Predicting Age through Facial Recognition using CNN

Ana Filipa Gonçalves (20222157); Cláudia Gomes (20221615); Frederico Rodrigues (20222159); João Silva (20222190); Mariana Pereira (20221628)
Universidade Nova de Lisboa, Information Management School
Deep Learning Neural Networks - Professors: Mauro Castelli; Mafalda Sá Velho

Abstract

This project focuses on developing a neural network for predicting the age of individuals through facial recognition, treating the task as a regression problem. Leveraging the UTKFace dataset, comprising over 20,000 annotated facial images, a convolutional neural network (CNN) is employed to learn hierarchical representations of facial traits for age prediction. The success is measured by the model's performance on validation and test sets, with Mean Absolute Error (MAE) indicating accuracy. The utility extends to applications in security, retail, healthcare, education, and event management, contributing to advancements in computer vision.

The importance of age detection is emphasized across various sectors, enabling personalized access control, targeted marketing, healthcare assessments, and content restrictions. As technology evolves, age detection through facial scanning becomes integral to diverse domains, necessitating ethical considerations for responsible deployment. The data preprocessing involves creating a standardized dataset, visualizing age distribution, and transforming images to grayscale and a uniform dimension. The CNN model is initially designed with a basic architecture, and subsequent optimization tests are conducted, including adjustments to convolutional layers, dense layers, epochs, and learning rates. The models are evaluated based on Mean Squared Error and MAE, with considerations for overfitting.

Results show that specifying the learning rate positively impacts performance, and adjustments in training parameters enhance the model's capability despite increased complexity. The study concludes with recommendations for future improvements, such as addressing age detection by ranges, employing oversampling or under sampling techniques, testing different models for separate age groups, and exploring alternative optimizers.

1. Introduction

In this project, the objective of our team was to develop a neural network capable of predicting the age of individuals based on facial recognition. The selected UTKFace dataset out as an extensive collection of facial images showcasing a broad spectrum of ages, ranging from 1 year old babies to individuals aged 116. Comprising over 20000 images, each comes annotated with details on age, gender, and ethnicity, and the file names are structured as <age>_<gender>_<ethnicity>_<date&time>.jpg. Every image is a representation of a person's face, representing the input of the system.

The focus is on predicting age, and the task is treated as a regression problem, aiming to output a continuous numerical value representing the age of a person in each image. These images capture diverse elements such as pose, facial expression, illumination, occlusion, resolution, and more. The dataset's versatility enables the model to learn from diverse facial characteristics, since the facial images from the UTKFace dataset are fed into the algorithm. Therefore, using a convolutional neural network (CNN), a deep learning approach, it is possible to develop effective learning hierarchical representations of facial traits and visual patterns for age prediction [1, 2]. The success of the project is measured by the model’s performance on validation and test sets, with a lower MAE indicating improved accuracy in age prediction.

This project extends its utility to various applications, including face detection, age estimation, age progression/regression, among others [3, 4]. Overall, it contributes to the advancement of computer vision applications with real-world implications in various domains.

2. The Importance of Age Detection

This capability for age detection has far-reaching implications across real-world applications, as age is often used to determine eligibility. For instance, in security and access control, integrating age detection

into facial recognition systems enhances authentication processes, ensuring age-appropriate access to secured areas or services [5].

In retail and marketing, businesses can leverage age detection to personalize advertisements and promotions, tailoring content to specific age demographics. Furthermore, it can contribute to the enhancement of customer experiences, from targeted content recommendations in entertainment platforms to age-appropriate gaming experiences [3]. In fields such as healthcare, age detection can support health assessments and demographic studies, offering valuable insights into the aging process and assisting researchers in understanding age-related trends. Besides, age prediction models can help in healthcare applications by measuring health-related indicators and assisting in age-related medical diagnoses [6]. Educational institutions can implement age detection for student authentication purposes, while event management benefits from age verification for compliance with age restrictions at venues. Moreover, age detection can be used by social media platforms and content-sharing websites to enforce age restrictions on certain types of content, assuring age-appropriate content consumption.

As technology advances, age detection through facial scanning using neural networks becomes increasingly integral to diverse sectors, contributing to a more personalized, secure, and efficient integration of facial analysis technologies in our daily lives. However, ethical considerations and privacy safeguards must be prioritized to ensure responsible and transparent deployment of these technologies [7].

3. Data Preprocessing

The project begins by creating a ready-to-use dataset for training machine learning models. This setup is crucial for tasks in computer vision where the model needs to learn to correlate visual patterns in the images with the numerical age labels. Therefore, the first step was to compile the image data and the extract age labels into a Pandas data frame with two columns: one for the images (in some form of image object representation) and the other for the corresponding age labels.

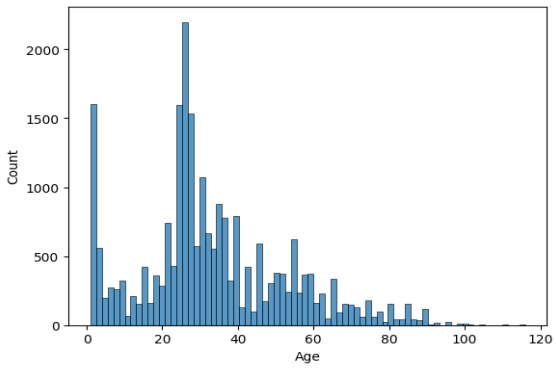

Figure 1

A histogram of ages was created to visualize the distribution of ages. This is useful for understanding the range of ages present and identifying any patterns or outliers in the dataset. We found the dataset unbalanced, as it contains lots of pictures of people with a specific age, but not a whole lot of others (*Figure 1*). We decided to keep the dataset as it is because it seems a good representation of reality (there are fewer people that live above 80 years old than people with age between 20 and 40). Moreover, if we chose to uniformize all the bins, we would also lose relevant information.

The next step was to transform the raw image data into a standardized format suitable for feeding into a neural network. This transformation involved converting images to grayscale and resizing images to a uniform dimension.

3.1. Converting images to grayscale

For this project we chose to do grayscale conversion to reduce computing complexity and to focus on relevant features. Color pictures are three times more complicated than grayscale images since they have three color channels (RGB). The grayscale conversion reduces the amount of data the model must process, enabling faster training and requiring less computational resources.

For age prediction, color information is generally less important than the shapes and textures captured in the image. Grayscale images highlight these aspects, allowing the model to focus on learning features more directly related to age, such as wrinkles or facial structure. Some images may be naturally grayscale or vary in color enhancements. Therefore, standardizing to grayscale ensures that all images are treated equally, allowing consistency in data, and preventing the model from learning biases based on color.

3.2. Resizing images to a uniform dimension

Regarding the nature of this project, resizing the images to a common size was an essential step since neural networks often work with fixed-size inputs. However, choosing the correct size can be sometimes

difficult because if the images are large, downsizing them will remove information and if the images are small, upsizing them will introduce false information.

Therefore, to ensure this uniformity in the input data, all the images were resized to 128x128 pixels (the most common used size in this type of projects because it has a good tradeoff between not remove important information and not to introduce fake information).

The resampling method used was the Lanczos resampling because is often used for high-quality image resizing.

The final step in the pre-processing of the input images was to convert them into numpy arrays to be easily manipulated and fed into machine learning models.

4. CNN Model - Basic

The neural network architecture is designed using the Keras Sequential and Functional API. It includes a two-layer architecture model, including two convolutional layers of 32 and 64 filters, both with a kernel size of 3x3 to the input images and ReLu activation to introduce non-linearity and allow the model to learn complex patterns in the input data. The input shape is specified as (128, 128, 1) indicating grayscale images of 128x128 pixels. The two max pooling layers reduce the spatial dimensions of the input data, retaining the most important features while reducing computational complexity. We have set the pool size to (2,2) indicating a 2x2 window for pooling. The flatten layer is applied to convert the output from the convolutional and pooling layers into a one-dimensional vector before entering the fully connected layers. The first dense layer has 128 neurons and is serving as a fully connected layer, whereas the second dense layer has a single neuron, and it is the output layer producing a continuous numerical value representing the predicted age. Both dense layers have a ReLu activation function.

The model is compiled using the Adam optimizer, Mean Squared Error (MSE) loss function, and Mean Absolute Error (MAE) as the evaluation metric. We chose this configuration as it is appropriate for optimizing the model's parameters in minimizing the difference between predicted and actual age values. The MSE measures the average squared difference between the predicted and actual age values. This is a common metric for regression problems as it penalizes larger errors more heavily, providing a comprehensive measure of overall model performance. In addition to MSE, Mean Absolute Error is employed as a supplementary metric. MAE measures the average absolute difference between the predicted and actual age values. It provides a more interpretable understanding of the average prediction error.

Next, the dataset was split into training (70%), validation (15%), and test (15%) sets using the train_test_split function from scikit-learn. This ensures separate sets for training, fine-tuning (validation), and evaluating (testing) the model. The model is trained using the training data for 50 epochs, with a batch size of 32. The training and validation loss are recorded over each epoch, so we plot the results and evaluate the model's learning progress and identifying potential overfitting or underfitting.

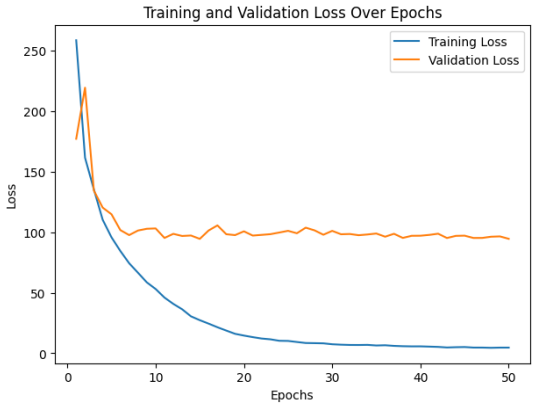

Figure 2

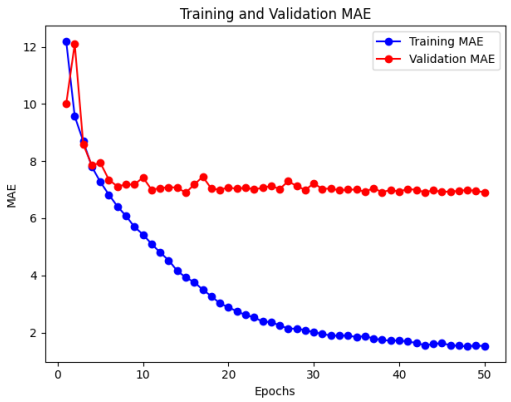

Figure 3

The model shows signs of overfitting, as indicated by the fluctuation in the validation loss (*Figure 2*). While the training loss consistently decreases, indicating that the model is learning, the validation loss does not follow the same downward trend, suggesting that the model may be learning to memorize the training data rather than generalizing to new data. The Mean Absolute Error (MAE) on the validation data decreases

over epochs but begins to plateau, which suggests that the model is not significantly improving after a certain number of epochs (*Figure 3*). This two-layer CNN may not be complex enough to capture the nuances in the data necessary for a more accurate age prediction.

5. CNN Model – Optimization

5.1. Test #1 – Add convolutional layer and reduce the number of epochs

We added an extra convolutional layer of 128 filters and reduced the number of epochs to 10 to try and mitigate overfitting and bring down the training time.

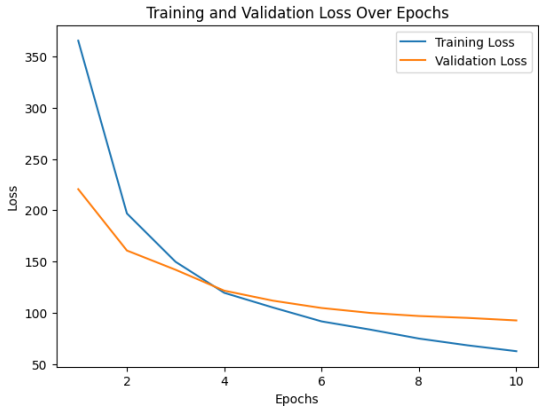

Figure 4

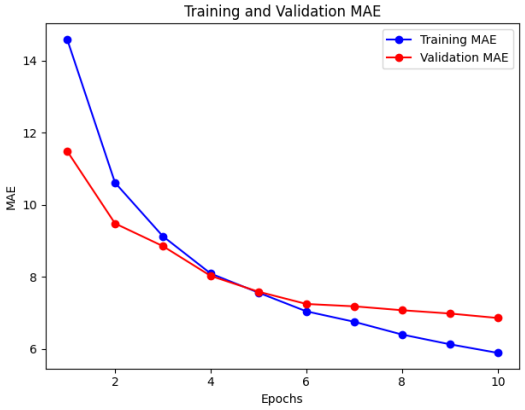

Figure 5

The model starts with a higher loss and MAE in the first epoch compared to the first model. This is not unusual as the addition of a new layer means the network has more parameters to adjust, and it might take a little longer to start converging. Throughout the 10 epochs, the model shows a significant improvement, with both training loss and MAE steadily decreasing (*Figure 4 and 5*). The addition of the third convolutional layer appears to have been beneficial in allowing the model to learn more complex features, which is reflected in the improved MAE on the validation data by the end of the 10 epochs.

5.2. Test #2 – Add dense layers and increase epochs

We added a new dense layer of 64 neurons and increased the epochs to 20.

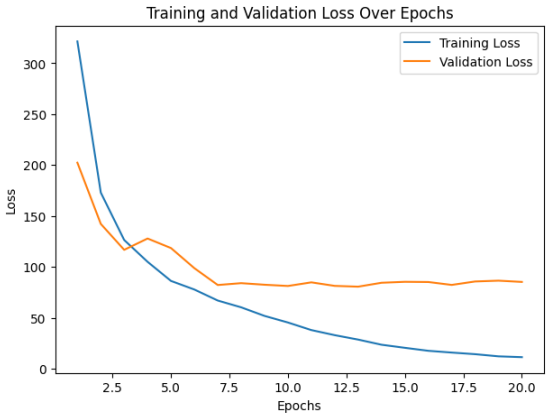

Figure 6

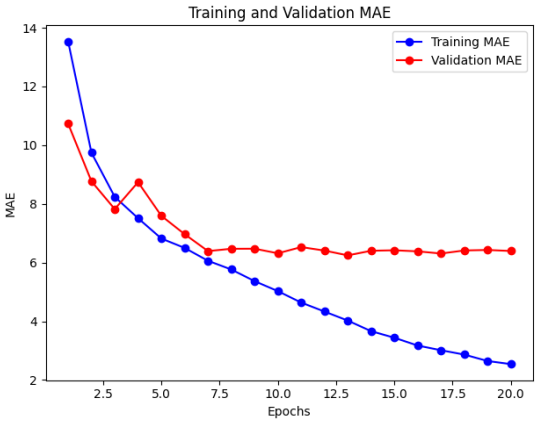

Figure 7

Adding a dense layer increases the model's capacity to combine the learned features in more complex ways, which can potentially lead to better performance but also increases the risk of overfitting. Increasing the number of training epochs to 20 gives the model more opportunity to learn from the data, which can be particularly useful given the additional complexity from the new dense layer. Initially, the validation MAE decreases, reaching its lowest point at epoch 20 with an MAE of 6.3219 (*Figure 7*). This is an improvement over the previous models. However, the validation loss is not consistently decreasing; it shows some variability and even increases at certain points (*Figure 6*). This could be an indication of overfitting, where the model learns to predict the training data well but does not generalize as effectively to the validation data. Due to its inconsistency, we decided to remove the dense layer and erase this experiment.

5.3. Test #3 – Add dense layers and reduce epochs

We added a fourth convolutional layer for increased model depth and decreased the number of epochs to 10.

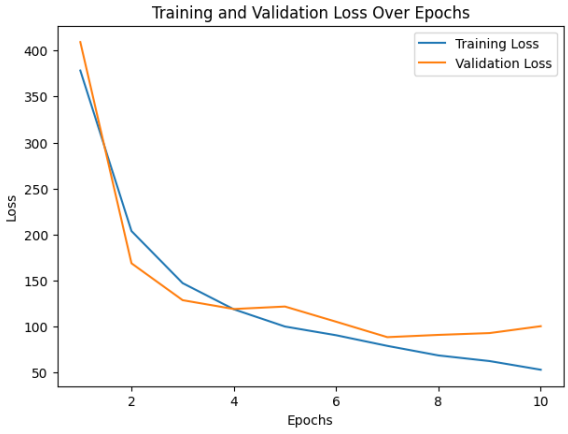

Figure 8

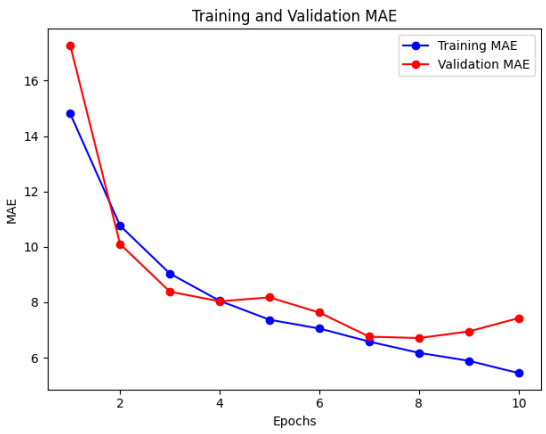

Figure 9

For the next experiment we increased model depth, adding a fourth convolutional layer to the architecture. This layer increases the model's ability to capture more complex features from the images, which could potentially improve the age prediction accuracy, for 10 epochs, which is consistent with the first iteration, focusing on capturing significant performance improvements early in training to avoid overfitting. In the initial training, the model began with a relatively high loss and MAE but showed improvement over time (*Figure 8*). The validation MAE decreased until the eighth epoch (*Figure 9*), suggesting that the model was learning and generalizing well up to that point. However, there was an increase on the ninth and tenth epoch, indicating potential instability in the learning process or the onset of overfitting. The lowest validation MAE was observed in the eighth epoch, after which the performance slightly deteriorated.

5.4. Test #4 – Keep structure and increase epochs

We kept the same structure as the previous model but increased the epochs to 20.

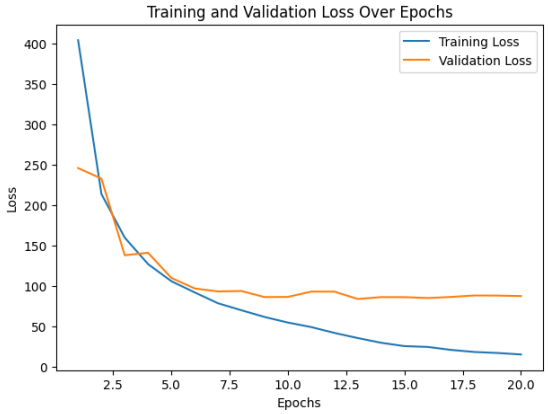

Figure 10

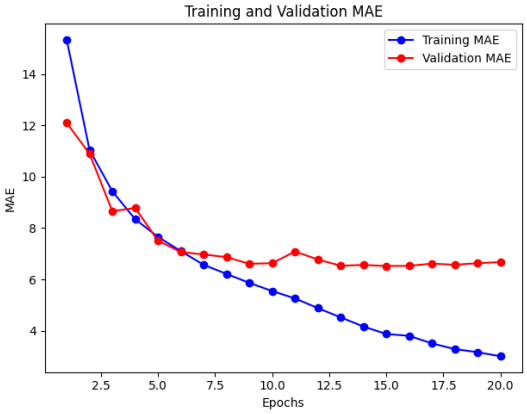

Figure 11

We opted to run the same experiment as the previous one but with 20 epochs. The test MAE after 20 epochs is approximately 6.886, which is an improvement compared to the 10-epoch training of the same model 7.049 MAE (*Figure 10 and 11*).

5.5. Test #5 – Set Adam Optimizer learning rate and reduce the number of epochs

We set the Adam optimizer learning rate at 0.001 and reduced the epochs to 15.

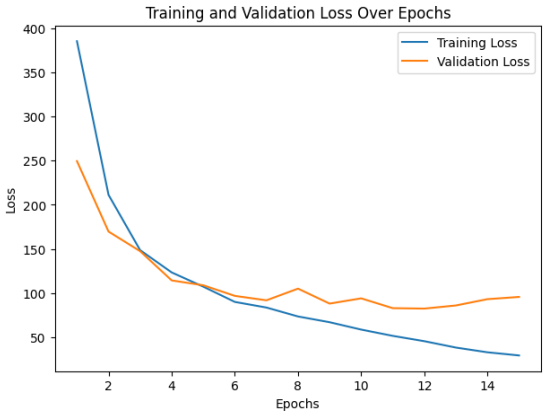

Figure 12

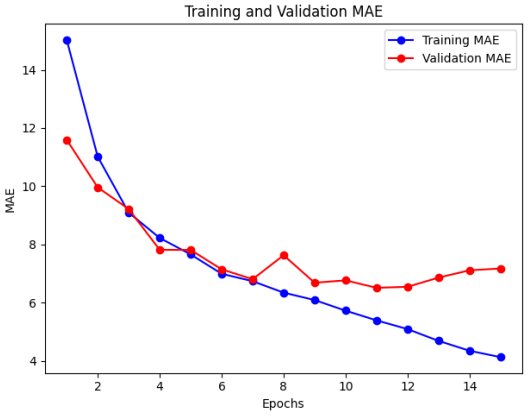

Figure 13

We set the learning rate explicitly at 0.001 for the Adam optimizer. The learning rate is a crucial hyperparameter in training neural networks as it determines the size of the steps the optimizer takes during the gradient descent process. The number of training epochs was reduced to 15. This decision could help prevent overfitting, particularly important for a model with a relatively complex architecture. The training loss and Mean Absolute Error (MAE) consistently decreased over the 15 epochs, indicating that the model was effectively learning from the training data. The validation MAE fluctuated and showed an increasing trend in the later epochs, particularly after epoch 10. This pattern suggests the onset of overfitting, where the model starts to specialize too much on the training data, reducing its performance on the validation data.

5.6. Test #6 – Data augmentation

We enhanced the dataset with data augmentation, a technique commonly used to artificially increase the size of the training dataset by applying various transformations (e.g., rotation, flipping, zooming) to the original images. This helps the model generalize better to different variations of the input data and reduce overfitting. We set the rotation range to 40, meaning that images will be randomly rotated within 0 and 40 degrees; the width and height shift range were set to 0.2, which determine the maximum fractional of total width and height by which the image can be shifted horizontally and vertically, respectively; the horizontal flip parameter was set to True, so images are randomly flipped horizontally and for the fill mode we used 'nearest' which fills the missing pixels with the nearest pixel value for filling in newly created pixels that may result from rotation or shifts.

However, data augmentation seems to deteriorate our model performance, so we will drop this feature and erase this model (*Figure 14 and 15*).

Figure 14

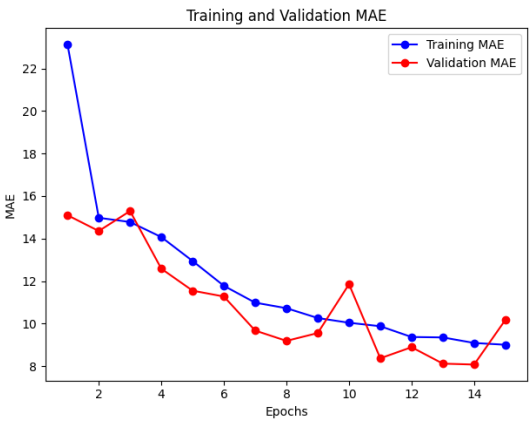

Figure 15

5.7. Test #7 – Integration of learning rate warmup into the training process

We scrolled back to test 5 and adapted, as we see that in early epochs our model isn't as stable as we aimed it to be. So, we integrate a learning rate warmup into the model training process.

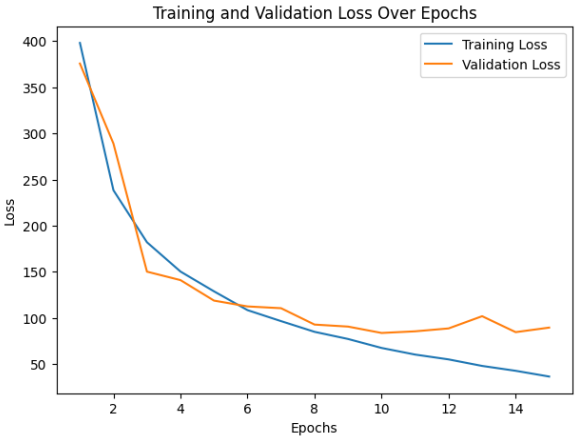

Figure 16

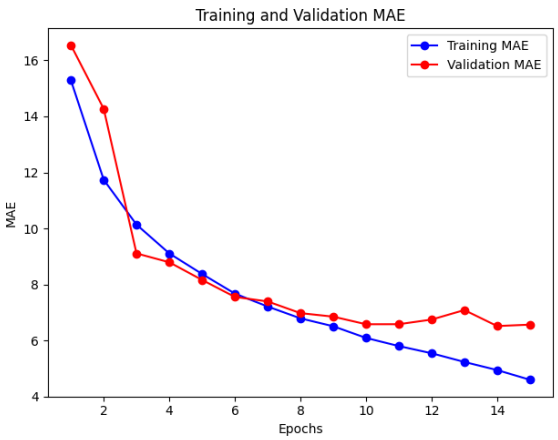

Figure 17

The warmup approach seems to have improved the model's initial stability and learning progress as seen in the decreasing loss and MAE. After the warmup period, the learning rate was kept constant, which seems to have worked well. The model continued to learn and improve across epochs without signs of overfitting, as the validation loss and MAE did not increase significantly (*Figure 16 and 17*).

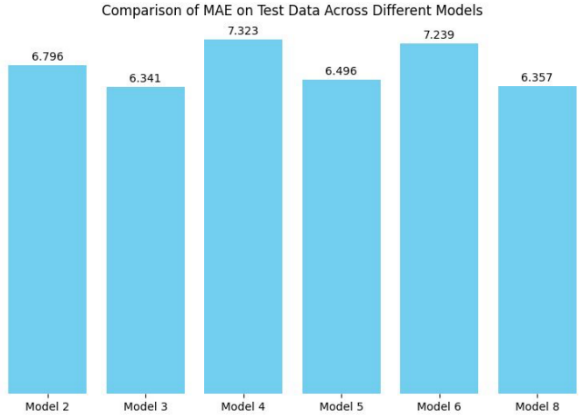

Figure 18

Considering that Test #7 (Model 8 on *Figure 18*) demonstrated the best stability performance on the training and validation sets, it also maintains a good Mean Absolute Error value on test data. Despite Test #2 (Model 3 on *Figure 18*) showing a slightly lower MAE, it exhibited instability on the training and validation sets, Test 7 is the more reliable choice.

6. Error Analysis

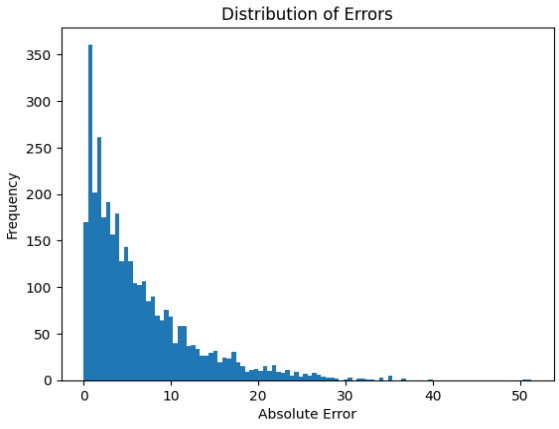

Figure 19

A histogram was created (*Figure 18*) to visualize the distribution of absolute errors between the ages predicted by the model and the actual ages. This allows us to understand the frequency with which the model's predictions deviate by a specific amount from the true ages in the test set. By analyzing the histogram, we identified the predominant frequency of errors, noting that the values are more focused within a range of up to 10 years. The average absolute error corresponding to each true age in the test set is shown in *Figure 19*. This sheds light on how the model performs differently across different age groups, which enables the identification of whether the model tends to excel or struggle within specific age ranges. Higher concentrations of errors are noticeable in older age groups. Following that, we picked out the pictures associated to the worst predictions based on the highest errors in the test set. This lets us visually check those images, giving

us a better understanding of when the model had trouble predicting age accurately. Looking at these pictures helps us find patterns or challenges in the data that might be causing errors in the model.

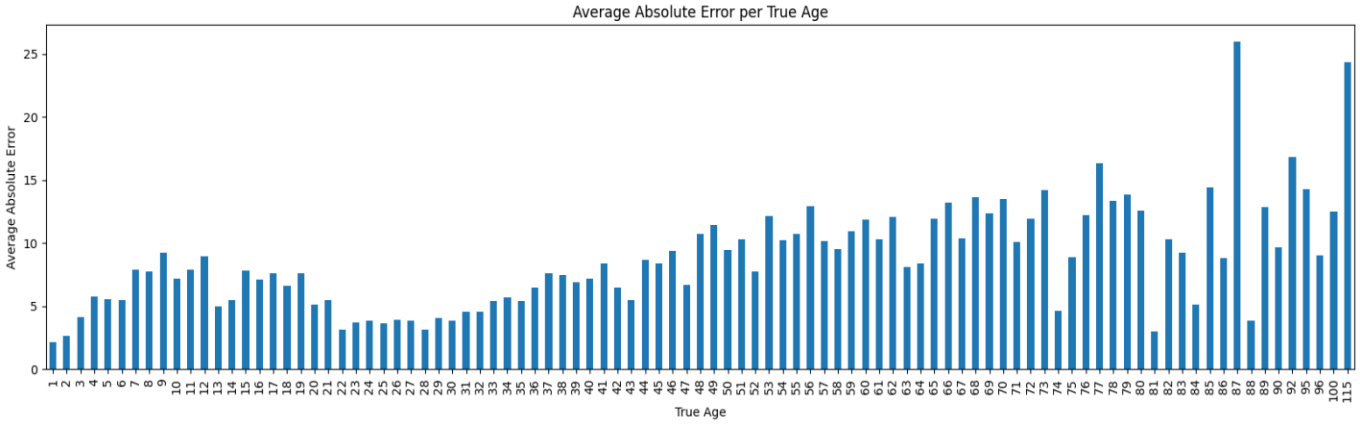

Figure 20

7. Conclusions and Future Improvements

Overall, specifying the learning rate and specifying a learning rate warmup in the initial epochs seems to have positively impacted the model's performance. A learning rate of 0.001 might have offered a good balance between efficient learning and the stability of the optimization process. Despite the increased complexity with the fourth convolutional layer, the adjustments in training parameters (learning rate and epochs) have led to better performance. This suggests that with the right training conditions, the added complexity can be beneficial. Reducing the number of epochs to 15 also appears to have helped mitigating overfitting while still allowing the model to learn effectively from the training data.

Our model still has a lot of room for improvement. We could address age detection by ranges instead of specific numbers, to overcome class imbalances, making the model more robust to variations within age groups and reduce sensitivity to precise age predictions. Class imbalances could alternatively be addressed by undersampling (taking some samples out of the majority class) and/or oversampling (making copies of minority class samples). We can also consider training different models for separate ages rather than one model for all age groups, so that each model can focus on predicting ages within a particular range, which could increase overall accuracy. We could also test data augmentation, based on the characteristics for certain age groups. For instance, younger age groups might benefit from more aggressive augmentation to handle differences in face expressions, but older age groups might require adjustments for changes related to aging. Although adam optimizer is the standard optimizer for this type of models nowadays, we could test different ones, such as RMSprop or SGD, to reveal how different optimization algorithms impact model convergence and generalization.

It's important to note that these enhancements would demand greater computational resources. However, the potential benefits, in terms of improved performance may justify the investment.

8. References

[1]. Oladipo, O., Omidiora, E.O. & Osamor, V.C. A novel genetic-artificial neural network based age estimation system (2022).

[2]. Chen S, Zhang C, Dong M, Lee J, Rao M. Using ranking-CNN for age estimation. In: IEEE conference on computer vision and pattern recognition (CVPR) (2017).

[3]. Dong Y, Liu Y, Lian S. Automatic age estimation based on deep learning algorithm. Neurocomputing (2015).

[4]. Guo G, Fu Y, Huang TS, Dyer C. Locally adjusted robust regression for human age estimation. In: Proceedings of IEEE workshop on applications of computer vision (2018).

[5] Vasavi, S., Vineela, P. & Raman, S.V. Age Detection in a Surveillance Video Using Deep Learning Technique. *SN COMPUT. SCI.* 2, 249 (2021).

[6]. Silva, N., Rajado, A.T., Esteves, F. *et al.* Measuring healthy ageing: current and future tools. *Biogerontology* 24, 845–866 (2023)

[7]. Punyani, P., Gupta, R. & Kumar, A. Neural networks for facial age estimation: a survey on recent advances. *Artif Intell Rev* 53, 3299–3347 (2020).