of the greatest benefits of this visualization library is that it allows visual access to huge amounts of data in easily digestible visuals. In the experiments is mainly used to plot distribution of samples classes and to plot losses trend with respect to the training epochs. With the use of this library the images are also displayed.

C. Dataset

As mentioned in Section I the Dataset chosen was the UTKFace one. To deal with the runtime error when loading a large amount of data the strategy chosen is to re-write the dataset in a csv file. To do so, a file python has been used. Basically, each $200 \times 200 \times 3$ image representing a cropped face, in which the labels are embedded in the file name as [age]_[gender]_[race]_[date&time].jpg, is opened and resized to a $64 \times 64 \times 3$ image by using the PIL library (not mentioned in the previous section since it is used just for this back-end operation). At this point this multidimensional array is flattened by the $ravel()$ function, transformed into a list of strings from which the two square brackets are removed. What remains is just a sequence of pixels on a single line. Then, the image name is split in such a way to extract the age,gender and ethnicity embedded in the file name and store them with the image pixels in a dictionary. Of course we build a dictionary for each image, that is further appended to a list. At the end when each image has been processed, the list of dictionaries is passed to a dataframe that organizes a table in which each line represent all the information related to each single image. Finally the dataframe is stored in a csv file using panda function $to\_csv()$.
This part is not contained in the main program, there, the csv file is directly used and loaded into another dataframe (Fig 4) which is used to perform each image pre-processing and other operations. An important clarification is that the gender and ethnicity labels are represented as integers values which is a good choice when dealing with neural networks. Here we have:

- gender: 0 for male and 1 for female.
- ethnicity: 0 for White people, 1 for Black people, 2 for Asian people, 3 for Indian people and 4 for Hispanic people.

Age labels are kept in their numerical form in the first approach, this is why in one of the classifier we perform a regression task by using a ReLU activation function. What we would like to have as output is then a numeric value indicating the estimated age for a given detected face.

On the other hand, in the second approach, where three classifications are performed each age is consider inside a range, the available ranges are:

- 6-22: these range classify people in their puberty and adolescence, this range is represented by the integer 0.
- 23-39: these range classify people in their early adulthood, this range is represented by the integer 1.
- 40-59: these range classify people in their second adulthood, this range is represented by the integer 2.
- 60+: these range classify people in their third, fourth and fifth adulthood, this range is represented by the integer 3.

The choice of just 4 age ranges is due to the size of the dataset. Since it is not so large a much more level of detail could result in problems during the classification tasks because each class could contain too few samples to be correctly classified. In the Section IV-I the comparison between the approach using a regression task to estimate age with the one using an age range classification will be shown.

	age	gender	ethnicity	img_name	pixels
0	85	1	0	20170110182945927.jpg.chip.jpg	207, 180, 150, 196, 169, 139, 202, 173, 143, 2...
1	8	0	2	20170117192413109.jpg.chip.jpg	24, 24, 26, 18, 22, 23, 17, 21, 22, 19, 19, 21...
2	76	0	0	20170111221823004.jpg.chip.jpg	244, 248, 249, 244, 248, 249, 244, 248, 251, 2...
3	85	0	0	20170111210319130.jpg.chip.jpg	167, 149, 129, 171, 163, 144, 211, 212, 196, 2...
4	80	0	0	20170117173628559.jpg.chip.jpg	186, 168, 146, 183, 167, 142, 181, 169, 143, 1...

Fig. 4. The csv file is loaded in a dataframe used then to perform operations on images. If the approach used is the one with three classification tasks, instead of the each single numerical age the integer representing the age range is present.

D. Dataset analysis

After loading the csv file in the aforementioned dataframe we can analyse some aspect regarding the distribution of samples among classes and consequently the presence of outliers in the dataset. The first thing to do is to extract the information from the dataframe and to re-obtain each image with its own labels. This is easy done by a function which takes each line of the dataframe and retrieve the labels and the related $64 \times 64 \times 3$ image which pixels are then normalized between 0 and 1. Normalizing pixels between this interval is a good practice, in fact neural networks process inputs using small weight values, and inputs with large integer values can disrupt or slow down the learning process. In Fig 5 an

example of images contained in the dataset are shown along with the respective labels converted in string (the labels have their string values just for the visualization, the integers values are kept when fed into the CNN).

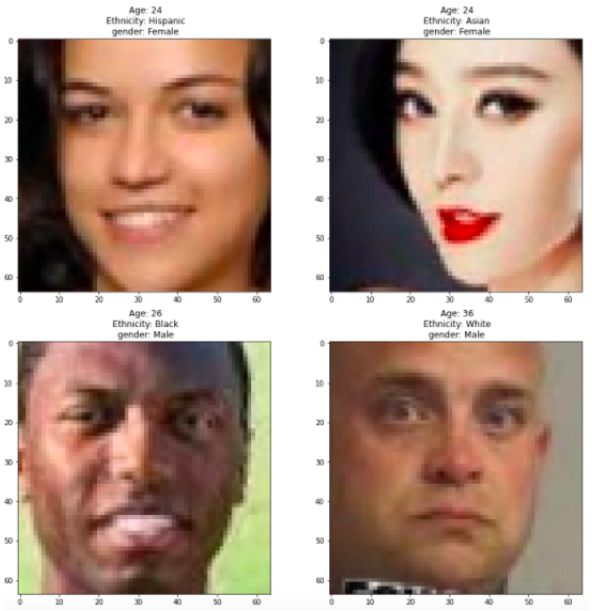

Fig. 5. Some samples of the dataset with their related labels

Before defying the train, validation and test sets on which our model will be evaluated, we need to perform some modification to the dataset. First of all we eliminate the ouliers images regarding ages less than 5 years old and greater than 90, this is done essentially because these classes contain few images which could badly affect the prediction of the model. Then, to analyze better the dataset, all the images, and each label is considered separately and stored in an array. In this way we can better analyze the distribution of samples for each class. From Fig 6 and Fig 8 we can notice and highly unbalanced dataset for both ages and ethnicity groups. In fact we can notice that the majority of samples are the one of the 26 years old people in the first case, while, in the second case we can notice that white people are by far more than the other ethnic groups. By contrast, in Fig 7 we can observe that the distribution of males and females is quite balanced.

An unbalance dataset is in any case present even if the model performing three classification tasks is considered. Here, in the image pre-processing the ages are grouped in ranges as introduced in Section IV-C. As in the case of the model using the first approach, in Fig 9 we can notice that the ages range are still unbalanced.

Since the problem to solve can be traced back to a supervised learning one we need to associate to each input X sample (image) its related y output labels (age,gender and ethnicity in this case). As we have a multi-label classification for each input X we have then 3 outcomes, y1, y2 and y3 respectively. Taking the latter consideration into account, a good way to

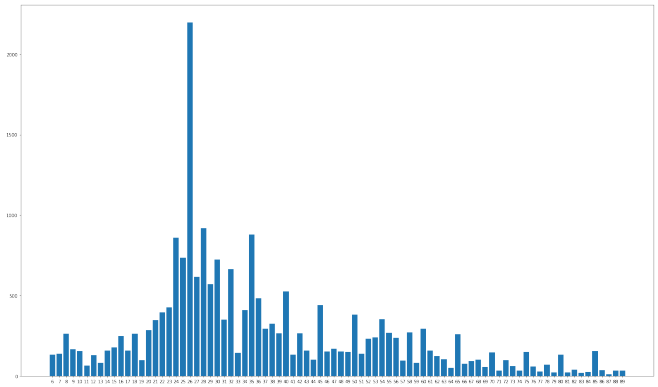

Fig. 6. Distribution of samples for each age

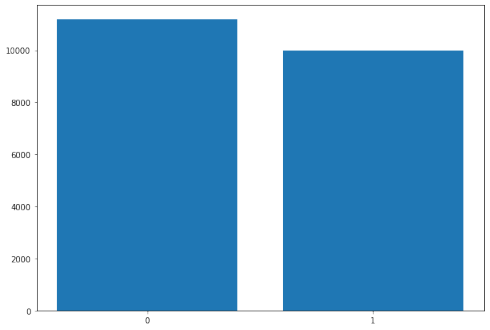

Fig. 7. Distribution of samples among male and female classes

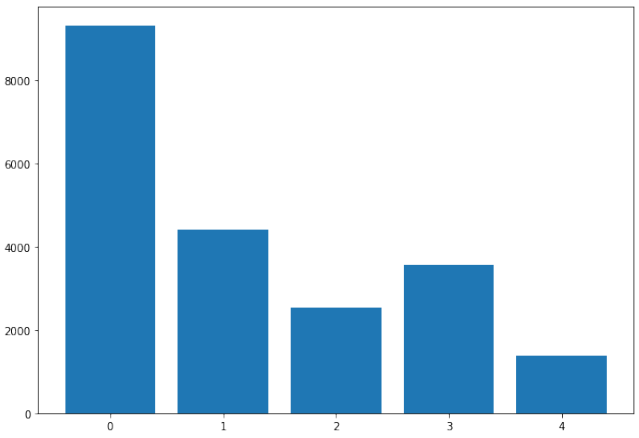

Fig. 8. Distribution of samples for each ethnic group

face this challenge is to create an array of images pixels and a matrix of labels. Basically this means that for a given index i in the images array, it corresponds to the same row index of the matrix containing 3 columns representing age, gender and ethnicity of the current image. In other words, suppose we would like to look at the first image in the array and also we would like to know its related labels. What to do is then to select $images[0]$ and $matrix[0]$, the result will be the image pixels at the first index related to the first row of the matrix containing the three labels for that image. Furthermore we can access to the individual value for each row, for example if we want to know only the age (or the integer of the corresponding age rage) we can type $matrix[0, 0]$, for the

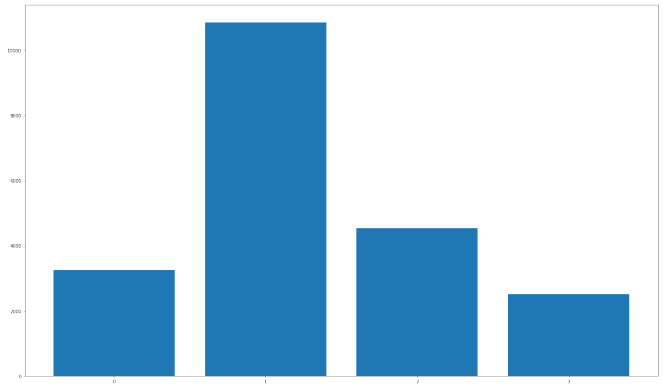

Fig. 9. Distribution of samples for age ranges defined for the age classification approach. On the x axis the integer indicating each age range is inserted

gender $matrix[0,1]$ and for the ethnicity $matrix[0,2]$. This two arrays will be then used for the train,validation and test splits.

E. Train,Validation, Test splits and Data Augmentation

Once we have our arrays of images and labels, a 80:20 split is adopted to define the test set as well. This is easily done by relying on the $train\_test\_split()$ function by passing the inputs (images array) and the related outputs (labels matrix). To help the CNN to train better, the shuffle of the samples is considered as well.

Before training the network a solution for the unbalanced training dataset must be found in order to better train the network and improve the final predictions. A well-known solution in these cases is the data augmentation technique. It can be used to artificially increase the size of the training set by adding slightly modified copies from existing data. Data Augmentation is a good practice to prevent overfitting or when the initial dataset is too small to train on and in general to obtain a better model performance. In this case, the data augmentation is performed not only to increase the number of samples but to try to balance the dataset a little bit.
However, this technique has not to be used to increase too much the dataset. As a matter of fact, by increasing too much the training set this could end up having a model that overfits because it starts to get used to the sample in the training set, performing good predictions on them but incorrect predictions on external samples. Therefore to make the training set more balanced we computed the average of samples for each class and we perform data augmentation only on the classes that does not exceed this average. In this way we increase slightly the training set without adding too much samples that could lead to an overfit. To perform data augmentation on these samples the best strategy is to select randomly a data augmentation technique (between flip, distortion and shifting [2]) and to apply it to the sample.

[2]https://medium.com/@schatty/image-augmentation-in-numpy-the-spell-is-simple-but-quite-unbreakable-e1af57bb50fd

While performing data augmentation on selected images, we must keep the related labels. To do so we then create another duality of arrays where in the fist the augmented images are stored and in the second (matrix) the related labels are contained in a similar fashion of the methodology already explained in Section IV-D. At this point we can simply create a total training set containing both the existing data and the augmented ones, with, of course their labels as well.
Finally, with the $train\_test\_split()$ the training set is further dived to obtain the validation test.

F. Model

In Section III the proposed model has already been explained. However in this section a clarification on the choices regarding the number of parameters used has to be further emphasized.
As mentioned before the face classification could be faced using two type of models.

1) the first one using two classifications on gender and ethnicity and 1 regression task to estimate the age.
2) the second using three classifications. Therefore, instead of estimating the age by a regression task, classify a face based on a age range which is considered as a class.

In both of cases the models accept as input $64 \times 64 \times 3$ images. The features extraction is performed by 3 convolutional layers with 32, 32 and 64 filters respectively, and 3×3 as dimension for the kernel sizes of each layer. After any convolutional layer a batch normalization layer is also inserted to stabilize the learning process and dramatically reducing the number of training epochs required to train the network. After the second and third convolutional layers, max pooling and dropout layers (both explained in Section II-B and Section III) are present as well. The resulting feature map obtained after the last convolution is then flatten in a 1-dimensional array and used for the classification. The latter is performed by three different classifiers with the same number of fully-connected layers containing the same number of filters but with a slight difference between the first and the second model:

1) in the first case of the model adopting two classifications and one regression task, after the flatten layer two dense layers with 64 filters followed by one dense layer with 1 filter in the case of age regression task, 1 filter for gender classification and 5 filters for the ethnicity classification are inserted
2) in the second case where the model perform three classifications, two dense layers with 64 filters followed by one dense layer with 4 filter in the case of age range classification, 1 filter for gender classification and 5 filters for the ethnicity classification are considered. Therefore the only difference with the first model is that in the age estimation the last dense layer performs a classification on the 4 classes representing the age ranges.

The dense layers (a part from the final one) are followed by dropout layers to further avoid the curse of the overfitting. Another important aspect is that the activation function used in each layer of the CNN is the ReLU one (a part from the one

used in the last dense one). In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input. The ReLU (rectified linear activation function) activation is the default activation function for many types of neural networks because it makes the model easier to train and often achieves better performance. Generally speaking the ReLU activation function is a piecewise linear function that outputs the input directly if it is positive, otherwise it outputs zero.

Regarding the last dense layers used for the final classifications (gender, ethnicity) and for the final regression task (age) a different activation functions have been used:

- age: for the classifier aiming to do the age regression task of a detected face, we still use a ReLU activation function, this is because a regression task has to be performed and therefore as an output a numeric value is required.
- gender: here a binary classification is used and, therefore, a prediction of a probability between 0 and 1 is needed (0 for males, 1 for females). In these cases the best choice is to use a Sigmoid activation function since it exists between (0 to 1). The choice of the average value 0.5 classify as males all the probabilities lower than this value, and as a females probabilities greater than it.
- ethnicity: a Softmax function is used in this classifier. This is the straightforward approach when dealing with classification problems that involve predicting a class label for a given input. This activation function converts then a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. Therefore this classifier will retrieve 5 probabilities (one for each ethnic group) for a given image.

By contrast, in the model performing three classifications, in the last dense layer of the age classifier another Softmax activation function instead of the ReLU is used.

G. Loss Functions

Since the model has to face with a multi-label classification which requires three outputs, the same amount of loss functions is needed. Despite this particular case, the choice of a loss function is always fundamental since neural networks are trained using stochastic gradient descent optimization algorithm and weights are updated using the backpropagation of error algorithm. Therefore, a given set of weights is used to make predictions and the error for those prediction is calculated. The idea behind the gradient descent algorithm is that it seeks to change the weights so that the next evaluation reduces the error. The computation of the error is then performed by the loss function, and the idea is to minimize it as much as possible at each step of the training phase.

However this is not a standard case, but more than one loss function is required since we have three errors which are generated at each step: One for the age estimation (or age range classification), one for the gender prediction and one for the ethnicity classification. To computed these errors the chosen loss functions are the following:

- MSE (or sparse categorical crossentropy for age range classification): the mean squared error loss function is used when facing a regression task. The loss is the mean overseen data of the squared differences between true and predicted values. MSE is sensible to outliers, that is why their are eliminated in the image pre-processing (Section IV-D). The advantage of using this function is that it tends to penalize large error with respect small ones. To balance more the training of the model the error obtained from this loss is further divide by 100. This choice is made to stabilize all the losses between themselves and improve the training performance.
  In the case of the age range classification the sparse categorical crossentropy loss function used is explained below.
- binary crossentropy: it is intended for use with binary classification where the target values are in the set 0, 1. This loss function calculates a score that summarizes the average difference between the actual and predicted probability distributions for predicting class 1. The score is minimized and a perfect cross-entropy value is 0.
- sparse categorical crossentropy: this crossentropy loss function is mainly used when there are two or more label classes. This is because one example can be considered to belong to a specific category with probability 1, and to other categories with probability 0.

The loss function is a fundamental estimator during the training phase. By analyzing the three loss function trends it is possible to understand if the model is improving or not. Furthermore it gives an overall vision if the model is overfitting or is underfitting. The trend of this functions is also influenced by tuning the hyperparameters of the network such as learning rate and batch size.

H. Training

The training phase is modelled by inserting EarlyStopping, ReduceLROnPlateau and ModelCheckpoint functions.
With EarlyStopping we would like to monitor the validation loss since it is more reliable than the training loss to understand if the model is overfitting. On this view EarlyStopping function is defined in such a way that that the training is stopped if the validation loss stops decreasing after a patience indicating the number of epochs with no improvement. To quantify a minimum improvement the min delta is also inserted. Most important we always restore the model weights from the epoch with the best value of the monitored quantity (validation loss in our case).
ReduceLROnPlateau is a really useful function since it is possible to reduce slightly the learning rate as soon as the validation loss has stopped improving. Still, the reduction is performed after a certain patience and again, a min delta is needed for ensuring the new optimum and to only focus on significant changes.
Finally, ModelCheckpoint is used to save the model which is considered the best according to the validation loss minimum

value.
On import aspect to underline is that usually the tuning of hyperparameters as batch size and learning rate can dramatically change the way on how the model learns. The general rule to set this parameters states “bigger batch size bigger learning rate”. Often, small batch result generally in rapid learning but a volatile learning process with higher variance in the classification accuracy. By contrast, larger batch sizes slow down the learning process but the final states result in a convergence to a more table model exemplified by lower variance in classification accuracy. For this reason the chosen batch size is 128 with a learning rate of 0.001.

In Fig 10 and Fig 11 the model losses during the training phase of the first model performing two classifications and one regression task are shown. Contrariwise, in Fig 12 and Fig 13, the plots of losses regarding the model adopting three classifications are displayed.

From the plots of both the models, it is possible to notice that sometimes fluctuations are present especially at the beginning of the training, but after a while the trends start to stabilize. If the model is trained for too much epochs usually the gender loss does not improve any more causing an overfitting leading to bad predictions. That is because the model is faster to learn the difference between males and females with respect the age estimation (or classification) and ethnicity classification. Therefore, as mentioned before, a too long training must be avoided or one classifiers could effect the others leading to overfitting.

In any case from the plots of both the models, even if the model performing three classification tasks has an overall loss less than the the model performing two classification tasks and one regression task, the former has a much more problems at predicting the correct age range as notable from the accuracy which is not as high as expected.
By contrast, in the case of the regression task it is notable how the mae (mean absolute error) reaches incredibly good results at age estimation in the training and validation phases.

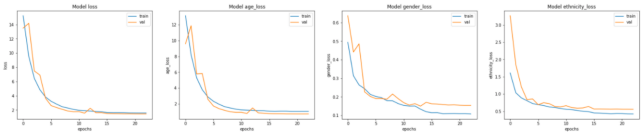

Fig. 10. Model with two classification and one regression task. Starting on the left, the overall model loss, the age loss, the gender loss and the model ethnicity loss trends are shown

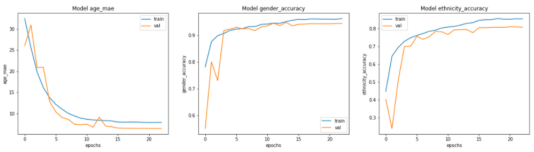

Fig. 11. Model with two classification and one regression task. Starting on the left, the age mean average error, the gender accuracy and the ethnicity accuracy trends are shown

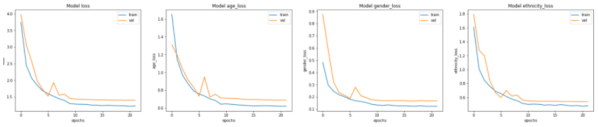

Fig. 12. Model adopting three classifications. Starting on the left, the overall model loss, the age loss, the gender loss and the model ethnicity loss trends are shown

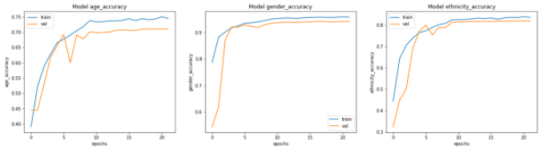

Fig. 13. Model adopting three classifications. Starting on the left, the age range accuracy, the gender accuracy and the ethnicity accuracy trends are shown

After the training phase, both the pre-trained models are possible to use since they are saved in an external folder. In this way by loading them, they are available any time a prediction on a detected face has to be made.

I. Model performance and results

After training both the models, their performances evaluation is performed on the test set. The results obtained, are shown in Table I[3], for the model with two classifications and one regression task, and Table II, for the second model performing three classification tasks.

Even if the age range accuracy is not so high, the results are quite satisfactory for both the models, taking also into account a lack of samples and the unbalance of the dataset.

TABLE I
PERFORMANCE RESULTS FOR THE MODEL WITH TWO CLASSIFICATION AND ONE REGRESSION TASK ON THE TESTING SET

Classifiers	accuracy	loss	mean absolute error (mae)
Age classifier	/	0.607	5.960
Gender classifier	0.968	0.093	/
Ethnicity classifier	0.882	0.357	/

TABLE II
PERFORMANCE RESULTS FOR THE MODEL USING THREE CLASSIFICATIONS ON TESTING SET

Classifiers	accuracy	loss
Age classifier (age ranges)	0.760	0.603
Gender classifier	0.961	0.122
Ethnicity classifier	0.871	0.388

The result just shown on the testing set are further clarified by including the obtained confusion matrices for both the models, it is possible to distinguish the correct and incorrect prediction on the test set. In Fig 14 the confusion matrix of the age and gender for the model performing two classification task and a regression one are inserted. The age confusion matrix is not present for the age estimation since the latter,

[3]mae is the metric to evaluate a regression model, this metric tell us how accurate our predictions are and, what is the amount of deviation from the actual values.

being a regression task, is often inaccurate at predicting the exact age of a person due to the mean absolute error, rather prediction of a closer value implies to define an accurate range of ten years where the correct age falls with the predicted one. Therefore considering a confusion matrix on age estimation in this case would have mean to design a 83×83 (age considered are from 6 to 89) in which each predicted age should have been compared with all the true ones. Clearly this implies a too large confusion matrix and for these reason this has not been inserted in the paper and in the experiments.

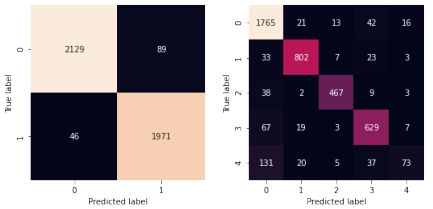

Fig. 14. Confusion matrices obtained for the experiments conducted using the model performing two classifications task and a regression one. Starting from the left, the gender and the ethnicity ones are displayed respectively

On the other hand in In Fig 15, the confusion matrices obtained on the test set by the model executing three classification tasks are considered. Here, since the age is estimated by classifying it in the related age range, the confusion matrix is present. Indeed, as mentioned above, the age ranges are in total four and they are seen as classes. Nothing change for ethnicity and gender classifications.

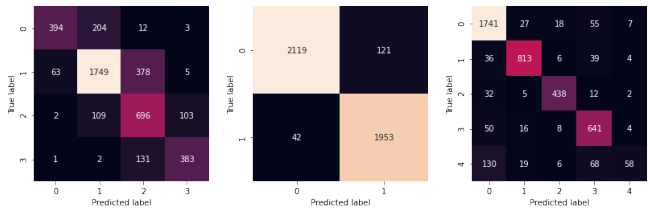

Fig. 15. Confusion matrices obtained for the experiments conducted using the model performing three classification tasks, here the confusion matrix related to age ranges is also present. Staring from the left, age, gender and ethnicity confusion matrices are inserted.

To better have an idea of how much the models are good at predicting the correct label for a given image, some samples contained in the dataset are given as an input. In Fig 16 some good predictions given by the model with two classification and one regression task on random samples are shown, here, the gender and ethnicity labels have been transformed in their respective string values. Furthermore, given a certain age, the age range is also displayed. This is useful to introduce a certain uncertainty in the age estimation, since sometimes can happen that an age is not perfectly predicted, but still, the correct age is in the age range of the predicted one.

In Fig 17, instead, good predictions given by the model adopting three classifications are shown. Here, in opposition from the first model, the age prediction is in term of age ranges defined as classes. This leads to loose a certain level of detail since age ranges are wide and it is easy to fall in the correct classification

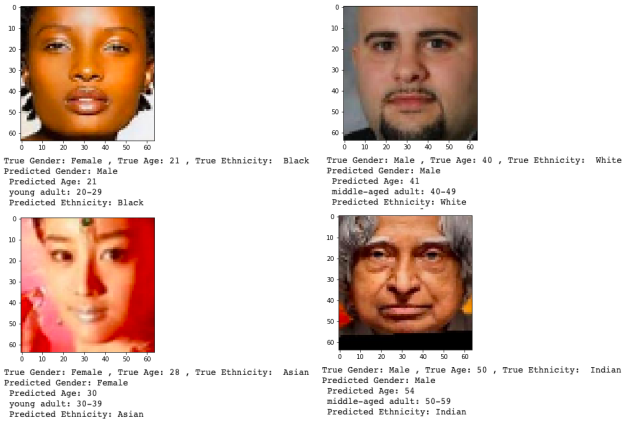

Fig. 16. Examples of some good predictions on given images from the overall dataset performed by the two classifications and one regression task model

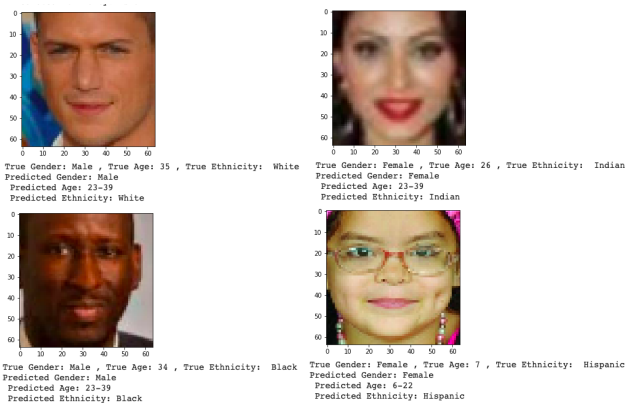

Fig. 17. Examples of some good predictions on given images from the overall dataset performed by the model adopting three classification methods

Of course, incorrect predictions are present as well for both the models. Sometimes happens that misclassification of one or more labels occur. This can have many reasons, starting from the fact that the dataset is not correctly balanced or that the network is not able to classify the image due to the lack of similar samples. In Fig 18 some bad prediction performed by the two classifications and one regression task model are available. Considering the first image on the left an incorrect ethnicity classification has occurred. This may have been caused by the fact a lack of Indian people in the dataset is present, and this may lead to an incorrect classification between Black and Indian people. In the second image on the right, another incorrect ethnicity classification is shown. In this case it could have been caused by the light of the photo which make the skin of the retracted face tricky for the CNN (could also have been caused by a lack of samples with different light colors and face positions). In the two images on the bottom part, incorrect age estimation and incorrect gender classification occurred. In the first case the situation is induced by the fact that the mean average error tells us that the variation error in the age estimation is around ± 5.9. In the second case, the incorrect gender classification may be due to a lack of samples of young males with long hair. In any case there could be

plenty of reasons behind incorrect predictions, that why we should always try to increase the dataset with more samples, improve the model, or sometimes change the hyperparameters.

Similarly in Fig 19 some bad prediction of the three classification model are shown. The reasons behind them are more or less the same of the first model, the only difference is that the age prediction is in most of the cases correct due to the size of the age ranges, therefore the most common mistakes are in the gender and ethnicity predictions.

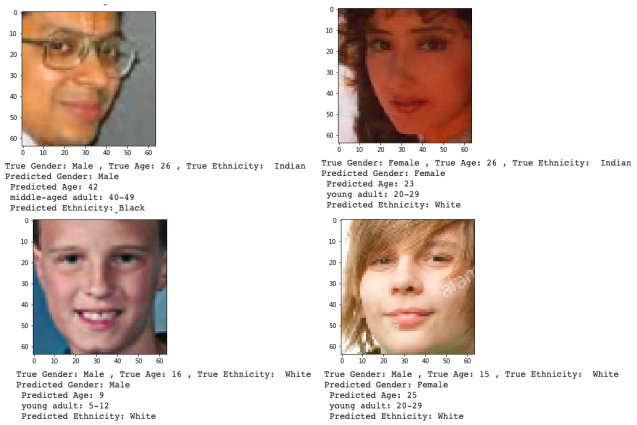

Fig. 18. Examples of some incorrect predictions on given images from the overall dataset in the case of the model with two classifications and one regression tasks

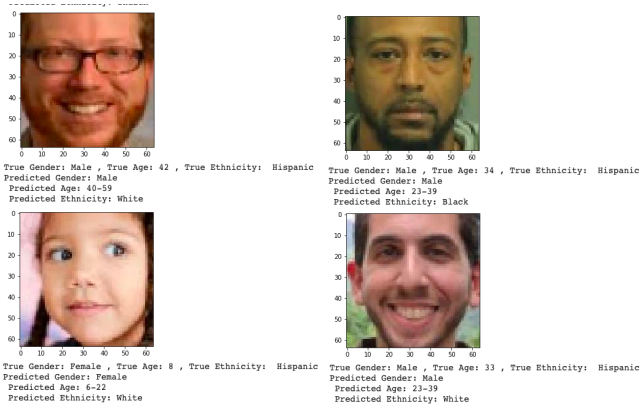

Fig. 19. Examples of some incorrect predictions on given images from the overall dataset of the model performing three classifications tasks

V. Face Detection and Classification

In the last part of this experimentation the classification of detected faces from external images taken outside the dataset is considered. As said in the introduction, the application is designed to classify unknown and non-profiled subjects from external sources (e.g images or videos) and to give an initial estimation and classifications of their age,gender and ethnicity. To implement such a system, OpenCV is a good starting point. Basically to detect faces we rely on the well-know $CascadeClassifier()$ [4] used an effective object detection method proposed by Paul Viola and Michael Jones.

[4]https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

Since the application work with only detected faces, the $haarcascade\_frontalface\_default.xml$ is the only Haar cascade model used to achieve this result. Afterwards, the detection is done using the $detectMultiScale()$ method, which returns boundary rectangles for the detected faces.
However, the $detectMultiScale()$ method parameters have to be carefully chosen. Fist of all the $scalefactor$ parameter, which specifies how much the image size is reduced at each image scale (used to create the scale pyramid.), plays an important role in face detection. In fact, by rescaling the input image, a resize of a larger face towards a smaller one is possible, making it detectable for the algorithm. This rescaling affects also the pixel values of the detected faces influencing directly the predictions of the pre-trained CNN. Therefore given an input image to the CascadeClassifier, the scale factor is set to 1.05 which means reducing size by 5 %, this increase the chance of a matching size with the model for detection. However, this parameter is also modified when irrelevant faces are detected. The other parameters are kept in their standard values without making any modification. In Fig 20 an example face detection is shown.

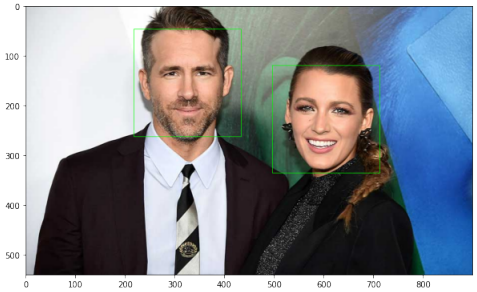

Fig. 20. Detection of faces in an external image

Using OpenCV means dealing with BGR images, therefore the detected faces are then converted to RGB, resize to $64 \times 64 \times 3$ images and their pixels are saved in a single line inside a pandas dataframe in a similar fashion already seen in Section IV-C.
At this point, in the classification of the detected faces phase, the image are extracted from the dataframe, the pixel are again reshaped in $64 \times 64 \times 3$ and normalized between 0 and 1. This step is of a crucial importance to obtain good classifications when dealing with Convolutional Neural Networks. Indeed, when giving an image in input to a CNN, it must have the same shape and pixels normalization as the images in the dataset on which the network has been trained.

Even by tuning the $scalefactor$ parameter to the value which permits to detect all the faces in a image, there are cases where this is not enough for a good prediction. In Fig 21 correct prediction performed by the model using two classifications and one regression task are displayed. The result obtained have been checked on the web by searching for the name of the celeb in the images.

By contrast in Fig 22 incorrect classification are shown. This could occur in one of the three predictions, but the most recurrent is the ethnicity one (probably the lack of

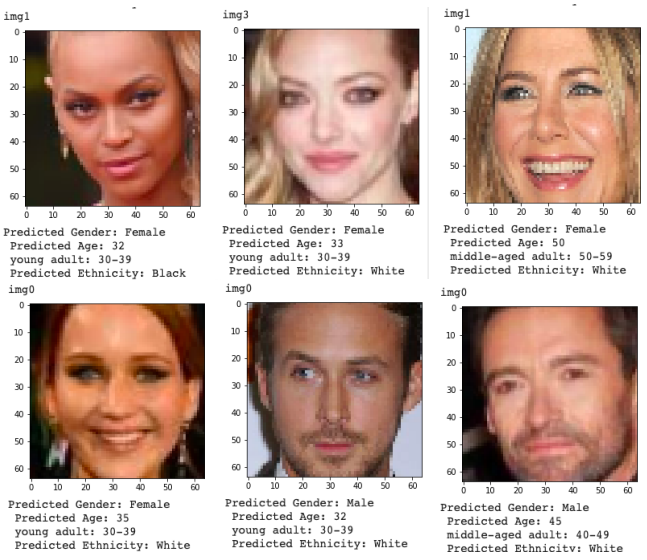

Fig. 21. Example of good predictions on random Celeb images found on the web performed by the model performing two classification task and a regression one

certain ethnic group samples has an important impact on this classification). Incorrect classification can be given by the nature of the image itself. Lights effects, face position, photo editing, make up and so on. All these aspects can dramatically change the predictions of the network since the Dataset used is not very large.

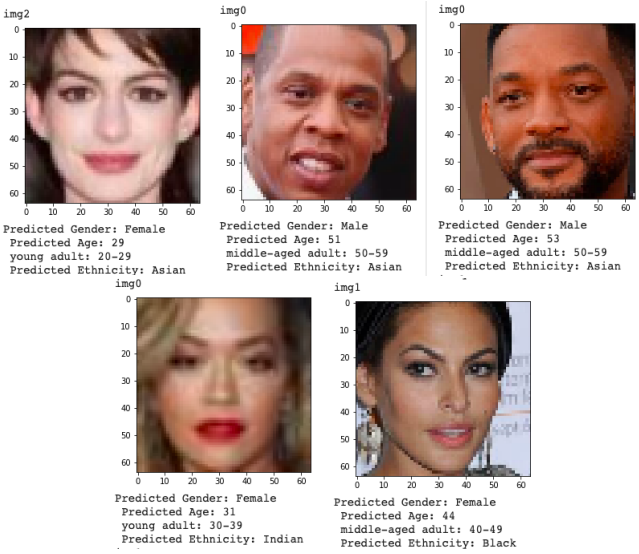

Fig. 22. Example of incorrect predictions on random Celeb images found on the web

In Fig 23 the same samples are valuated with the model performing three classifications task. Here, for the first time we can notice, looking at the predictions, that they are not as accurate as in the first model at predicting both ages and ethnicity. This could be give by a lack of samples within the same age ranges causing a misclassification the predicted class. Moreover, for the ethnicity misclassification, this could be induced by the motivation explained above. In any case this model seems to be worse than the one using two classifications and a regression task on external samples.

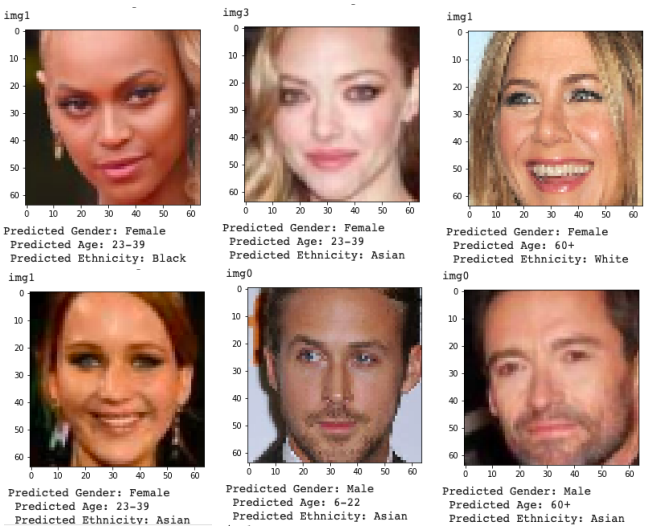

Fig. 23. Example of predictions on random Celeb images found on the web performed by the model performing three classification tasks

Again, it is important to remember that the prediction are affected by the $scalefactor$ since directly operates on the image pixels, with different values for this parameter the outcomes could be totally different.

VI. Conclusion

In this paper a face detection and classification application has been described. The face classification task has been executed as a multi-label classification problem comparing two approaches. The first by using a model which performs two classifications and one regression task in order to estimate the age, the second by adopting three classification aiming to estimate an age range of the detected face. After executing some experiments assuming the same conditions for both the models, we have demonstrated that the first approach is better an more precise. As a matter of fact the second, by estimating an age range, tends to loose a level of detail. This is further underlined when facing external samples. In this case the second model performs worse prediction compared to the first one. By contrast, by predicting the exact age, with the regression task, of a detected face it is possible to define a level of uncertainty of 10 years in which the predicted age falls along with the correct one.

Further improvement can be done using a different platform which allows the use of a much more larger dataset. In fact, when performing face classification a huge dataset should be needed. This is because each human being is unique and sometimes it is difficult to estimate his age, or to classify his gender and ethnicity. Having a large dataset with more samples with different face skin colours, parts (e.g eyes, noses, lips, etc.), make up, face orientation and position, but also with different light effects, photo editing, background colors and so on, can help the network at learning and therefore to classify better external samples outside the

dataset. Furthermore a really nice improvement would be to assure some sort of balancing between samples of each class. As a matter of fact having, as in this case, a deeply unbalanced dataset containing by far more samples of 26 year old people affects deeply the age regression task (and olse the classification in age ranges). The same for the ethnicity classification. The latter is influenced too much by the presence of white and black people with respect to Asian, Indian and Hispanic ones. This situation leads the model to perform incorrect predictions when a person has a darker complexion, or when the photo light tends to be darker or lighter. Hence, making the dataset larger with more samples would result in greater heterogeneity and therefore and improvement of predictios of external samples.

Despite OpenCV allows a good face detection, this system is not perfect even by tuning the $scalefactor$ parameter. During the experiments, some faces inside groups of people where not detected at all. This problem is solved by using the MTCNN (Multi-task Cascaded Convolutional Networks) [2] which performs a better face detection. However, the problem with this methodology is that the retrieved detected face is rectangular instead of square. This is not suitable for the pre-trained CNN since by resizing the image as a square this would mean to stretch it making difficult for the network to perform a good classification.

Even though both the models could be further improve we are satisfied for the results achieved. By having access to more resources, a video analysis could be also carried out facing a real situation, for example when we need to profile a people captured by video surveillance. However also for images this could be an useful solution when human operators are not fast enough or they have to deal with many requests.

REFERENCES

[1] J. Brownlee, "How do convolutional layers work in deep learning neural networks?" *Machine Learning Mastery*, 2020.

[2] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.