Comparison Between Two Biometric Architectures for Gender and Age Classification

Biometric systems

Group members:
Federici Tommaso 2214368, Fornetti Lucia 2214370
federici.2214368@studenti.uniroma1.it,
fornetti.2214370@studenti.uniroma1.it

professor: De Marsico Maria

February 10, 2025

Abstract

Although age and gender classification are not tasks strictly related to biometric systems, they are still relevant in the field of computer vision and find numerous applications. In particular, for our system, we envisioned applications in the fields of human-computer interaction, marketing and customer experience, or more simply demographic analysis.

Our approach focused on the extraction of visual biomarkers of aging, such as wrinkles or skin texture, to train the network to predict a person's age as accurately as possible. The age task is particularly challenging so we reduced the problem to a classification task, dividing the ages into 8 classes. In this report, we compare two different architectures for the tasks of gender and age classification. The first is a multitask model developed by us using the fine-tuning technique on a neural network already trained on face recognition (FaceNet). The second is the combination of two monotask models trained for gender and age detection, both trained using the caffeNet network as a base. The dataset used for training our model is the UTKFace dataset, which contains images of faces with associated age, gender, and ethnicity labels. This dataset is widely used for training and evaluating face-related tasks due to its diversity and size.

Contents

Tool Used

- **Python**: All the code for this project was written in Python.
- **PyTorch**: An open-source deep learning framework that provides tools for building and training neural networks.
- **UTKFace Dataset**: A large dataset containing over 20,000 facial images labeled with age, gender, and ethnicity, commonly used for facial recognition and age prediction tasks.
- **FaceNet**: A deep learning model used for face recognition and verification, based on a triplet loss function to map faces into a compact Euclidean space.
- **OpenCV**: A popular open-source computer vision library used for real-time image processing, manipulation, and computer vision tasks.
- **CaffeNet**: A deep learning architecture based on the Caffe framework, typically used for image classification and feature extraction tasks.
- **Matplotlib**: A Python plotting library used to create static, interactive, and animated visualizations in data analysis.
- **NumPy**: A fundamental package for numerical computing in Python, providing support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on them.
- **OS**: A Python module providing functions to interact with the operating system, such as file handling, directory management, and process control.
- **scikit-learn (sklearn)**: Used for splitting the dataset into training and validation sets with the `train_test_split` function

Chapter 1

Real world applications

The idea of developing a system for age and gender classification comes from its potential applications across various sectors, such as marketing, user interfaces, and demographic analysis. We envisioned three different scenarios where such a system could be effectively implemented.

1.1 Marketing and Customer Experience

This system could be used in marketing to analyze customer demographics and tailor marketing strategies accordingly. By identifying the age and gender of customers, businesses can create targeted advertising campaigns, promotions, and product offerings that cater to specific demographics. For example, a retail store could use this system to analyze the age and gender of customers entering the store and adjust its product displays, promotions, and marketing messages to better appeal to its target audience. This could help increase customer engagement, drive sales, and improve the overall customer experience.

1.2 Human-Computer Interaction

A second possible scenario we envisioned involves enhancing user experience with device interfaces. A device equipped with this system could adapt its interface to better accommodate the user. For example, a restaurant kiosk that detects an elderly customer could simplify the interface, enlarge text size, and provide more detailed explanations of the available services.

1.3 Demographic Analysis

Lastly, this system could be used for demographic analysis in various settings, such as public spaces, transportation hubs, or events. By analyzing the age and gender distribution of people in these environments, organizations can gain valuable insights into the demographics of their audience. This information can be used for various purposes, such as improving public services, optimizing resource allocation, and enhancing security measures. For example, in a transportation hub, understanding the age and gender distribution of passengers can help in designing better facilities and services that cater to the needs of different demographic groups. Similarly, at events, organizers can use this information to tailor their offerings and improve the overall experience for attendees.

Chapter 2

First Architecure: Fine-tuning on FaceNet

In this chapter we describe the first architecture made by us, which is a multitask model for gender and age classification.

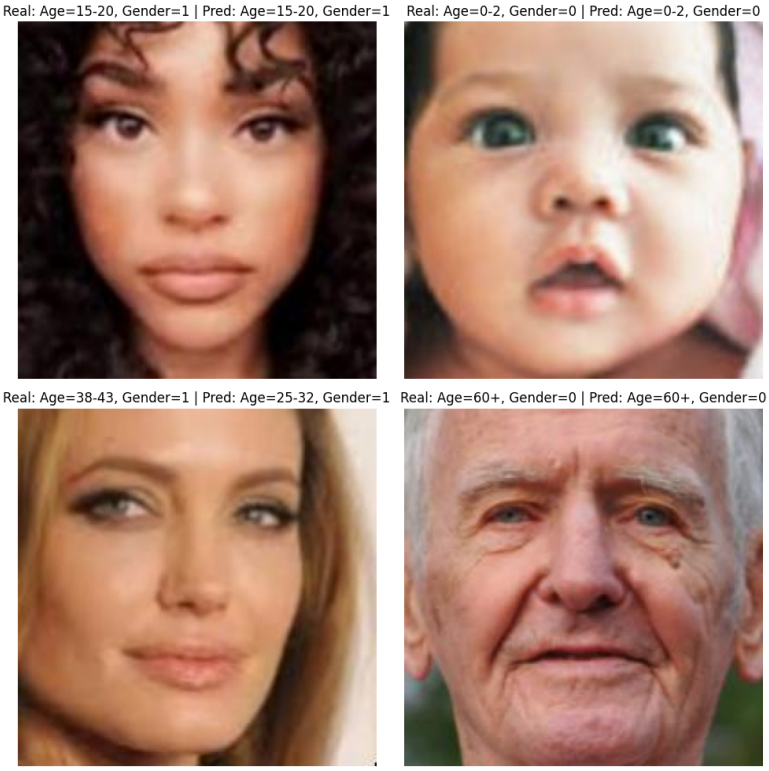

Figure 2.1: Example image first architecture

2.1 FaceNet

The model is based on the FaceNet architecture, developed by Google [1], which is a neural network trained for face classification. FaceNet uses a deep convolutional network to map face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. The network is trained using a triplet loss function, which ensures that the distance between an anchor image and a positive image (same identity) is smaller than the distance between the anchor image and a negative image (different identity) by a margin. [citare paper originale FaceNet]

2.2 Fine-tuning

We used the fine-tuning technique to adapt the network to our tasks. Fine-tuning involves taking a pre-trained network and slightly adjusting its weights to better fit the new task. Specifically, we replaced the final layers of the FaceNet model with new layers suitable for gender and age detection. The new architecture consists of a shared convolutional base followed by two separate branches: one for gender classification and one for age regression. We also retrained the last linear layer of the shared base to better fit the new tasks.

2.3 Dataset

The dataset used for training and evaluating our model is the UTKFace dataset: is a large-scale face dataset with long age span (range from 0 to 116 years old). The dataset consists of over 20,000 face images with annotations of age, gender, and ethnicity. The images cover large variation in pose, facial expression, illumination, occlusion, resolution, etc. [kaggle link to UTKFace]. The dataset can be downloaded from the Kaggle website. In particular every image is labeled in this way: "age_gender_race_date&time.jpg", where 'age' is an integer between 0 and 116 and 'gender' is a binary value (0 for 'Male' and 1 for 'Female), that are the only two values that we are interested in. Each image is a 200x200 pixel RGB image containing a face. To better tailor the dataset to our task, we decided to use only the ages within certain ranges, reducing our dataset to slightly under 17,000 face images; in particular, we used the following ranges: '0-2', '4-6', '8-13', '15-20', '25-32', '38-43', '48-53', '60+'. Those are common age ranges used in the literature for age detection task. The dataset was divided into training, validation, and test sets with a ratio of 70%, 10%, and 20% respectively.

2.4 Preprocessing of the images

While attempting an initial simple training of the model and reviewing the literature, we realized that the age prediction task is the more challenging of the two. To better train the network, we preprocessed the images to extract the visual biomarkes of aging, such as wrinkles or skin texture. The preprocessing steps included:

1. Gray scale conversion: to reduce the computational cost of the model. In particular we used the OpenCV function cv2.imread() using the flag "IMREAD_GRAYSCALE".

2. Gaussian blur: to reduce possible noise in the image. In particular we used the OpenCV function cv2.GaussianBlur() with a kernel size of 5x5.

3. CLAHE (Contrast Limited Adaptive Histogram Equalization): to improve the contrast of the image. In particular we used the OpenCV function cv2.createCLAHE().

4. Gabor Filter: to extract texture features from the image. In particular we used the OpenCV function cv2.filter2D() with a Gabor kernel.

Those choices were made based on the idea that the biomarkers of aging can be better extracted using the above techniques. The RGB color space was not necessary for our task, so we decided to reduce the images to grayscale. Due to the possible presence of noise in the images, we decided to apply a Gaussian blur to reduce it. The CLAHE technique allows us to improve the quality of the examined photos in terms of illumination and contrast, addressing potential issues related to photos taken under adverse lighting conditions. The Gabor filter is a linear filter used for edge detection and it's particularly useful for texture analysis. We used it to extract texture features from the images, which can be useful for age detection tasks. In Figure 2.2 we can see an example of the preprocessing steps applied to an image.

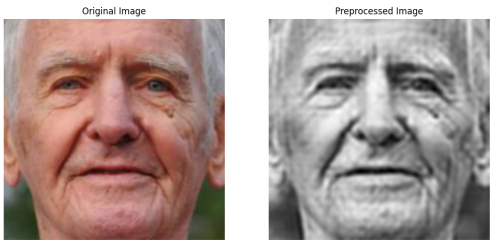

Figure 2.2: example of preprocessing steps applied to an image

2.5 Model Loading and Weight Freezing

The base model used in this project is the *InceptionResnetV1* model, pretrained on the VGGFace2 dataset. This model is a variant of the original *FaceNet* architecture, designed to improve efficiency and performance by combining the strengths of the *Inception* and *ResNet* architectures. The model is loaded using the `facenet_pytorch` library, which provides a convenient interface for loading the pretrained version of *InceptionResnetV1*.

To adapt the model to the multitask classification problem (gender and age detection), the original output layer is replaced with two separate layers: one for gender classification and another for age classification. The *gender_head* layer (with `out_features = 2`) produces the gender output, while the *age_head* layer (with `out_features = 8`) is responsible for predicting the age range. These layers are initialized with random weights, and their parameters are fine-tuned during training.

Additionally, we freeze the weights of all pre-trained layers in the base model to retain the knowledge learned from the original dataset. This is done by setting the `requires_grad` attribute of all parameters in the base model to `False`, preventing them from being updated during training.

Only the weights in the last fully connected layer (the `last_linear` layer), which are responsible for gender and age classification tasks, are unfrozen. The `requires_grad` attribute for these parameters is set to `True`, ensuring that they are updated during training to adapt to the specific task at hand.

2.6 Training

Before starting the training phase, we used the `os` library to read the files inside the folder containing the images for training. We saved three different arrays: the paths to the images and the labels for age and gender. In our model, each age range corresponds to a class represented by an integer between 0 and 8. Therefore, we implemented a function `"age_to_class"` that takes an age as input and returns the integer corresponding to the class of the given age (e.g. $9 \rightarrow$ '8-13' $\rightarrow 2$).

After preparing the dataset, we were able to start the training phase. The model was trained using the PyTorch library. Below, we outline the key decisions and techniques used in the implementation.

2.6.1 Loss function and optimizer

We employed CrossEntropyLoss for both the gender and age classification tasks. For age classification, weighted cross-entropy was used to address class imbalance, where weights were computed based on the inverse frequency of each age class in the training set. The optimizer used was Adam, with a learning rate of 0.0001. To reduce the learning rate when the validation loss plateaued, helping the model converge better, we used a learning rate scheduler, in particular the ReduceLROnPlateau scheduler from the PyTorch library.

2.6.2 Data augmentation and preprocessing

As we saw in the previous chapter, we applied some preprocessing techniques to the images to extract the visual biomarkers of aging. We also applied data augmentation techniques to the images to increase the diversity of the training set. In particular, we used random horizontal flips, random rotations, and random crops. All this transformations were applied using the PyTorch library.

2.6.3 Training process

The model was trained for 30 epochs, but the training was stopped early when the validation loss stopped decreasing. The early stopping technique was implemented with a patience parameter set to 5. The last model saved, the one with the lowest validation loss, was the one after the 19th epoch. In Figure 2.3 we can see the training and validation loss curve during the training process.

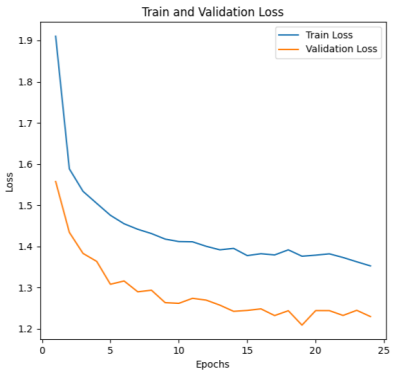

Figure 2.3: Training and validation loss curve

Chapter 3

Second Architecture: Fine-tuning on CaffeNet

The second system is based on a project found on GitHub [2], which utilizes two monotask systems based on CaffeNet and a facial recognition system. Our task was to adapt this system for testing on the UTKFace dataset in order to compare its performance with our model.

3.1 CaffeNet

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR)/The Berkeley Vision and Learning Center (BVLC) and community contributors [3]. CaffeNet is a deep learning architecture based on the Caffe framework, typically used for image classification and feature extraction tasks.CaffeNet consists of five convolutional layers, three max-pooling layers, and three fully connected layers.

3.2 Architecture adaptation

From the original project, we extracted the two models for age and gender detection. These are monotask models, fine-tuned on CaffeNet. The face detector was not required for our purpose, so we removed it and adapted the rest of the code to ensure that the models could directly accept images from the UTKFace dataset as input. To load these models is used the OpenCV module.

To adapt the dataset to the model, we implemented a function called `"map_age_to_range"` that takes an age as input and converts it into an age

range format accepted by the model for age detection (e.g. 9 → '(8-13)'). A second function, `"map_range_to_class"`, maps these age ranges to the corresponding classes used in our model (e.g. '(8-13)' → 2). This allows us to modify the code initially used for testing the first model, which will be explained in detail in chapter 4, so that it could be used for testing this second model as well.

Chapter 4

Testing

In this chapter, we describe the testing phase of the two models. We used the UTKFace dataset to evaluate the performance of the models on the tasks. As we saw in the previous chapters, the dataset was divided into training, validation, and test sets. The test set (20% of the original dataset) was used to evaluate the performance of the models on unseen data. The metrics used for evaluation were:

- **Accuracy**: the percentage of correctly classified samples.
- **Precision**: the ratio of correctly predicted positive observations to the total predicted positives.
- **Recall**: the ratio of correctly predicted positive observations to the all observations in actual class.
- **F-score**: the weighted average of Precision and Recall.
- **Miss Rate (False Negative Rate)**: the ratio of false negative predictions to the actual positive observations.
- **Fall-Out (False Positive Rate)**: the ratio of false positive predictions to the actual negative observations.
- **ROC curve (for gender detection)**: a graphical representation of the true positive rate against the false positive rate.
- **Mean Error (for age detection)**: the mean absolute error between the predicted and actual age classes.
- **Standard Deviation (for age detection)**: the standard deviation of the errors between the predicted and actual age classes.

- **Confusion Matrix**: a matrix showing the number of true positive, true negative, false positive, and false negative predictions.

The metrics were calculated for both the architectures and the results can be found in chapter 5. We first initialized the variables needed to calculate the metrics and then iterated over the test set, feeding the images to the models and collecting the predictions. The predictions were then compared with the actual labels to calculate the metrics.

4.1 Gender metrics

To evaluate the performance of the models for the gender task we used the metrics described above, only except for the mean error and standard deviation, which are not applicable for binary classification tasks. The variables used to calculate the metrics are related to the male gender and were the following:

- `true_positive`: the number of correctly predicted
- `true_negative`: the number of correctly predicted
- `false_positive`: the number of incorrectly predicted
- `false_negative`: the number of incorrectly predicted

Obviuosly, the variables related to the female gender are the same, but with the opposite meaning (e.g. `true_positive_female = true_negative_male`). The variable `correct_gender (= TP + TN)` is the number of correctly predicted samples.

To compute the ROC curve first of all we converted the predictions to probabilities using the `softmax` function. Then we used the `roc_curve` function from the `sklearn.metrics` module, which returns the false positive rate, true positive rate, and thresholds. We then used the `auc` function to calculate the area under the curve.

4.2 Age metrics

To evaluate the performance of the models for the age task we used the metrics described above, including the mean error and standard deviation but not the ROC curve. The variables used to calculate the metrics are related to the age classes and were the following:

- `true_positive`: an array of size 8 containing the number of correctly predicted samples for each age class.
- `true_negative`: an array of size 8 containing the number of correctly predicted samples for each age class.
- `false_positive`: an array of size 8 containing the number of incorrectly predicted samples for each age class.
- `false_negative`: an array of size 8 containing the number of incorrectly predicted samples for each age class.
- `error_distances`: an array containing the absolute error between the predicted and actual age classes for each sample.

To calculate the mean error and standard deviation, we used the `mean` and `std` functions from the `numpy` library, respectively.

Chapter 5

Results

In this chapter, we present the results of the testing phase for the two models.

5.1 Gender results

	Architecture1	**Architecture2**
Accuracy	87.35%	82.83%

Table 5.1: Accuracy Comparison between the two architectures

	Arch1		**Arch2**	
	Male	**Female**	**Male**	**Female**
Precision	89.03%	85.61%	81.31%	84.80%
Recall	86.54%	88.25%	87.34%	77.88%
F-Score	87.76%	86.91%	84.21%	81.19%
Miss Rate (FNR)	13.46%	13.76%	12.66%	22.12%
Fall-Out (FPR)	11.75%	11.58%	22.12%	12.66%

Table 5.2: Gender Performance Metrics for Architectures 1 and 2

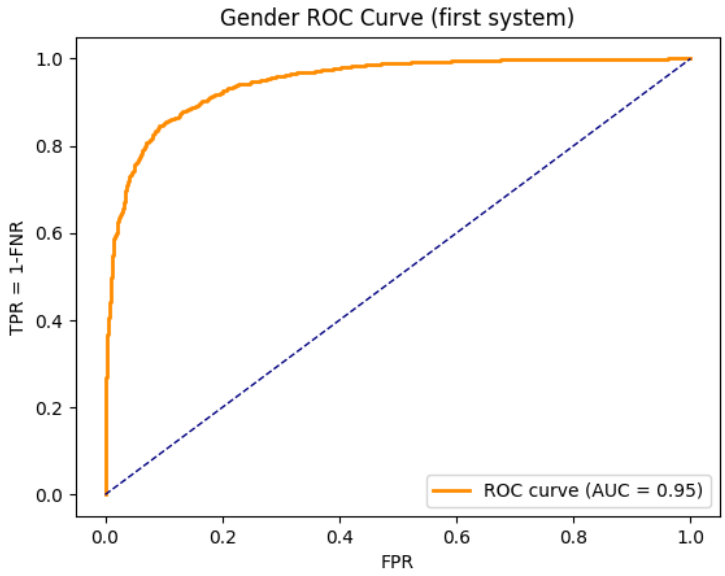

Figure 5.1: ROC curve for Architecture 1

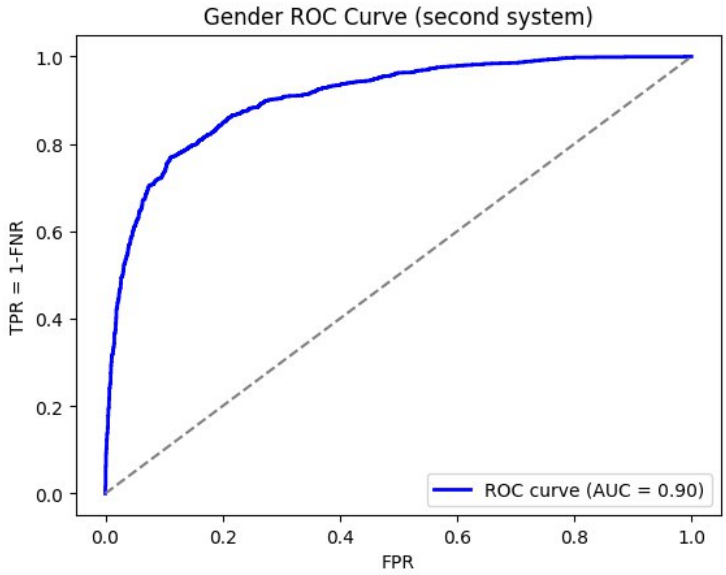

Figure 5.2: ROC curve for Architecture 2

5.2 Age results

	Architecture1	**Architecture2**
Accuracy	61.81%	44.46%
Mean error	0.48	1.04
Standard deviation	0.74	1.32

Table 5.3: Performance Comparison between the two architectures

	'0-2'	**'4-6'**	**'8-13'**	**'15-20'**	**'25-32'**	**'38-43'**	**'48-53'**	**'60+'**
Precision	88.92%	58.56%	71.50%	59.73%	55.45%	42.46%	51.87%	78.86%
Recall	90.31%	47.79%	58.72%	27.16%	86.16%	32.69%	37.09%	83.98%
F-Score	89.61%	52.63%	64.48%	37.34%	67.48%	36.94%	43.25%	81.34%
Miss Rate (FNR)	9.69%	52.21%	41.28%	72.84%	13.84%	67.31%	62.91%	16.02%
Fall-Out (FPR)	1.19%	1.43%	1.77%	3.11%	25.60%	7.16%	3.86%	3.61%

Table 5.4: Age Performance Metrics for Architecture 1

	'0-2'	**'4-6'**	**'8-13'**	**'15-20'**	**'25-32'**	**'38-43'**	**'48-53'**	**'60+'**
Precision	71.69%	49.55%	37.82%	12.67%	63.65%	21.23%	14.11%	17.89%
Recall	75.65%	18.58%	15.67%	16.37%	61.88%	20.88%	16.92%	92.63%
F-Score	73.62%	27.03%	22.16%	14.28%	62.75%	21.05%	15.39%	29.99%
Miss Rate (FNR)	24.35%	81.42%	84.33%	83.63%	38.12%	79.12%	83.08%	7.37%
Fall-Out (FPR)	3.03%	1.84%	4.17%	6.08%	26.83%	9.46%	6.59%	12.43%

Table 5.5: Age Performance Metrics for Architecture 2

Figure 5.3: Distribution of errors for Architecture 1

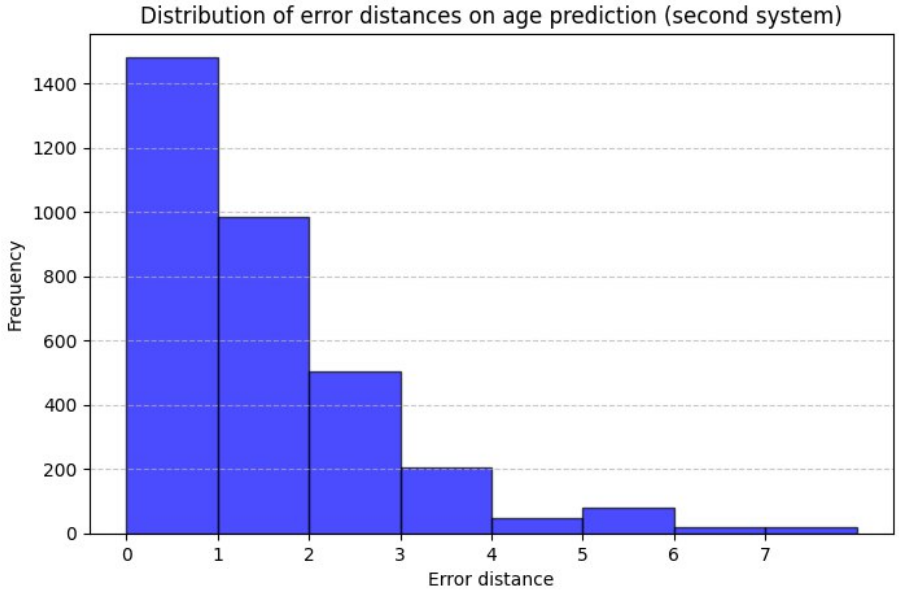

Figure 5.4: Distribution of errors for Architecture 2

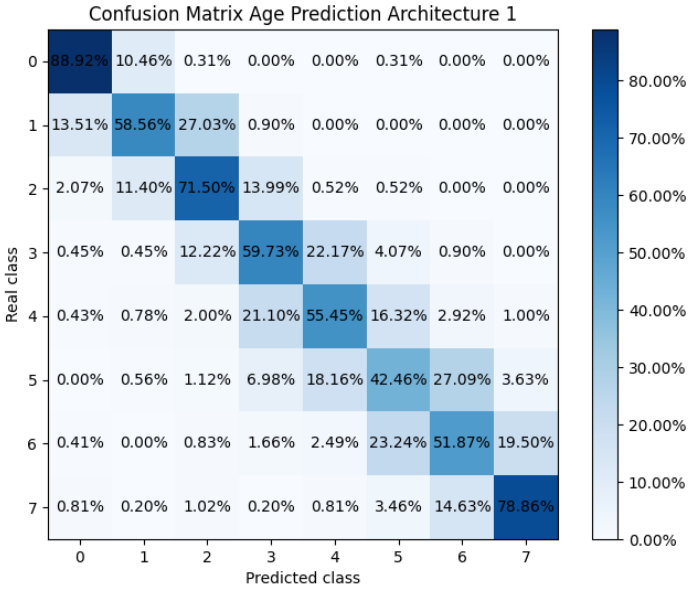

Figure 5.5: Confusion Matrix for Architecture 1

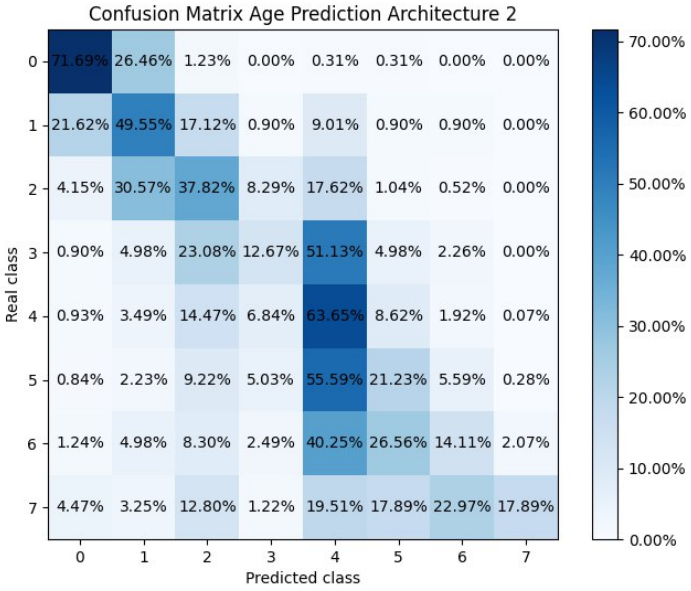

Figure 5.6: Confusion Matrix for Architecture 2

Chapter 6

Conclusions

In this report, we explored and compared two distinct architectures for the tasks of gender and age detection. The first architecture involved fine-tuning a pre-trained FaceNet model into a multitask system for simultaneous gender and age prediction. The second architecture consisted of two separate monotask models, each fine-tuned on the CaffeNet architecture for either gender or age detection. The testing phase has been carried out on the UTKFace dataset for both of them.

As expected, the gender task proved to be significantly easier than the age task. Both models delivered excellent performance on the gender classification task, although the first model appears to be slightly better (5.1, 5.2).

Both architectures achived good performance on age classification too, but the first model outperformed the second one (5.3, 5.4, 5.5). The results show that the first architecture is more accurate and precise in this task. Low mean error and standard deviation indicate that, even in the case of misclassification, the predicted values do not deviate significantly from the actual ones. In this case, the first system outperforms the second. This better performance on the age task is even more evident when looking at the confusion matrix. For the first architecture (5.5), the highest values are well distributed around the diagonal, showing that the predicted values are rarely far from the true ones. The second architecture also performs well, but still falls short compared to the first (5.6). The first architecture performs very well in the '0-2', '8-13', and '60+' classes, while still achieving good results in the other classes. In contrast, the second architecture delivers its best results in the '0-2' and '25-32' classes, with the '60+' class being among the worst-performing.

This difference in results could be attributed to several factors, two of which are likely:

- Dataset used: The dataset used for testing is the same as the one used for training the first architecture (of course, the testing images were never seen by the model during the training phase).
- Image preprocessing: The first model also includes an image preprocessing step (the same one used during training).

Possible future improvements could involve:

- Further fine-tuning of the models on a new dataset: The models could be further fine-tuned to improve their performance on the tasks and to have a more reliable comparison.
- Add a face detection system: The addition of a face detection system could increase the model's versatility.
- Adapting the model for video streaming: Modifying and optimizing the model to work effectively with real-time video streams, allowing a more efficient analysis of a large number of individuals. This could be useful for applications such as demographic analysis.

Bibliography

[1] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. *FaceNet: A unified embedding for face recognition and clustering*. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2015. pp. 815-823. DOI: 10.1109/CVPR.2015.7298682.

[2] https://github.com/MohammedNayeem9/Age-and-Gender-Detection-using-OpenCV-in-Python

[3] @articlejia2014caffe, Author = Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor, Journal = arXiv preprint arXiv:1408.5093, Title = Caffe: Convolutional Architecture for Fast Feature Embedding, Year = 2014