Age, Race and Gender Detection Project

Abstract. Since the emergence of social platforms and social media, automatic age and gender classification pertinent to a growing number of applications. Nevertheless, compared to the enormous performance improvements reported recently for tasks related to facial recognition, the performance of present approaches on real-world images still falls far too short. In this project, we demonstrate how performance on these tasks may be significantly improved by learning representations using deep Convolutional Neural Networks (CNNs). In order to do this, we employed a straightforward convolutional net architecture that can be applied even with a finite supply of training data. We assessed my model using the most recent UTKFace dataset to estimate age, race and gender.

Student 1: Parth, 2020UCA1812
Student 2: Mukul Gupta, 2020UCA1847

Netaji Subhash University Of Technology, New Delhi
CACSC19, CSAI-1

17th April, 2023

1. Introduction

Facial analysis has emerged as a crucial aspect in various applications, including security, entertainment, healthcare, and marketing. One of the essential tasks in facial analysis is classifying the age, race, and gender of a person. With the advent of social media and social platforms, automatic age and gender classification have become increasingly important. However, the current approaches for age and gender classification tasks on real-world images have limitations in their performance. This paper proposes a deep learning-based approach to improve the performance of age, race, and gender classification tasks significantly. The proposed approach uses the UTKFace dataset, which contains diverse real-world imaging situations, to train and evaluate the model. The approach employs a simple convolutional neural network architecture, which can be applied effectively even with a limited supply of training data. Additionally, data augmentation techniques are employed to expand the size of the training set. The experimental results demonstrate that the proposed approach achieves high accuracy in age, race, and gender classification tasks.

2. Problem Statement

The goal is to create an application capable of identifying a person's age, race and gender, given the image of the person's face. The following broad tasks are to be achieved:

i. Download and preprocess UTKFace dataset.
ii. Train a classifier to generate outputs that correspond to 'age', 'race' and 'gender' based on the numerical representations of the images.
iii. Evaluate the model using the test dataset.

3. Metrics

classification_report is a method available in the *scikit-learn* library available in Python to build a text report, which we will be using to evaluate our model against the test dataset.It uses the following main classification metrics:

a.) Accuracy: The fraction of predictions that are correct as compared to all of the predictions (correct or incorrect).

$$Accuracy = \frac{true\ positives + true\ negatives}{dataset\ size}$$

b.) Precision: The fraction of positive predictions that are actually correct as compared to all the positive predictions.

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}$$

c.) Recall: The fraction of positive predictions that are actually correct as compared to all the positive cases in the data.

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

d.) F1-score: Combines both, *precision* and *recall* by taking the harmonic mean of the two.

$$F1\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall}$$

e.) Support: Number of actual occurrences of a label in the dataset.

4. Analysis

4.1. Data Exploration

The dataset UTKFace is a large-scale face dataset containing the faces of people whose ages range from 0 to 116 years. 23,705 pictures with annotations for age, race and gender make up the dataset. This dataset, which contains a variety of real-world imaging situations like noise, illumination, pose and look, serves as a standard for face photographs.

The file name contains integrated the labels to the corresponding image, formatted as '`[age][gender][race]_[date&time].jpg`'.

i. '`[age]`' is an integer specifying the age.
ii. '`[gender]`' is a binary digit (0 or 1) denoting the gender (male or female respectively).
iii. '`[race]`' is a digit between 0 and 4 (inclusive) denoting the race - White, Black, Asian, Indian and Others (Hispanic, Latino, Middle Eastern, etc.)
iv. '`[date&time]`' denotes the date and time using the format '`yyyymmddHHMMSSFFF`'.

4.2. Exploratory Visualization

Each image is colored and has the average dimensions of `(200,200,3)`. The average age of a person is 33 years after rounding off to the nearest integer. The minimum and maximum age of a person is 0 and 116 years respectively.

Fig. 1 Sample Input Images from UTKFace dataset

The following *boxplots* show how the age distribution varies with gender and race.

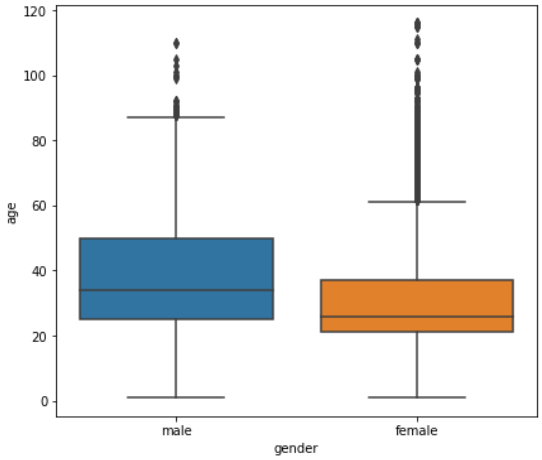

Fig. 2 Age distribution corresponding to both genders

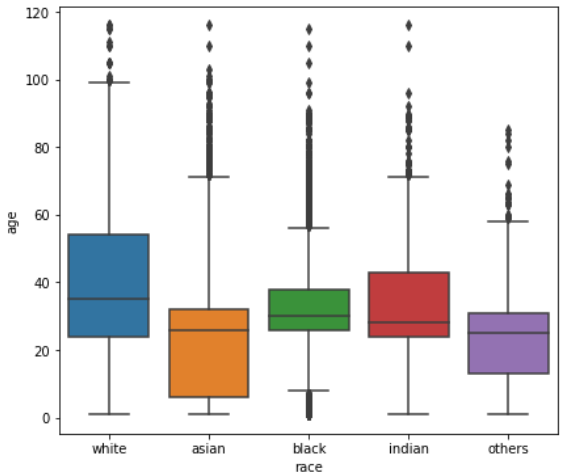

Fig. 3 Age distribution corresponding to all races

The boxplots reveal some information about our the data. Most of the male-gendered people have their ages between 25 and 50, while most of the female-gendered people have their ages between 20 and 40. Most of the White people, Asians, Black people, Indians and 'Others' range in the interval (20,50), (5,30), (25,40), (25,55), (25,50) and (10,30) respectively.

The following histogram visualizes the distribution of age:

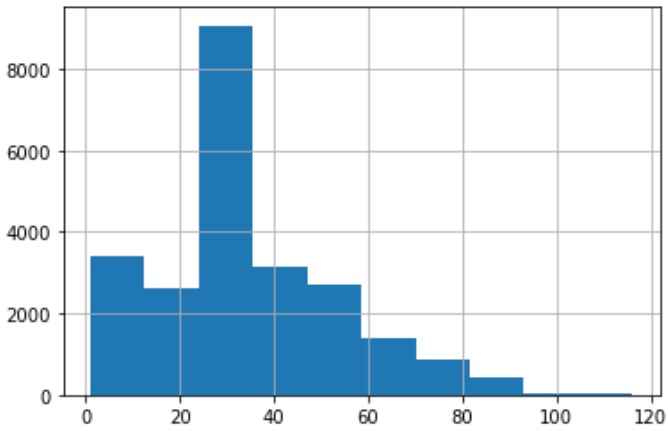

Fig. 4 Age distribution depicted by a histogram

4.3. Algorithms and Techniques

The classifier used is Convolutional Neural Network, which is the state-of-the-art algorithm for the majority of image-processing tasks. However, it requires much larger datasets than the other options. Fortunately enough, the UTKFace dataset is large enough for our application. CNNs are Artificial Neural Networks (ANNs) that convolute *kernels*/*filters* along the *input layer* and provides responses, called *feature maps*.

CNNs break down an image into smaller, simpler features represented by *filters*, instead of attempting to process the entire image at once. To extract the pertinent data, these *filters* are applied to different regions of the *input layer*. The hierarchical ap-

proach allows CNNs to learn complex patterns in data more efficiently, while reducing the risk of *overfitting*. CNNs are more "automatic" than other image classification algorithms, which means the network learns to optimize the *filters* on its own. This makes CNNs faster and more efficient than other methods.

The following hyper-parameters can be tuned to improve the efficiency of our CNN classifier:

a. Number of *epochs*
b. Batch size
c. Learning rate
d. Weight decay
e. Momentum
f. Number of Layers
g. Layer types
h. Layer parameters
i. Activation function
j. *Loss function* and its parameters

5. Methodology

This section demonstrates data preprocessing and the actual implementation of our classifier.

1. Data Collection and Preprocessing:
The UTKFace dataset will be used for training the convolutional neural network (CNN). The dataset consists of over 20,000 facial images with annotations for age, race, and gender. The dataset will be preprocessed by resizing the images to a fixed size, converting them to grayscale, and normalizing the pixel values to a range between 0 and 1.

2. Data Augmentation:
To increase the size and diversity of the dataset, data augmentation techniques will be applied, including rotation, translation, and flipping of the images. This will help prevent overfitting and improve the generalization ability of the model.

3. Model Architecture:
A convolutional neural network will be used for age, race, and gender prediction. The architecture will consist of several convolutional layers followed by pooling layers, which will extract features from the images. The output from the convolutional layers will be fed into fully connected layers that will classify the input image into different age, race, and gender categories.

4. Training and Validation:
The model will be trained on the UTKFace dataset using a stochastic gradient descent optimizer with a learning rate of 0.001. The model will be trained for 50 epochs with a batch size of 64. The training process will be monitored using a validation set, and early stopping will be applied if the validation accuracy stops improving.

5. Evaluation:
The trained model will be evaluated on a test set to measure its accuracy, precision, recall, and F1-score. Additionally, confusion matrices will be generated to show how the model performs on different age, race, and gender categories.

6. Fine-tuning:
To improve the performance of the model, fine-tuning techniques such as transfer learning can be applied. The pre-trained models such as VGG, ResNet, and Inception can be used as a starting point to improve the performance of the model.

7. Deployment:
The trained model can be deployed in various applications, such as facial recognition systems, age estimation applications, and gender/race-based marketing analysis. The model can be deployed on a cloud-based platform or on-premise servers, depending on the requirements.

5.1. Data Preprocessing

The following steps are taken for the preprocessing of our data:

a. Each image is resized to the dimensions `(198,198,3)`. Each pixel value is normalized between 0 and 1.
b. The order of the images is randomized. The data is split into training and test datasets.
c. ‘gender’ and ‘race’ are one-hot encoded. ‘age’ is normalized between 0 and 1.

d. Each image is convert into its numerical form using *numpy* arrays.

5.2. Implementation

The process of implementation can be divided into two steps:

a. Creating the model.
b. Fitting the model on the training dataset (training).

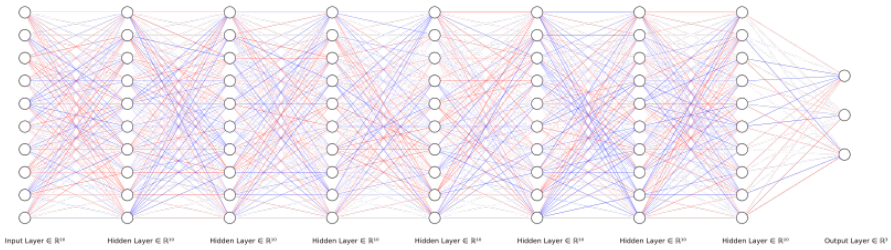

Fig 5. A representation of the CNN

During the first step, the basic structure of the CNN classifier is created. This step can be further divided into the following steps:

a. Creating and setting up the parameters for one layer.
b. Creating multiple layers using those settings, one on top of the other.
c. Creating a *bottleneck layer*.
d. Adding 3 *dense layers*, each corresponding to one output ('age', 'race' and 'gender').
e. Selecting *loss functions* corresponding to each output.
f. Selecting *metrics* corresponding to each output.

Here, a few choices have to be made like choosing the *activation function*, the *loss function* and the metric for each *dense layer*. Here, by a little experimentation and a little research, we land on the following conclusion:

	Age	Race	Gender
Loss Function	Mean Squared Error	Categorical Cross Entropy	Categorical Cross Entropy
Activation Function	Sigmoid	Softmax	Softmax
Metric	Mean Absolute Error	Accuracy	Accuracy

There are 9 layers in total; the first layer is the *input layer*; layer 2 to 7 (inclusive) are *hidden Convolutional 2D layers*; layer 8 is the *bottleneck layer*, which is a *Global-MaxPool2D layer* that downsizes the previous layer; layer 9 is the output layer. Additionally, *RMSProp* was used for gradient descent.

The shape of the *input layer* is `(198,198,3)`. The first layer contains 32 *filters*, the second layer contains 32*2 *filters*, the third layer contains 32*3 *filters* and so on. The *dense layer* contains 128 units.

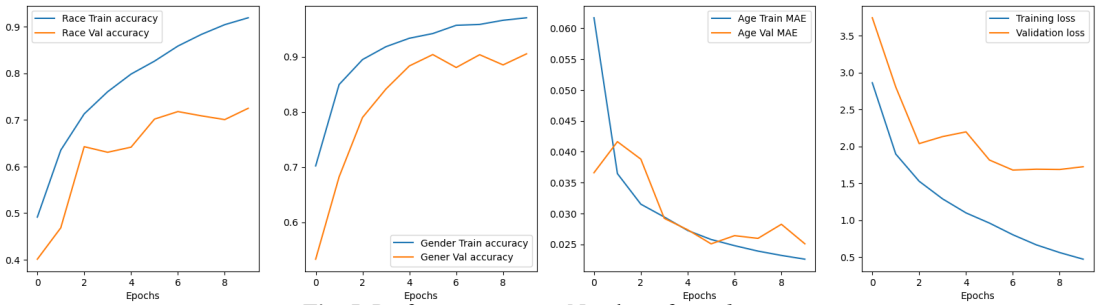

Fig. 5 Performance versus Number of *epochs*

Now, this architecture is fit over our pre-processed data. *Batch size* is taken as 64 and observations of the performance of the model is measured after each *epoch*.

6. Result

Now that our classifier is trained on the training dataset, it is time to evaluate it on the test dataset. After pre-processing the test data, predictions are made using *predict_on_batch()* function.

The following metrics indicate the output accuracy and the loss of the classifier when predicting 'age', 'race' and 'gender'.

```
'loss':                     1.7623138427734375,
'age_output_loss':          0.02597474865615368,
'race_output_loss':         0.9579477310180664,
'gender_output_loss':       0.2734430134296417,
'age_output_mae':           0.12785083055496216,
'race_output_accuracy':     0.7198401093482971,
'gender_output_accuracy':   0.9117006063461304
```

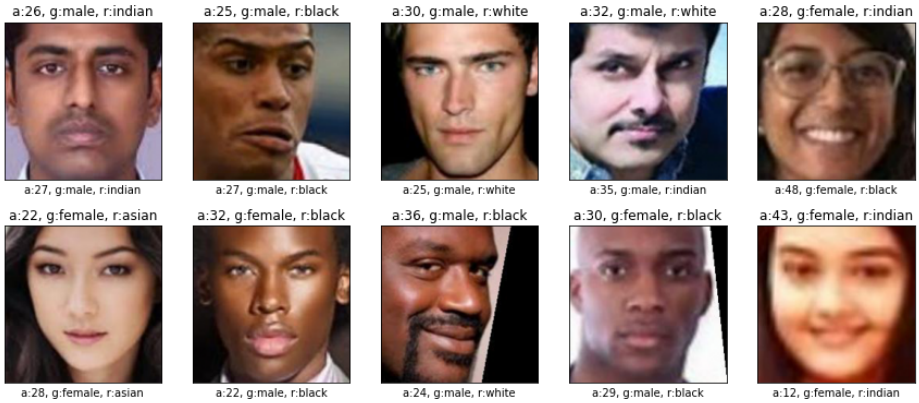

Fig. 6 Sample output

A Classification Report is generated using the *classification_report* method for 'race' and 'gender'. The following result is obtained:

```
Classification report for race

              precision    recall  f1-score   support

           0       0.80      0.73      0.77        49
```

```
              1       0.81      0.93      0.87        28
              2       0.50      0.87      0.63        15
              3       0.68      0.65      0.67        23
              4       0.33      0.08      0.12        13

    accuracy                           0.71       128
   macro avg       0.63      0.65      0.61       128
weighted avg       0.70      0.71      0.69       128


Classification report for gender

              precision    recall  f1-score   support

           0       0.94      0.88      0.91        72
           1       0.85      0.93      0.89        56

    accuracy                           0.90       128
   macro avg       0.90      0.90      0.90       128
weighted avg       0.90      0.90      0.90       128
```

7. Conclusion

The CNN model recognizes the age and gender of a person given the image of their face very accurately without much output loss. It does not perform as well when predicting the race (71% accuracy), however one must remember that often people do not belong to a single race and are mixed (for example, 50% White and 50% Indian). This is likely the reason we do not achieve such a great performance when predicting the race, since multiple classification solutions to a single image exist, and the model has to choose between them.

We believe that our project demonstrates the point that *Convolutional Neural Networks* work much better than the other existing techniques when it comes to applications of image processing such as face recognition.

8. Reflection

The making of this project can be summarized in the following steps:

a. Describing the problem, and downloading the relevant dataset.
b. Analyzing, segmenting and pre-processing the data.
c. Creating the architecture for the classifier.
d. Training the classifier on the training dataset.
e. Evaluating the model using the test dataset.

Personally, step c was the hardest for us since there was the need to tune the hyper-parameters to optimal values for our application. This step required much research from other people's projects.

Making this project was a very insightful experience for the both of us as it allowed us to expand the boundaries of our knowledge relating to neural networks.

9. References

Keiron O'Shea, Ryan Nash (2015). An Introduction to Convolutional Neural Networks
arXiv:1511.08458 **[cs.NE]**