Project of Human Age Detection

Jan Kruszewski, Kacper Leszczyński, Michał Olędzki, Patryk Prusak, Adrian Rudź

supervisor
dr hab. inż. Agnieszka Jastrzębska

Warsaw University of Technology
24.01.2024

Contents

1 Introduction

Computer vision is a significant branch of machine learning. From FaceID[1] to self-driving cars[2], some computers, whether in the form of a smartphone or car on-board computer, possess the ability to interpret their environment in terms of visual data. To explore how machines can learn to see, we implement a project of human age recognition consisting of exploratory data analysis, face recognition, and the creation and training of an age recognition model together with statistical tests, use-case scenarios, and a user manual.

1.1 Requirements

The created solution must have the following functionalities:

1. an application with a simple GUI allowing the user to detect age through a web-cam, images stored in a certain directory, or a video file
2. source code for data preprocessing, training, testing, and saving the model
3. age detection should happen by marking bounding boxes for each visible face and displaying a predicted age, additionally in the case of image and video files, the results should be saved on the hard drive
4. face recognition can be implemented via an already existing model, in the case of many available models, the choice can be given to the user with the use of GUI, the program mustn't accept any command line arguments
5. models should be tested and assessed in terms of established metrics such as precision, recall, F1 score, sensitivity, accuracy, and errors: RMSE and MAPE
6. some testing scenarios should be prepared and documented

1.2 Existing Solutions

Understandably, human age recognition found its use in real-world scenarios in business or entertainment needs. There is a range of applications that offer some form of human age recognition. One can find several free solutions, i.e.:

1. How old do I look - Age app[3].
2. How old am I? Face age app[4].

Both are available to download from the Google Play Store, however, there is no information regarding the details of implementation. Looking at the reviews, their performance is good enough for entertainment purposes but not much more. There are also more reliable business solutions such as Yoti - facial age estimation[5] or Everypixel Labs - age recognition[6] offering more security and robustness for a given price. One can conclude then that there is a space for a free human age recognition solution offering more robustness and transparency.

2 Exploratory Data Analysis

A necessary prerequisite for machine learning to occur is a collection of data - a dataset[7]. A chosen or collected dataset needs to be explored, visualized, and understood in terms of certain characteristics, this process is called Exploratory Data Analysis (EDA)[8]. This allows for informed decisions in terms of augmentation or modification of data going forward, and perhaps can even explain certain traits of the trained model. EDA can consist of different processes such as: identifying patterns, assessing quality, checking whether the dataset is balanced, using visualizing methods such as boxplots or histograms, and various other statistical tests depending on needs and purposes.

2.1 Dataset

UTKFace[9] is a dataset of choice. It is a large-scale face dataset available free of charge that:

1. consists of 20k+ face images in the wild (only a single face in one image)
2. provides the correspondingly aligned and cropped faces
3. provides the corresponding landmarks (68 points)
4. contains images that are labeled by age, gender, and ethnicity

There are similarly good alternatives available, such as Faces: Age Detection Dataset[10] or Age Detection - Human Faces[11]. However, given the range of ages, size, popularity, and quality of images, we have decided to use UTKFace.

2.2 UTKFace contents

Upon closer inspection, the dataset consists of 23705 images of ages ranging from 1 to 116, with the average age being 33 years old. All images are 200 by 200 pixels and the average size is 5 KB. At this point, UTKFace's parameters seem to be enough for the needs of the task at hand.

	age	**file size (KB)**	**width**	**height**
count	23705.000000	23705.000000	23705.0	23705.0
mean	33.300907	4.942352	200.0	200.0
std	19.885708	0.806115	0.0	0.0
min	1.000000	2.011719	200.0	200.0
25%	23.000000	4.402344	200.0	200.0
50%	29.000000	4.855469	200.0	200.0
75%	45.000000	5.366211	200.0	200.0
max	116.000000	11.743164	200.0	200.0

Table 1: Dataset overview

UTKFace contains the largest amount of images for people in their 20s, which might make the trained model biased, it is something that needs to be taken into consideration when assessing the quality of the trained model.

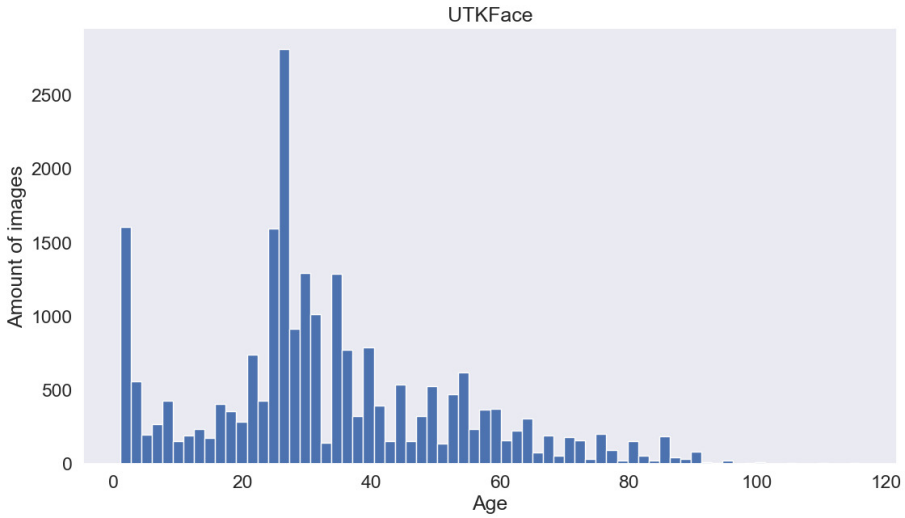

Figure 1: Age distribution

The constructed boxplot shows that there are some outliers present. Outliers are values that notably differ from the overall pattern. However, since removing them would mean the dataset's range of ages would decrease, we decided to keep them to ensure the model is trained on the whole available range of ages in hopes of improving accuracy.

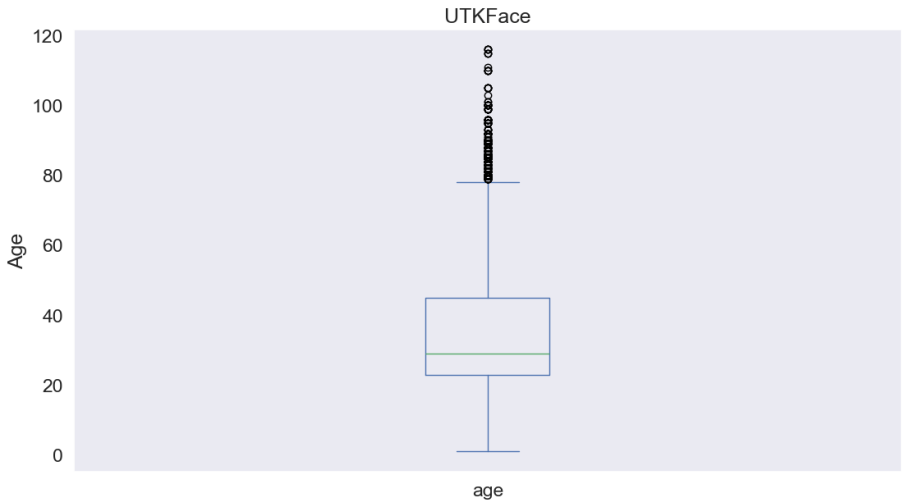

Figure 2: Age boxplot

2.3 File Size

All images are of resolution 200 by 200 pixels resulting in reasonable file sizes, with the average file size being 4.94 KB. Furthermore, images can be downsized during data preprocessing if needed to improve training speed.

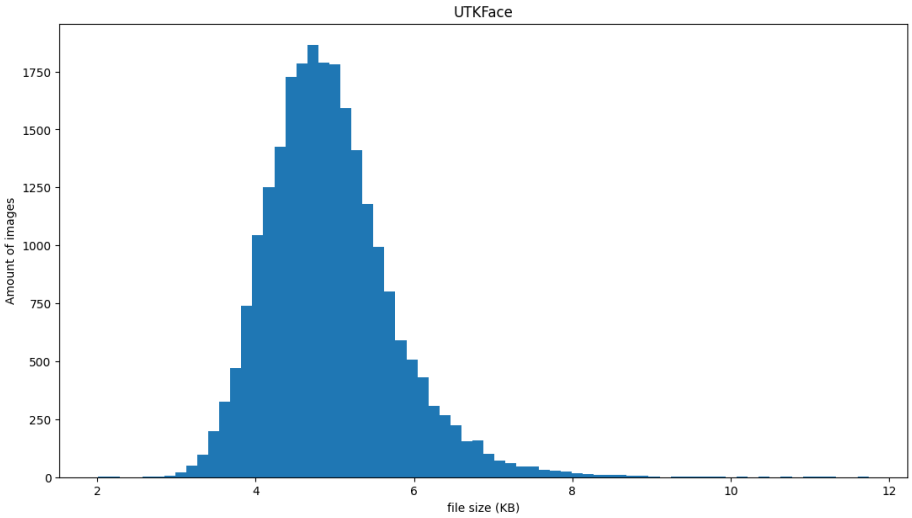

Figure 3: File size distribution

2.4 Color Distribution

Upon analyzing the mean color distribution present in the dataset images, one can conclude that black pixels are dominant. A possible reason for that might be the dark background present in pictures or perhaps the photos were taken in dark lighting conditions. This could be verified by examining the images. Moreover, investigating distributions for particular RGB channels reveals certain trends. Images of the youngest people have a higher mean for each of the RGB channels which indicates a higher brightness of these images. As we progress toward people in their 20s, the RGB mean lowers indicating images of lower brightness. Finally, the mean slightly increases towards the end of the dataset. Overall, these insights although not significant at first glance might prove to be useful later on.

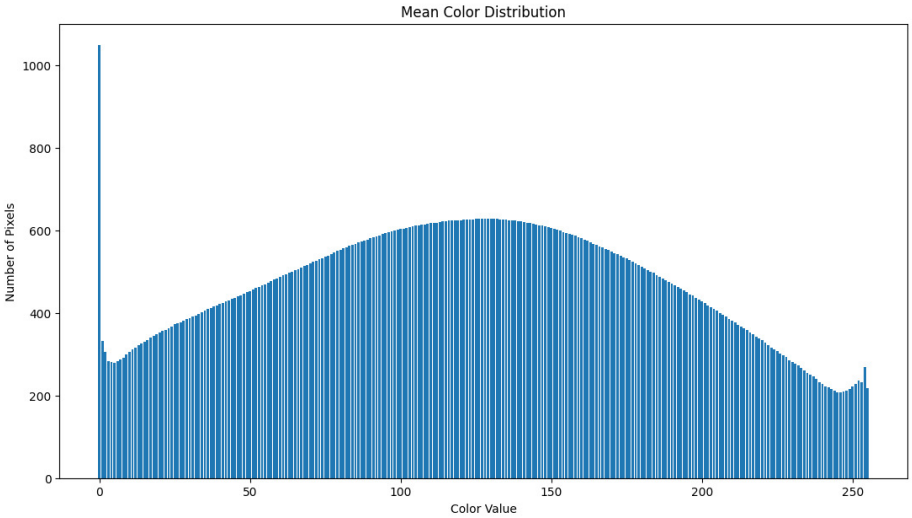

Figure 4: Mean color distribution

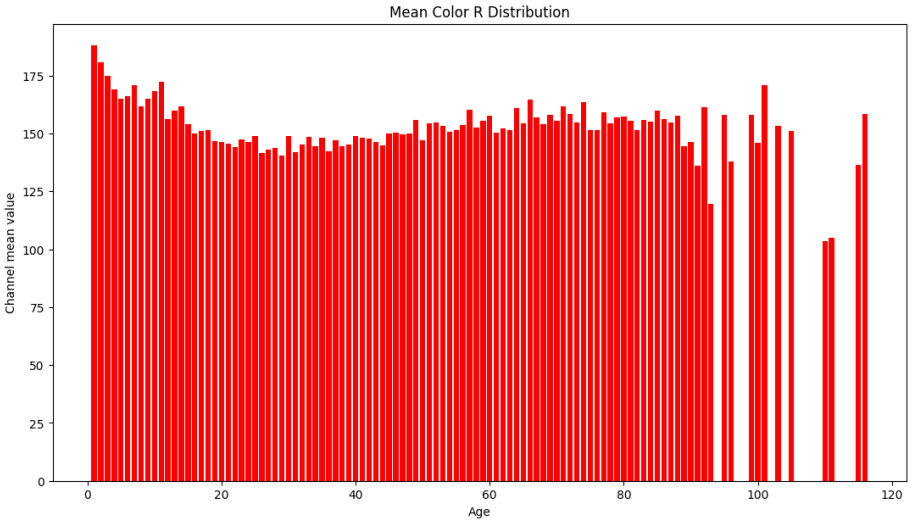

Figure 5: Mean red channel color distribution

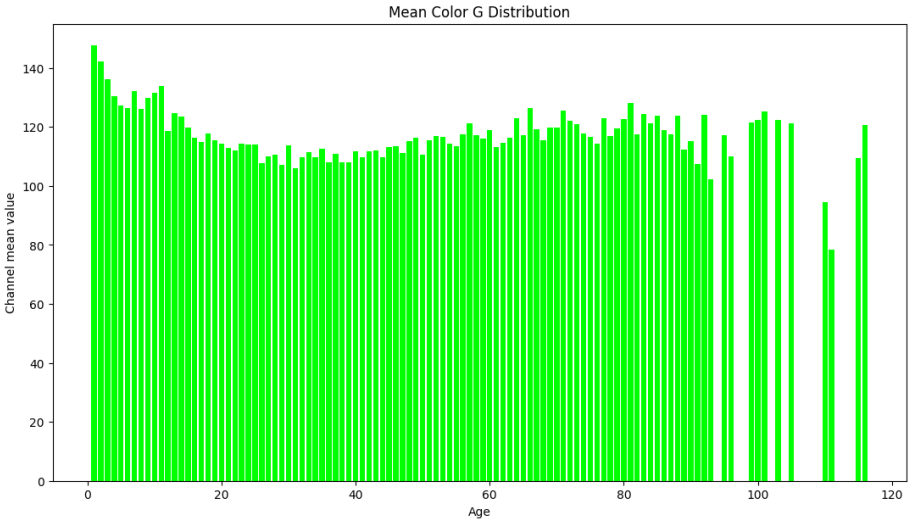

Figure 6: Mean green channel color distribution

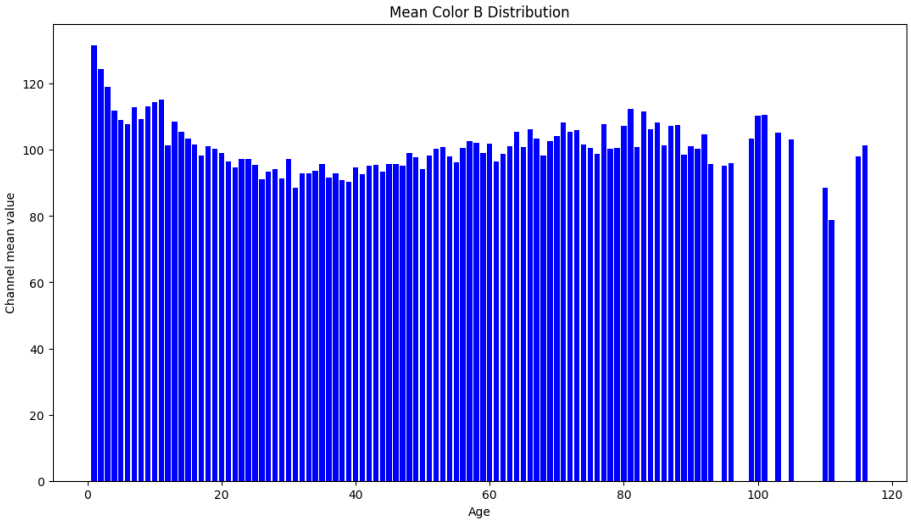

Figure 7: Mean blue channel color distribution

2.5 Grayscale histogram

Images can be transformed to grayscale and analyzed in regards to the image's histogram. In the below example, the histogram detects a large number of light pixels (>200) which most likely correspond to the face, and many dark pixels (<50) corresponding to the background.

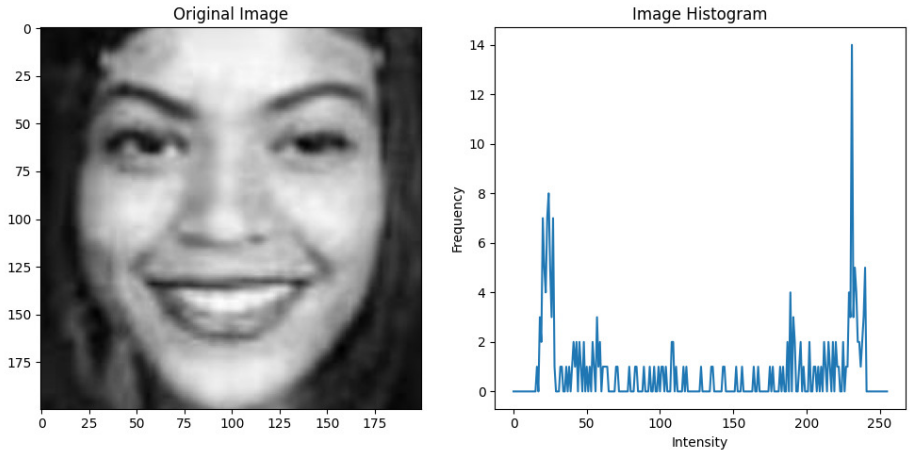

Figure 8: Image histogram

2.6 Similarity

Structural Similarity Index (SSIM), first introduced in 2004[12], is a measure of similarity between two images. SSIM score ranges from -1 to 1, the higher the value, the more similar the images. Dataset images have been sorted from youngest to oldest and then compared between each other, starting from the first vs last picture and advancing towards the middle. The average SSIM computed in such a way amounted to

$$0.33523684987556973$$

suggesting that the images in the dataset are more similar to each other than not similar.

2.7 Histogram of Oriented Gradients

The Histogram of Oriented Gradients (HOG) is a popular feature descriptor used for object detection[13]. The process of obtaining HOG consists of:

1. calculating the gradient of pixel intensity
2. obtaining the orientation, and magnitude of gradient vectors
3. dividing the image into cells, for each cell a 9-bin histogram is calculated
4. optional block normalization

The HOG can then be visualized, some examples of HOG for images from the dataset are presented below.

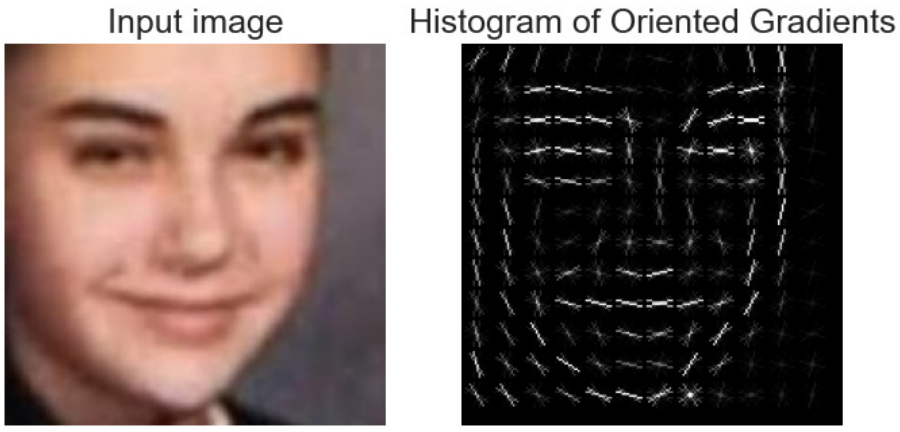

Figure 9: HOG 16 years of age

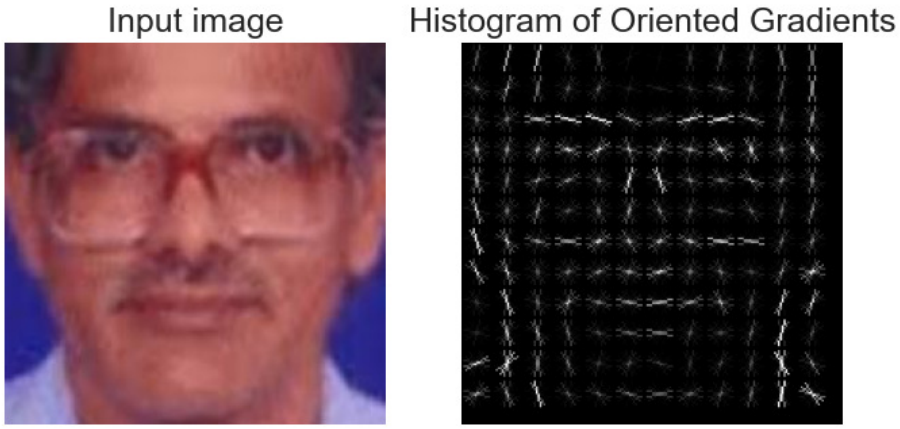

Figure 10: HOG 58 years of age

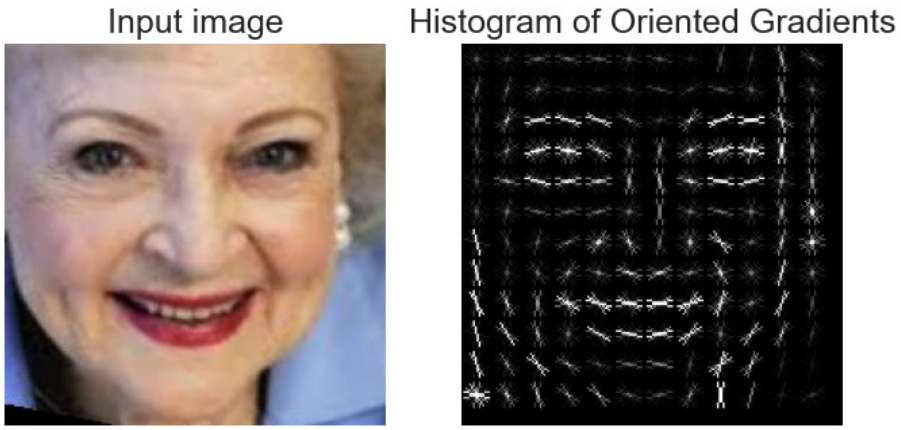

Figure 11: HOG 85 years of age

The images can be transformed to grayscale, which might reduce the effects of lightning and increase training efficiency since the images will then be of smaller size. Looking at the HOG results of grayscale images, one can notice that the features although still visible are bleaker.

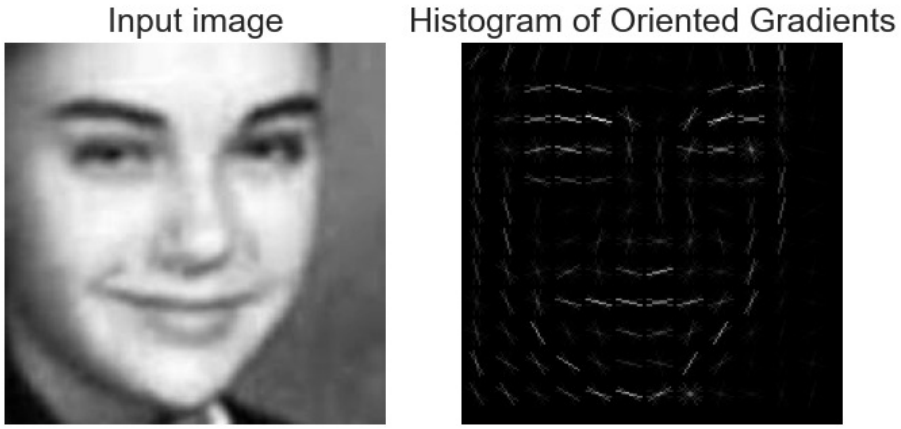

Figure 12: Grayscale HOG 16 years of age

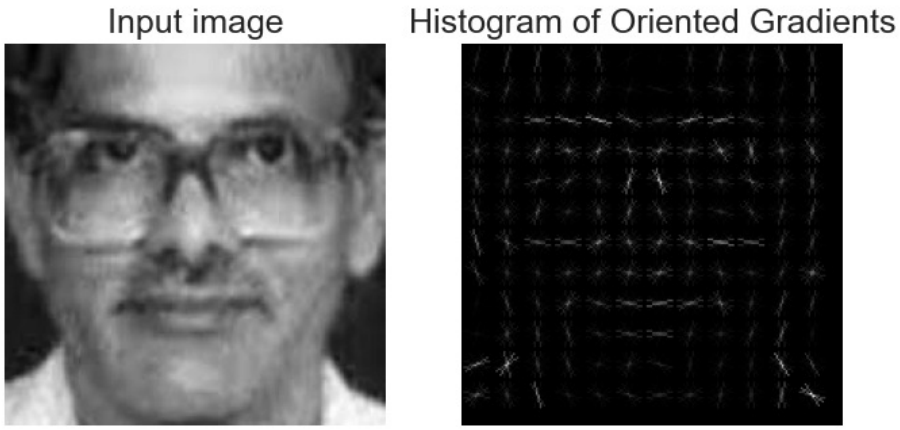

Figure 13: Grayscale HOG 58 years of age

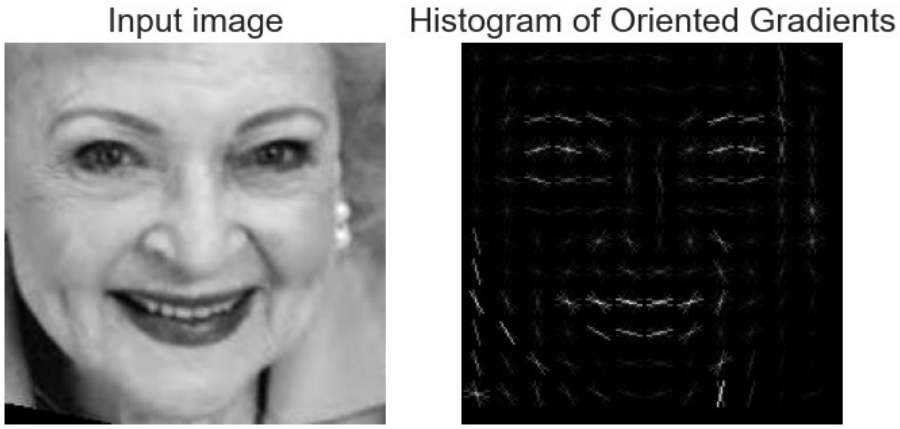

Figure 14: Grayscale HOG 85 years of age

2.8 HOG Comparison Between Classes

To assign different classes to the images from the dataset we need to find something that differentiates them. Using the histogram of oriented gradients one can compare different age groups. First, we divide the dataset into two groups: images of people younger than 50 years of age and images of people 50 or older. Next, we compute the histogram of oriented gradients for each image and count the average number of white pixels for each group. As a result for the first class (age < 50), the average number of white pixels in HOG is:

$$226.8372689389892$$

for the second class (age ≥ 50) the average is:

$$186.958692600762$$

what stands out here is the fact that the average of white pixels in HOG lowers as people's age increases. To present this trend more clearly one can divide the dataset into more classes, for example into images of people aged $[0, 10)$, then $[10, 20)$, and so on, this results in 12 classes (maximal age in the dataset is 116). In general a similar trend ensues, classes of people with older ages have fewer white pixels in HOG.

Age	**HOG mean**
<0,10)	425.36
<10,20)	325.44
<20,30)	168.46
<30,40)	181.74
<40,50)	170.91
<50,60)	182.03
<60,70)	192.34
<70,80)	191.29
<80,90)	212.60
<90,100)	123.77
<100,110)	123.68
<110,116>	45.38

Table 2: HOG average white pixels per age group

3 Face Recognition

The subject of Face Recognition is recognizing human faces in input visual data, this also extends to identification and verification based on facial features. Facial recognition technology has found its way into our everyday lives, i.e. smartphone unlocking via face recognition, and even to the justice system by aiding law enforcers to identify criminals[14].

The process of face recognition consists of detecting and locating a face in input data, i.e. from a webcam. Then the image can be normalized, which involves cropping, aligning, and compensating for differences such as lightning. Afterward, features are extracted from the normalized image that then can be classified as needed.

Behind face recognition stands a machine-learning algorithm, most commonly a Convolutional Neural Network (CNN)[15], other solutions are also possible i.e. Haar Cascade or HOG-based solutions. CNNs consist of multiple layers and can identify features from input data and then perform classification based on the extracted features. An important parameter of CNN is its depth which affects how sophisticated features the CNN can identify. As the input data, i.e. an image, goes through the layers, certain features are detected by filters and then the result is passed on to the next layer. Initially, the detected features can be quite simple such as lines or edges as the depth of the model increases so do the extracted features. The layers can be distinguished into types:

1. Convolutional layer is responsible for detecting features of an input by using a specific filter. The filter (or kernel), a matrix, is being moved over the image, at each point calculating the dot product between the kernel's weights and pixel's values under the kernel, resulting in a feature map.

2. Pooling layer aims to decrease the dimensions of the image, while still retaining critical data to improve the efficiency of the model and reduce overfitting. Commonly used max-pooling retains the maximal value in a certain window, or average pooling that saves the average value instead.

3. Fully connected layer has the task of performing classification based on the resulting features extracted by previous layers.

As mentioned in the previous sections, a machine learning model undergoes a process of training, and CNNs are no different, afterwards the model can be used to identify faces in unfamiliar images.

3.1 Technical Details

For face recognition, we use the built-in solution of YuNet provided by OpenCV as a primary solution. There are many different models for face detection, some of them more accurate than YuNet[16], i.e. Retinaface-resnet50, however taking into consideration that the solution must detect faces and ages in real-time, the chosen model needs to be quite efficient. YuNet provides a great balance between efficiency and accuracy. Optionally, we also offer the OpenCV Haar Cascade model and a model that has been trained on a dataset extracted from a couple of videos and labeled by hand. The dataset has been augmented by random rotations, changes of brightness or cropping, and split into test, train, and validation sets.

3.2 Performance

The machine learning model's performance can be measured in terms of the following[17]:

1. precision - the measure of true positives over total detections

$$\frac{TP}{TP + FP}$$

2. recall - shows whether a model can find all objects of a target class

$$\frac{TP}{TP + FN}$$

3. F1 score - it is an overall performance indicator for binary classification models, showing how well the model recognizes true positives while minimizing false negatives

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

4. accuracy - evaluates how well a model detects true positives and negatives

$$\frac{TP + TN}{TP + TN + FP + FN}$$

5. errors - one can measure errors in terms of root mean squared error or mean absolute percentage error

Face recognition is often tested on the Labelled Faces in the Wild dataset[18]. The chosen face recognition model (YuNet) has achieved a remarkable accuracy of 99.60%[19]. On the UTKFace dataset, the face recognition model achieved 99.66% accuracy. We also offer the Haar Cascade classifier that achieves an accuracy of 91.21% on the UTKFace dataset. While this method may not match the accuracy of state-of-the-art deep learning models, its inclusion allows for a comprehensive performance evaluation across different methodologies. We have also trained our model that achieves 99.16% accuracy on UTKFace but is slower in processing speed.

4 Age Recognition

Age Recognition has a lot in common with Face Recognition in terms of used technology and processes[20]. Here, we also use a Convolutional Neural Network (CNN) that has been explained in previous sections. The model is given a preprocessed augmented dataset on which it learns to extract desired features and classify the result.

To better understand what is happening during the training of a model it's key to understand neural networks in general. There are architectures simpler than that of a CNN, such as a multilayer perception inspired by the brain. They consist of a number of entities resembling neurons that are connected in particular ways. Let's take a dataset consisting of some fixed-sized grayscale images of handwritten digits as an example. The constructed neural network consists of several layers. As a first layer let us represent each pixel in an image as a separate artificial neuron. Each neuron is assigned a value, also referred to as its activation, upon input, the value is the grayscale value of a corresponding pixel. The last layer represents the output, in the case of numbers, the layer has 10 artificial neurons in it. Values in the last layer translate to how confident the neuronal network is that a neuron representing a specific class, such as a certain digit, is the correct answer. In between there are several hidden layers of custom-size. Data is passed from one layer to the next, certain patterns of activation cause another pattern in the next layer, and so on. The process of one layer influencing the next consists of computing the values of neurons in the next layer, which is done by taking a weighted sum of neurons from the previous layer plus some bias, where weights are the edge weights between neurons from the previous layer and a particular neuron from the next layer. However, in this case, we would like the result of the weighted sum to be in the range of 0 to 1 to indicate the level of confidence, this can be achieved by applying some activation function, i.e. sigmoid function, that produces such a result. We end up with quite a lot of parameters that influence how the network behaves. To choose the proper values for those weights and biases, the network needs to be trained on a good-quality dataset. The training process can start with a random assignment of

weights and biases in the network, knowing what the correct answer is for each data point, we can define a cost function that indicates how far off the network's answer is from the correct answer. The task becomes to choose such weights and biases so that the cost is minimized. The cost function can be defined as the squared differences of proper output neuron values and resulting values, that way we can measure the average cost and judge the network's effectiveness. To find the function's local minimum we can make use of a negative gradient then the weights and biases can be adjusted given the gradient with the use of backpropagation which is an algorithm for determining how weights and biases should change given training data point to cause the biggest decrease to the cost. However, doing backpropagation for all available data points to get the average change for each weight and bias is considerably slow, so training data can be divided into smaller batches to speed up the process [21][22].

It's important to clarify how we define a class. A class can be specified in different ways, i.e. a class could be a range of ages such as 21-25, 26-30, and so on. Alternatively, the classes could be divided based on certain features, we could analyze when on average the process of aging starts being visible in the form of wrinkles, more pronounced facial features, and similar and then divide the classes accordingly. Other differentiators could be found such as gray hair, or facial features development around the age of 3. In this case, we decided that each age corresponds to a different class.

To measure the age recognition model's effectiveness we focus on the Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) that more accurately represent the model's performance in this context.

Age recognition is faced with a couple of obstacles that should be taken into consideration when deciding which dataset to use, during the assessment of the model's performance, and when deciding in what conditions the model will be used. The aging process is complex and varies among individuals, different factors such as genetics, lifestyle, or not using enough sunscreen contribute to the visible signs of aging, which make a person look older or younger than the actual age in common understanding. Other obstacles include face modifications, certain health conditions, ethnicity, or light intensity that might make certain features more pronounced.

4.1 Technical Details

The code needed for the creation, training, and execution of the age recognition model was all written in Python with the use of libraries such as Tensorflow, Keras, and OpenCV. The Convolutional Neural Network built has the following architecture. Four pairs of convolutional layers and max pooling layers followed by flattening layers and three fully connected layers. The purpose of the dropout layer is to prevent overfitting, which is a phenomenon when a trained model is accurate for the training data but not for new data. It's crucial to choose a reasonable number of epochs. An epoch is one pass of the training dataset through the model training. Too many epochs might result in overfitting, whereas not enough epochs cause underfitting resulting in low accuracy. A correct number of epochs can be chosen by trial and error, in this instance the chosen number is 10, we started a bit higher with 20 epochs, however, this resulted in overfitting. Keras provides different optimizers, we have chosen the Adaptive Moment Estimation (Adam) optimizer, which is a stochastic gradient descent based

method. Optimizers are methods that aim to reduce the losses, to improve the accuracy by changing certain properties of the network such as weights. Finally, rectifier (ReLU) is the activation function of choice, it can be expressed as $f(x) = max(0, x)$.

4.2 Performance

After the preparation and training of the age recognition model, it's time to check the quality of the results. The primary measures used are the Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). The first model that seemed to achieve the best results was trained on the whole UTK Face, therefore to get objective results, we had to use another dataset to perform tests. We have chosen a dataset from kaggle.com - Age Recognition Dataset[23]. It suited our needs well, as it had a similar structure to the UTK in terms of labeled information so that we could use it almost instantly to retrieve some useful testing information. Overall the performance resulted in RMSE: 9.89 and MAPE: 24.50%, moreover model achieved in particular groups of the Age Recognition Dataset[23] following results:

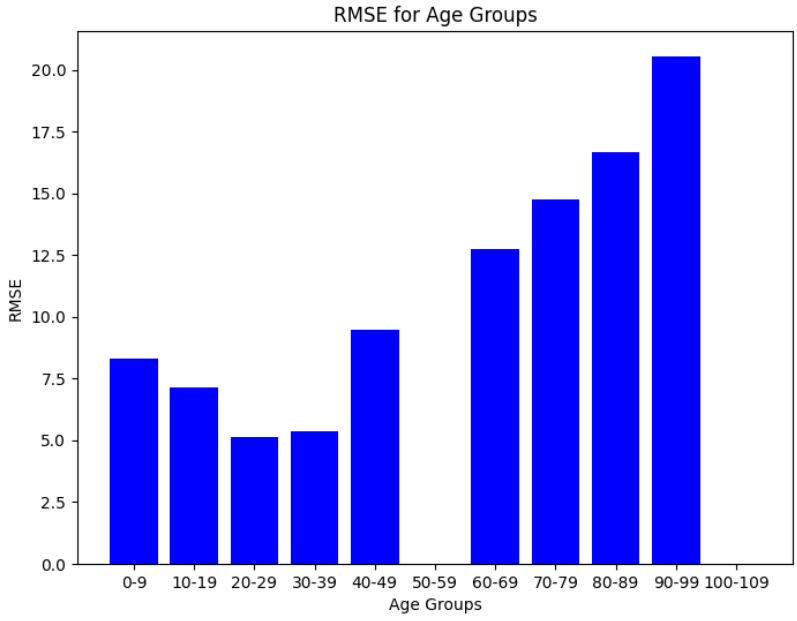

Figure 15: RMSE plot

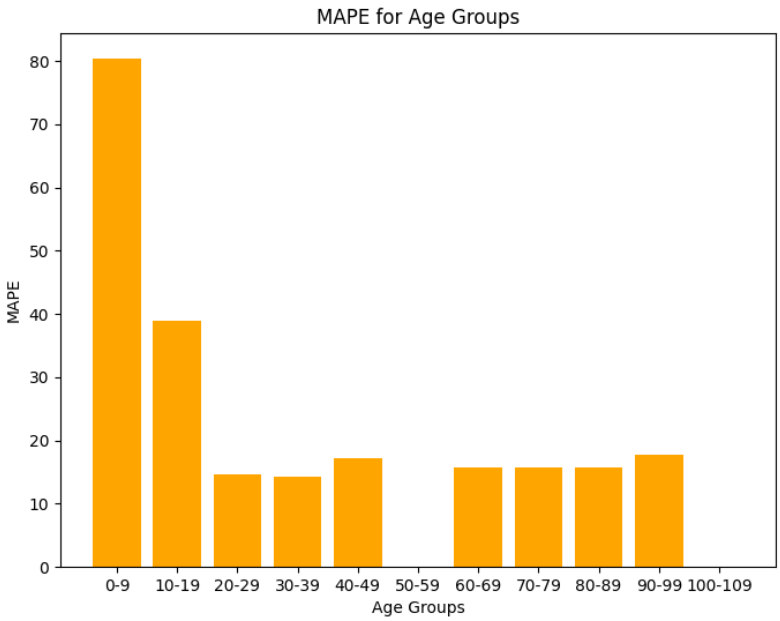

Figure 16: MAPE plot

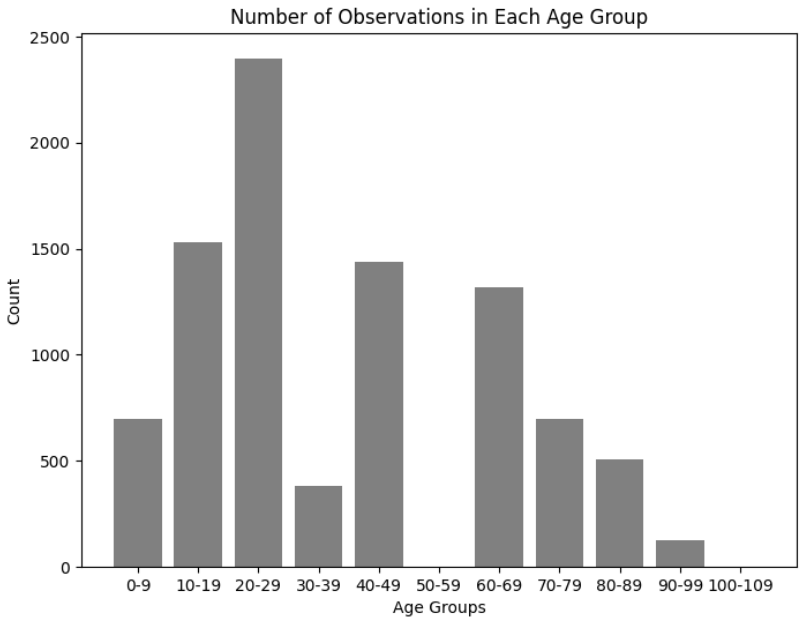

Figure 17: Age groups count plot

The process of training can be observed by looking at the graph of losses. Several outcomes are possible here, the graph could indicate underfitting, overfitting, or a good fit.

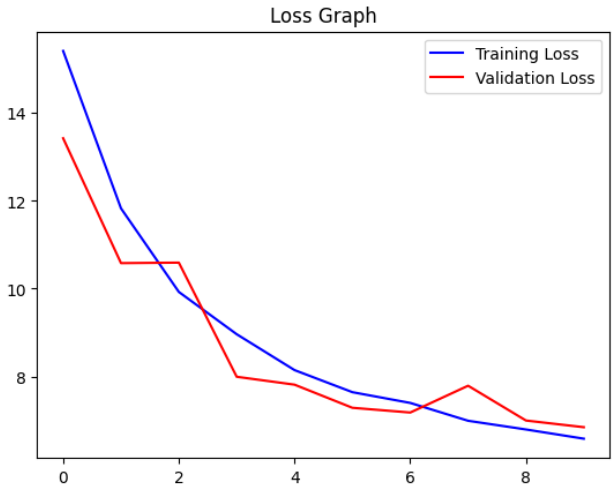

Figure 18: Loss graph

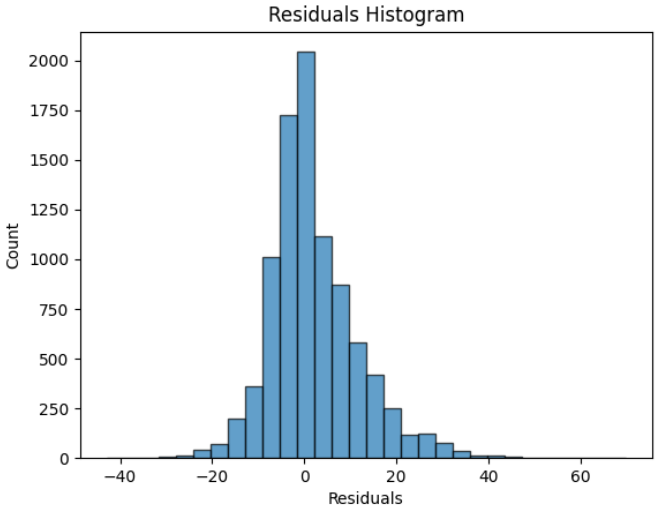

Figure 19: Loss age groups plot

The results follow a normal distribution and indicate a good fit, which means that the model should be able to accurately detect human age on unfamiliar data. However, the training dataset did not include some of the age groups. The next step was to train another version of the model on the subset of the Age Recognition Dataset[23] and then later on, test it on the same dataset, but a different subset. Below, there are presented results of the other version of the model:

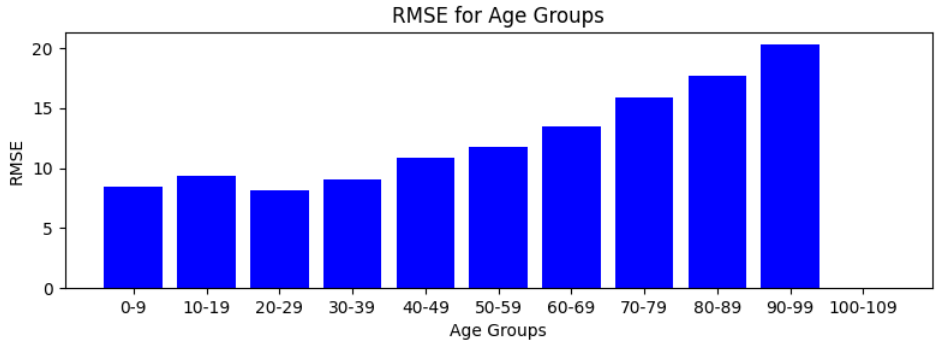

Figure 20: RMSE plot

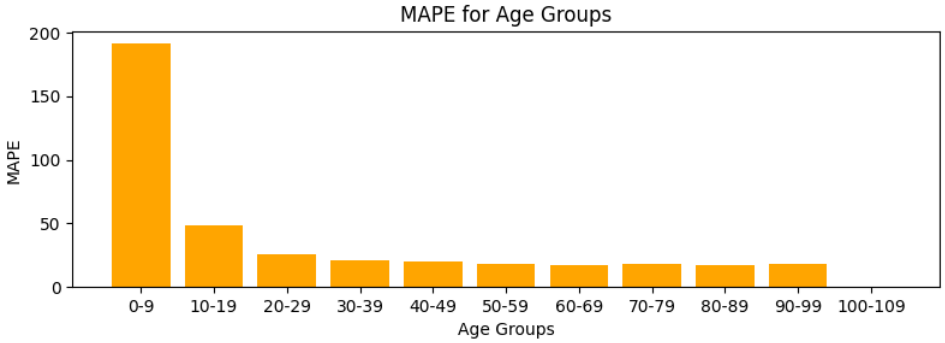

Figure 21: MAPE plot

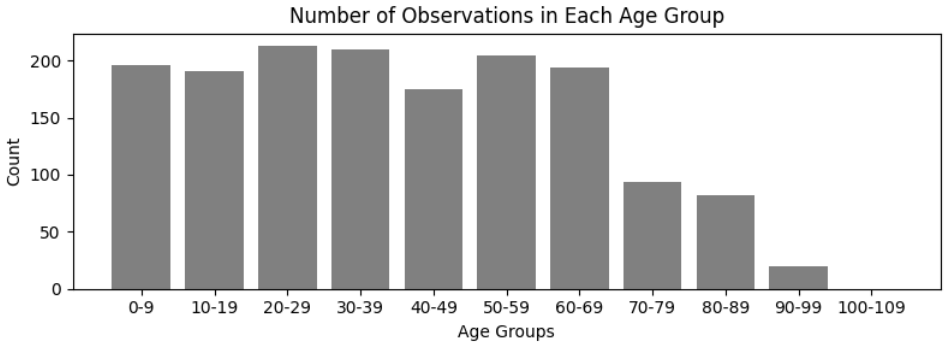

Figure 22: Age groups count plot

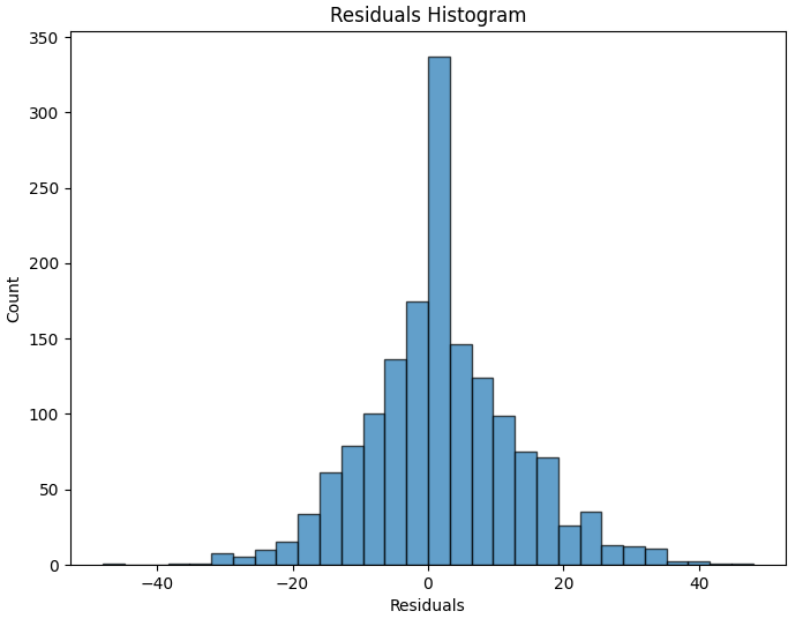

Figure 23: Loss age groups plot

The residual histogram follows the normal distribution. However, the newer version of the trained model seems to achieve worse results, where RMSE amounted to 11.37 and MAPE resulted in 44.62%. We didn't stop there, with subsequent trial and error to

improve the model but the results were similar. The model presented at the start of this section achieved the best results.

5 Case Studies

To test the model's effectiveness in different conditions a couple of case studies have been prepared. First, we use random erasing to make the model's job more difficult, it simulates situations where a part of the face is covered. For each image, we cover a part of the image ranging from 0.2 to 0.5 of the image's size. As images, we have taken a subset of the UTK Face dataset.

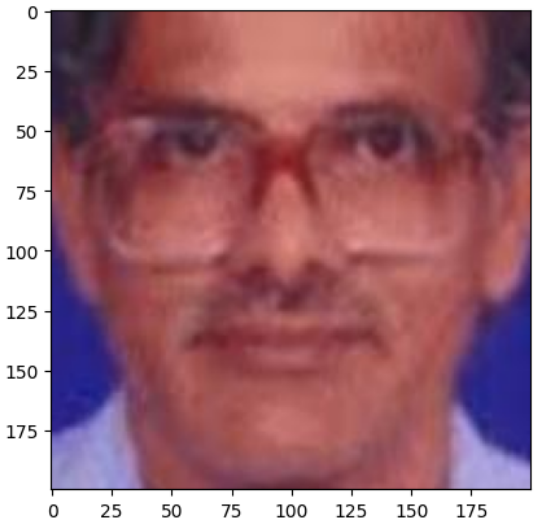

Figure 24: Original image

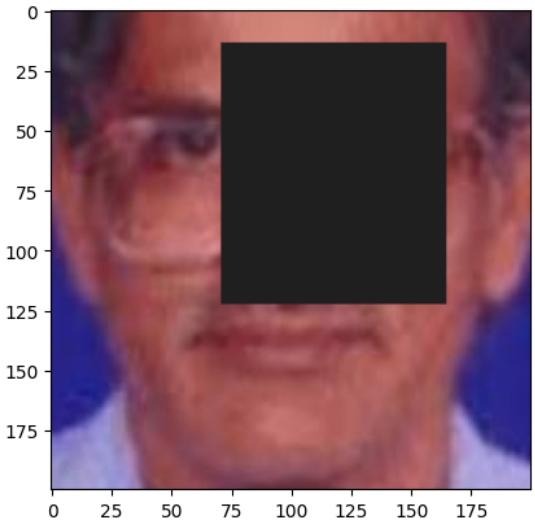

Figure 25: Random erasing

The resulting RMSE is 7.87 with 22.24% MAPE. Next we randomly significantly increase or decrease brightness to simulate more difficult light conditions.

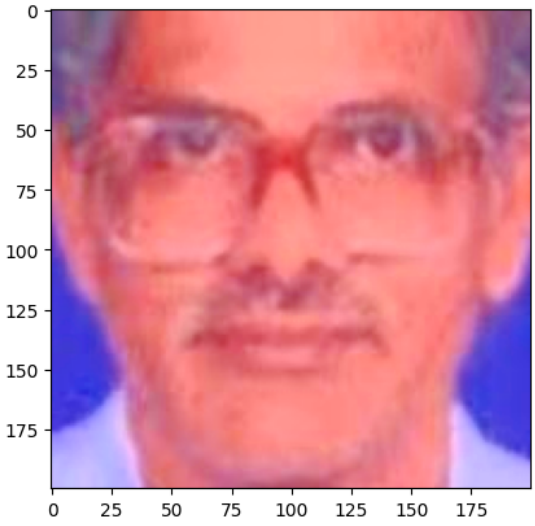

Figure 26: Increased brightness

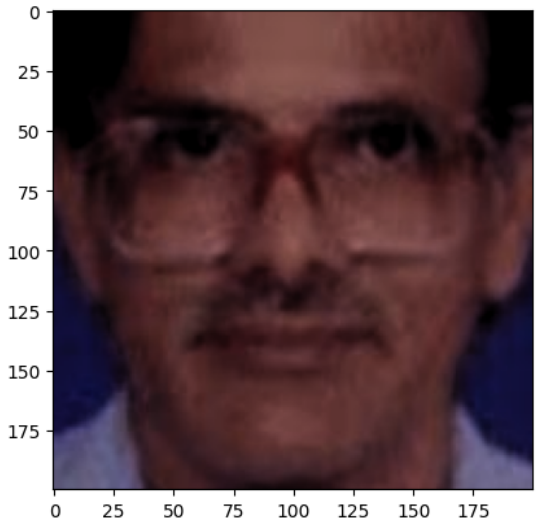

Figure 27: Lowered brigthness

RMSE for brighter images results in 6.30 with 17.11% MAPE, whereas for darker images in 6.558 with 16.13% MAPE. These tests show that, in general, the model's accuracy doesn't significantly suffer from difficult conditions therefore it can be used in real-world scenarios.

6 Conclusions

The primary goal of recognizing human age based on input data (pictures, videos, live webcam) has been achieved. The model has been tested at multiple stages of development and fine-tuned to achieve optimal performance and high accuracy. Moreover, the final resulting model has been tested in various metrics such as accuracy, Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) yielding a satisfactory result. Face detection via YuNet achieves 99.60% accuracy and the age recognition model has a 9.89 RMSE and 24.50% MAPE. Presented case studies show the model's effectiveness even in non-ideal conditions. Such scenarios present real-world applicability of the constructed model. Looking forward, the model's performance could be refined over an extended time resulting in improved metrics. Lastly, the application could be adopted as a free alternative to the before-mentioned existing solutions.

Appendix A - User Manual

Requirements

To run the program Python must be installed on the machine. Preferably version 3.11.0, the program has not been tested on other versions of Python. Required libraries are

tkinter, tensorflow, numpy, cv2 (OpenCV), PIL (Pillow), os, shutil, copy, argparse, they can be installed by executing:

```
pip install -r requirements.txt
```

To execute commands present in Jupyter notebooks, Jupyter Notebook should be installed together with the necessary libraries present at the beginning of each notebook, then in the parent directory Jupyter can be started and code present in notebooks can be executed.

Usage

The program can be started by (assuming one is in the project's directory):

```
python main.py
```

Then an interactive GUI will be displayed. The user can choose one of three available face detection models and start the process of age detection from a webcam, video file, or directory with images. After using one of those features presented results are saved to the directory "frame_photos" in the solution folder. After each usage of the face and age detection folder is cleared.

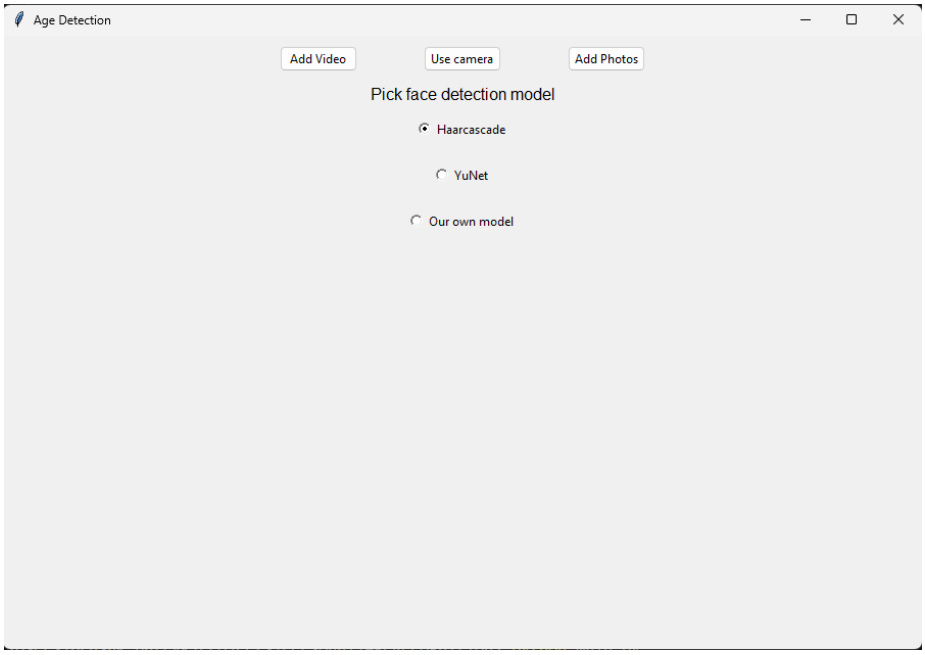

Figure 28: Application GUI

List of Figures

List of Tables

References

[1] *About Face ID advanced technology*. URL: `https://support.apple.com/en-us/102381`. (accessed: 24.01.2024).

[2] *Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks*. URL: `https://arxiv.org/abs/1807.05720`. (accessed: 24.01.2024).

[3] *How old do I look - Age app*. URL: https://play.google.com/store/apps/details?id=io.edia.agerecognitionapp&hl=en_US. (accessed: 24.01.2024).

[4] *How old am I? Face age app*. URL: https://play.google.com/store/apps/details?id=com.testa.agebot&hl=en&gl=US. (accessed: 24.01.2024).

[5] *Yoti - facial age estimation*. URL: https://www.yoti.com/business/facial-age-estimation/. (accessed: 24.01.2024).

[6] *Everypixel Labs - age recognition*. URL: https://labs.everypixel.com/age-recognition. (accessed: 24.01.2024).

[7] Rafiuddin Khan. "Importance of Datasets in Machine Learning and AI Research". In: *DataToBiz* (). URL: https://www.datatobiz.com/blog/datasets-in-machine-learning/#:~:text=Dataset%20is%20a%20collection%20of,Image%20or%20video%20classification.

[8] *Exploratory Data Analysis*. URL: https://www.epa.gov/caddis-vol4/exploratory-data-analysis#:~:text=Exploratory%20Data%20Analysis%20(EDA)%20is,step%20in%20any%20data%20analysis.. (accessed: 24.01.2024).

[9] *UTKFace: Large Scale Face Dataset*. URL: https://susanqq.github.io/UTKFace/. (accessed: 24.01.2024).

[10] *Faces: Age Detection Dataset*. URL: https://www.kaggle.com/datasets/arashnic/faces-age-detection-dataset. (accessed: 24.01.2024).

[11] *Age Detection - Human Faces*. URL: https://www.kaggle.com/datasets/trainingdatapro/age-detection-human-faces-18-60-years. (accessed: 24.01.2024).

[12] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, Eero P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE TRANSACTIONS ON IMAGE PROCESSING* 13.4 (2004), pp. 600–612. URL: https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf.

[13] *Histogram of Oriented Gradients*. URL: https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html. (accessed: 24.01.2024).

[14] *NYPD Questions and Answers Facial Recognition*. URL: https://www.nyc.gov/site/nypd/about/about-nypd/equipment-tech/facial-recognition.page. (accessed: 24.01.2024).

[15] Stanisław Osowski, Krzysztof Siwek. "CNN application in face recognition". In: *PRZEGLĄD ELEKTROTECHNICZNY, Warsaw University of Technology, Military University of Technology* (2020), pp. 142–145. URL: http://pe.org.pl/articles/2020/3/31.pdf.

[16] *What is Face Detection? Ultimate Guide 2023 + Model Comparison*. URL: https://learnopencv.com/what-is-face-detection-the-ultimate-guide/#Comparison-of-Face-Detectors. (accessed: 24.01.2024).

[17] *Classification Performance*. URL: https://c3.ai/introduction-what-is-machine-learning/evaluating-model-performance/. (accessed: 24.01.2024).

[18] *Labelled Faces in the Wild dataset*. URL: https://vis-www.cs.umass.edu/lfw/. (accessed: 24.01.2024).

[19] *DNN-based Face Detection And Recognition*. URL: https://docs.opencv.org/4.x/d0/dd4/tutorial_dnn_face.html. (accessed: 24.01.2024).

[20] Alice Othmani, Abdul Rahman Taleb, Hazem Abdelkawy, Abdenour Hadid. "Age estimation from faces using deep learning: A comparative analysis". In: *Computer Vision and Image Understanding* 196 (2020). URL: https://www.sciencedirect.com/science/article/abs/pii/S1077314220300424.

[21] 3Blue1Brown Grant Sanderson. *But what is a neural network? | Chapter 1, Deep learning*. URL: https://www.youtube.com/watch?v=aircAruvnKk&ab_channel=3Blue1Brown. (accessed: 24.01.2024).

[22] *What is a neural network?* URL: https://www.ibm.com/topics/neural-networks. (accessed: 24.01.2024).

[23] Rashik Rahman. *Age recognition dataset*. URL: https://www.kaggle.com/datasets/rashikrahmanpritom/age-recognition-dataset?fbclid=IwAR1B_FO0IqywAr3KatYXpzgMTizXxqtIRL2sUx05gZ-C0Hvdz6w6SY3AEDc. (accessed: 24.01.2024).