National Yang Ming Chiao Tung University (國立陽明交通大學)

Computer Vision Final Project
AgeMazing

Team 33

Quentin Ducoulombier (312551811)

Anaé Virama (312551809)

Achille Furger (312551813)

Computer Vision

林奕成

2024-06-16

I. Goal

The objective of this project is to develop a face detection system capable of accurately estimating the age of persons from both images and real-time video streams. This entails designing and implementing an algorithm that can effectively identify and localize faces within static images as well as dynamic video frames, followed by an age estimation model that can analyze the facial features to predict the age of the person. The system aims to achieve high precision and reliability across diverse demographics and varying image conditions.

Figure 1 : AgeMazing project goal

II. Method

A. Data Preprocessing and Age Detection in Deep Learning

Our dataset is a combination of two well-known datasets: UTKFace and AFAD. UTKFace is widely used for age detection due to its broad age range, ethnic diversity, and natural photo settings. However, after analysis, we found a lack of images of Asians. Since our project involves live testing primarily on Asians, we supplemented the dataset with AFAD (Asian Face Age Dataset).

To understand the distribution of images in each age class, we plotted a histogram. The ages were then grouped into predefined intervals to simplify classification. For instance, ages were categorized into groups such as babies (1-4 years), children (5-13 years), students (14-25 years), etc. These groups were used to reorganize the images into training and test sets, with a split of 80% for training and 20% for testing.

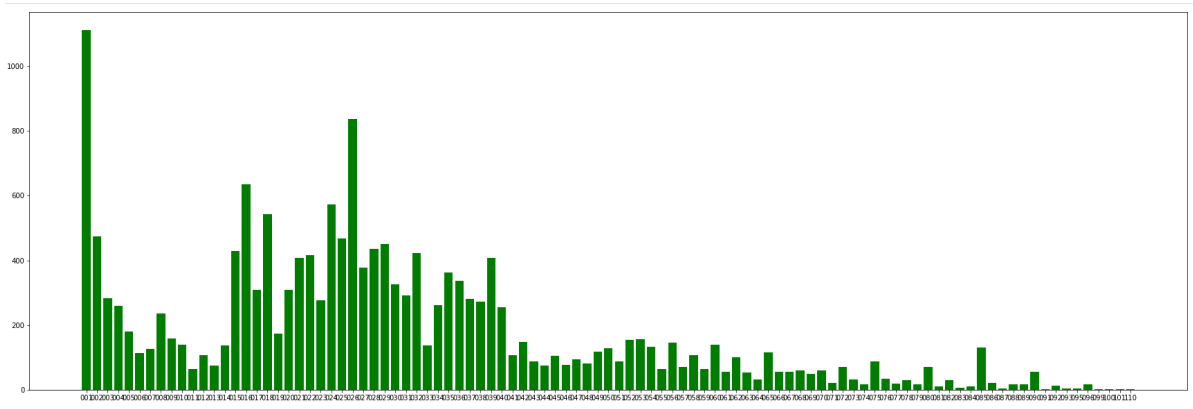

Figure 2 : Repartition of number of images by age

We selected and configured four pre-trained deep learning models: ResNet50, Xception, MobileNetV3, and InceptionResNetV2, chosen from the Keras library. To adapt these models to the specific task of age classification, we added custom classification layers to the end of the models. The models were then compiled with specific configurations, using suitable loss functions such as categorical cross-entropy, efficient optimizers like Adam, and performance metrics like accuracy.

Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7

Figure 3 : Keras website with all the available models and the performances

To enhance the robustness and generalization of the model, data augmentation techniques were used. This included image transformations such as rotation, zoom, and shift, applied randomly to the training images. Data generators were configured to continuously provide batches of preprocessed and augmented images to the model during training.

The model was trained using the data generators, with the application of early stopping. This technique, applied due to the diversity of models, stops training when performance on the validation set no longer improves, thereby helping to prevent overfitting. Loss and accuracy curves for the training and validation sets were plotted to monitor the model's behavior and detect signs of overfitting or underfitting.

After training, the model was evaluated on the validation set to measure its accuracy and loss. Mean Squared Error (MSE) was also calculated to assess the accuracy of the predictions. A confusion matrix was generated to visualize the model's performance in terms of correct and incorrect classifications, helping to better understand the model's errors and identify the most challenging age classes to predict.

To test the model on unseen data, a new image was downloaded and appropriately preprocessed. The model then made a prediction on this image, and the results were displayed to verify the model's generalization capability.

B. Face Detection method

For the face detection component of the project, the method involves using a combination of OpenCV's deep learning face detector and dlib's face detector along with its shape predictor. The process starts with loading a pre-trained deep learning model from OpenCV, which is based on a Single Shot MultiBox Detector (SSD) framework with a ResNet-10 backbone. This model is specified by the `*deploy.prototxt*` that defines the architecture of the neural network and `*res10_300x300_ssd_iter_140000_fp16.caffemodel*` files that contains the pre-trained weights of the model.

To detect faces in real-time video streams, frames are captured from the webcam using the `VideoStream` class from the `imutils` library. Each frame is resized to 400 pixels in width to speed up processing. The frame is then converted into a blob, which is a preprocessed input format required by the OpenCV face detector. A blob is a binary large object that represents the image after mean subtraction, scaling, and channel swapping. This preprocessing step helps standardize the input and enhances the model's performance.

The blob is passed through the deep learning network to obtain face detections. The confidence score of each detection is evaluated, and detections with a confidence score below a threshold (0.5) are discarded to filter out weak detections. For each valid detection, the bounding box coordinates are computed and adjusted to ensure the bounding box is square. This adjustment is crucial because the images in the dataset used for training the age estimation model are all square, ensuring consistent input dimensions.

The detected face region of interest (ROI) is extracted and converted to grayscale for further processing with dlib's shape predictor. Dlib's shape predictor detects facial landmarks, which are used to obtain the coordinates of the eyes and nose. These landmarks are important for aligning the face using an affine transformation to ensure consistent orientation. Aligning the heads is intended to improve the accuracy of the age prediction model, although this alignment process is not yet fully functional.

This approach makes sure that faces are accurately detected and properly aligned in both images and real-time video streams, providing a good foundation for subsequent age estimation.

III. Difficulty encountered

During the project, several challenges were encountered that impacted the development and performance of the face detection and age estimation system.

A. Lack of diversity

One major difficulty was the lack of diversity in the training dataset, particularly the insufficient representation of Asian faces. This raised concerns about the model's ability to perform accurately during the demonstration in class, especially when detecting and estimating the age of individuals with Asian features.

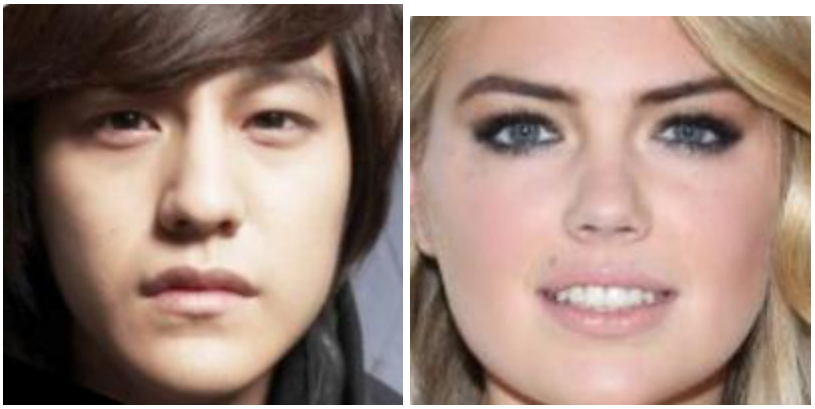

Figure 4 : Example of images in the datasets

B. Face alignement

Additionally, aligning the faces for better age estimation proved to be problematic. Although facial landmark detection and alignment were implemented to improve prediction results, this process did not work as effectively as anticipated. The alignment of faces, intended to standardize the input and enhance model performance, was not fully functional and sometimes introduced errors rather than improving accuracy.

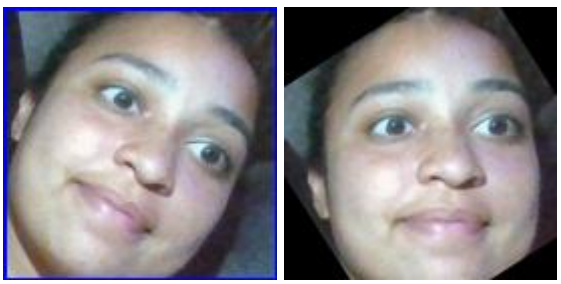

Figure 5 : Original face and the aligned version

We can see that there are some black parts due to the rotation of the original image.
A way of improvement would be to replace those black parts by the corresponding one in the original image or video stream.

C. Performance

Another significant challenge was the performance of the age prediction model. Despite extensive efforts to train and optimize the model, the predictions were not as accurate or reliable as expected. This issue highlighted the need for further improvements in the model's architecture and training process to enhance its accuracy.

We did our best to address these challenges, but some issues remain unresolved. It indicates areas for future improvement and refinement. This affects the overall performance of our project, but despite these difficulties, the results are still satisfactory.

IV. Libraries and Open Sources

For the face detection module, several libraries were utilized to implement and optimize the detection algorithms.

OpenCV (`cv2`) played a central role, providing functions for image processing, computer vision tasks, and integrating a pre-trained deep learning model for real-time face detection.
Numpy (`np`) was essential for efficient numerical computations and array operations, supporting data manipulation tasks within the detection pipeline.
TensorFlow (`tensorflow`) facilitated the loading of deep learning models and preprocessing of input data.
Additionally, the `**imutils**` library offered convenient tools for video stream handling and image resizing, contributing to the real-time capabilities of the face detection system.
Dlib provided advanced functionalities such as facial landmark detection and alignment, enhancing the accuracy of face localization.
Together, these libraries formed a robust foundation for building and deploying the face detection component of the project.

For the age prediction module, **TensorFlow** (`tensorflow`) was extensively utilized for building, training, and deploying deep learning models. This included defining neural network architectures, handling data through preprocessing pipelines, and performing model evaluations. The `ImageDataGenerator` from TensorFlow enabled efficient data augmentation and batch processing, crucial for enhancing model generalization and performance.
Pandas (`pd`) supported data management tasks, facilitating dataset organization and manipulation.
Matplotlib (`plt`) was used for visualizing training metrics and evaluation results, aiding in model analysis and interpretation.

Additionally, standard **Python** libraries such as `os` provided utilities for file and directory operations, contributing to dataset preparation and management.
These libraries collectively empowered the age prediction module to effectively train and deploy models for accurately estimating the age of individuals based on facial images.

V. Contribution

The contributions to this project were divided among the team members, each focusing on specific aspects to ensure comprehensive development.

Quentin selected the deep learning models, trained them, and managed all tasks related to the dataset, including its preparation and handling.
Achille focused on the preprocessing of images, ensuring that the data fed into the models was appropriately formatted and enhanced to maximize prediction accuracy.
Anaé was responsible for the face detection component, implementing and refining the algorithms that identify and localize faces in both images and real-time video streams.

Each member's efforts were important in bringing together the various elements of the AgeMazing project.