# QAC239 Final Project
# Exploring Age Classification Models using Deep Neural Networks

**Github Repository: https://github.com/xynlophyl/qac239_finalproject**

**Nelson Lin**

## Introduction

Computer Vision is a field in Computer Science that focuses on training computers to recognize and understand visual information. Researching into Computer Vision seeks to replicate human vision, and our methods in recognizing patterns within complex visual cues. Within the field, there are a variety of opportunities for real-world applications, such as Text Extraction, Augmented Reality, and Object Detection/Tracking. Through this project, I hope to explore the various implementations of face detection and age classification.

## Data & Methods

**Models: Face Detection**

For face detection, I utilized the Deep Learning face detection model we learnt in class, Dlib's MMOD Human Face Detector. The model utilizes a Convolutional Neural Network (CNN), which is a commonly used neural network infrastructure for image analysis. A CNN is a deep learning model designed for tasks in computer vision, such as image classification and object detection, by abstracting key patterns and features in numerical-based matrix inputs.

**Models: Age Detection**

The age classification model used in this project was obtained off the OpenCV python library. Initially developed by Gil Levi and Tal Hassner, the model also uses a CNN to predict ages based on facial features of a given subject. The model was exported as a Caffemodel, which stores all its parameters and weights in a supporting .prototxt file.

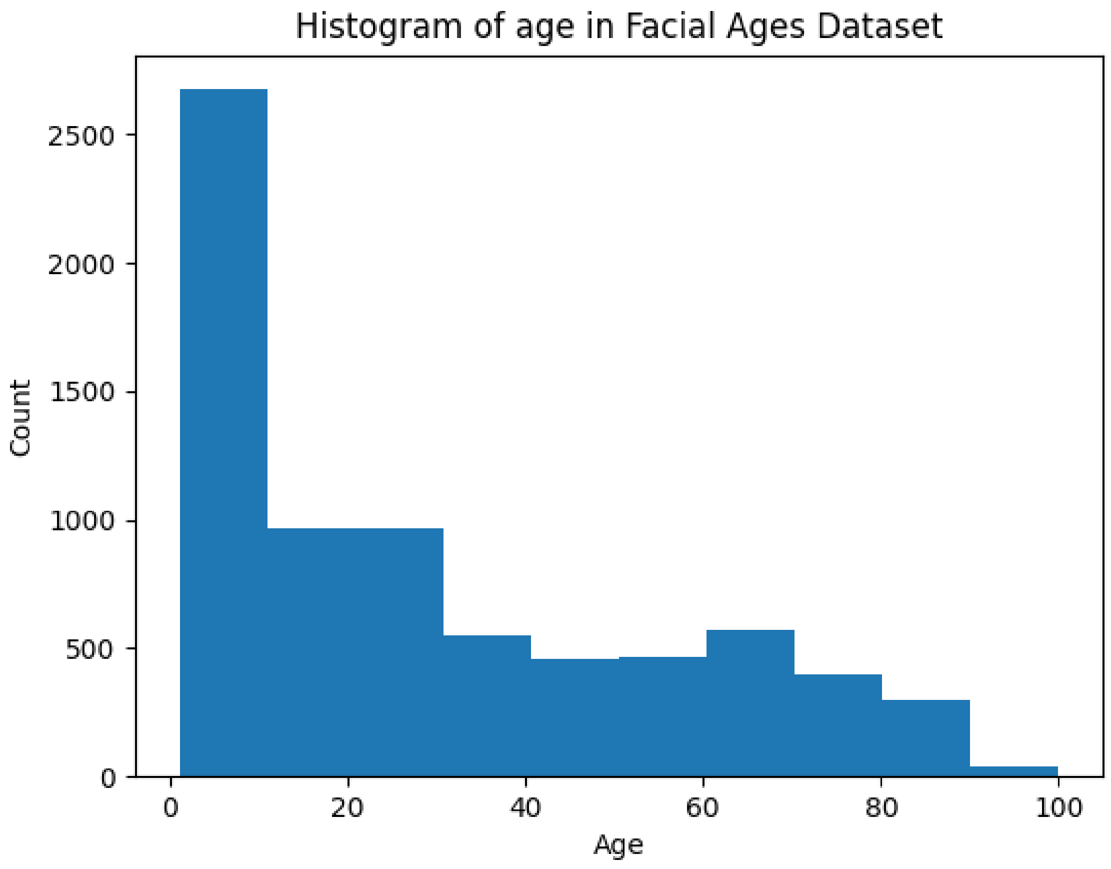
The dataset used in the pre-trained age detection model is called the OUI-Adience Dataset. Collated by the Open University of Israel, the dataset features images of unfiltered faces, specifically collected for the purpose of training Machine Learning detection models, in both age and gender classification. The images in the dataset are intended to be as representative towards challenges faced in real-world photography as possible, this means that each image is prone to various types of noise, such as lighting and inconvenient angles. Within the dataset, there are 26,580 photos of 2,284 subjects. Each photo is also labeled into 8 distinct age groups:

(0-2, 4-6, 8-13, 15-20, 25-32, 38-43, 48-53, 60-100)

I was not able to gather any statistical data about this dataset, such as the mean age of the dataset, or the distribution of images per age label.

**Evaluation Dataset: Facial Ages (Kaggle)**

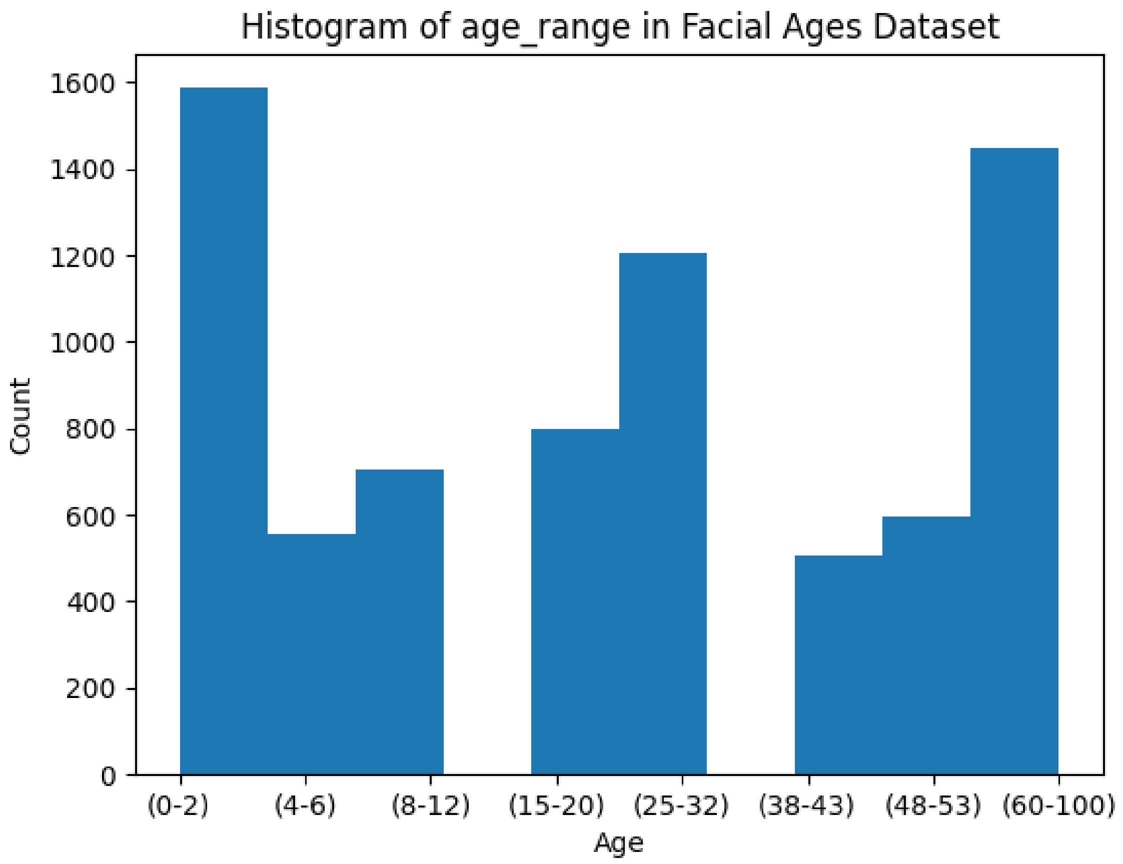
The source of the main dataset used in this project was obtained off the data science website Kaggle, uploaded and compiled by user Fazle Rabbi. The dataset contains 7395 facial images, each with a corresponding age label within the range [1 - 110]. Using this dataset, my goal was to evaluate pre-trained models on their accuracy and efficiency, whilst also attempting to continue training and fine-tuning the model to improve its predictions on specific age groups.



The distribution of the dataset is shown above. As shown, the dataset has a heavy skew towards young ages, especially between 1-10 years old. This means that in terms of training, the model will have a large sample within those year groups, and can hopefully have a better performance when predicting young faces.

**Data Preprocessing**

Before running the model and beginning my evaluation, I needed to load the Facial Ages dataset onto a Pandas dataframe. In order to utilize the dataset with the OpenCV model, I needed to map the ages of each image to their respective age labels given by the model. Since the model's age labels are not exhaustive (i.e. there are missing ages between labels), the first step was to filter out the missing ages out of my dataframe, before performing the age group mapping. Below is the updated distribution plot, grouped by age groups rather than individual age:



Next, to evaluate and train the model, I gathered a training and testing set, using sklearn's train_test_split function, with sampling seed = 1234. My resulting dataset was split into 3 sets:

- pre-training test (10% of the dataset): to evaluate the existing pre-trained model
- training (70%): to fine-tune the model
- post-training test (20%): to evaluate the initial and final model, and compare their performance

**Data Description**

The resulting evaluation set has the following variables:

- File Path: image relative file path
- Age: Age of face in image
- Actual Range: Mapped age label (from Age)
- Predicted Range: Model's predicted age range
- Face Detection Time: Time taken for model to detect face
- Age Detection Time: Time taken for model to predict age
- Total Detection Time: (Face Detection Time) + (Age Detection Time)

A sample of the set looks like the following:

| File Path | Age | Actual Range | Predicted Range | Face Detection Time | Age Detection Time | Total Detection Time |
|---|---|---|---|---|---|---|
| 042/2113.png | 42 | (38-43) | (38-43) | 15.34479 | 0.043685 | 15.38848 |
| 065/8544.png | 65 | (60-100) | (8-12) | 17.2328 | 0.014986 | 17.24779 |
| 028/5002.png | 28 | (25-32) | (4-6) | 13.06837 | 0.013531 | 13.0819 |
| 080/5404.pn | 80 | (60-100) | (25-32) | 12.64578 | 0.013475 | 12.65925 |

| | | | | | | |
|---|---|---|---|---|---|---|
| g | | | | | | |

**Methodology**

My framework for running each model, and evaluate their performance, is guided by the following procedure:

1. Load an image using cv2.imread() and the image's file path
2. Resize the image, if needed, and create a gray-scale copy
3. Use Dlib's face detection model to obtain the rectangles of the face in the gray-scale image
4. Input the rectangles and the color-scale image into OpenCV's age classifier model to obtain a vector of probabilities for each age label
5. Take the age label with the highest probability (using argmax) to get the model's age prediction

Note: When performing predictions using the age classification model, the initial parameters for the model's mean were set to (78.4263377603, 87.7689143744, 114.895847746).

## Results

**Pre-Training Evaluation**

Before beginning training, I first ran the OpenCV model with the pre-training test dataset. To evaluate the model's performance, I compared each predicted age label with the actual age range, which was mapped from their actual age previously. In terms of overall performance, the model had an accuracy of 33.29% (243 out of 730). The model had the greatest accuracy when predicting the (0-2) age group, with 66.30% accuracy (120 out of 181), while performing the worst at the opposite end of the spectrum, having a 0.79% prediction rate (1 out of 126) when predicting in the (60-100) age group.

I speculate that the model's varying performance could be due to the difference in ranges between the two age groups. Since (60-100) has a larger distribution of ages, there is a higher variance of the facial features that can occur in faces that fit within this age range. Thus, while training the model, I believe that it would be more difficult for the model to find patterns and prominent identifying characteristics, when compared to that of the (0-2) age group.

Analyzing the speed of predictions, the overall model took an average of 13.22s to return a prediction. Looking deeper into each model's efficiency, the majority of the prediction time lies in the face detection model, taking 13.21s compared to the 0.01s taken by the age classification model.

This surprised me, as when evaluating face detection models in class, the Deep-CNN model was significantly faster, with a mean speed of 1.1s. I believe that this may be due to the fact that I performed my predictions in this project using my laptop's CPU, rather than the GPU processing available on Google Colabs.

**Training / Troubleshooting**
My next step is to train the model, using a fine-tuning approach. Fine-tuning a model is the process of adjusting a pre-trained model's parameters to a specific task or dataset. In this case, my plan was to fine-tune the dataset to help it perform better on the facial age dataset, using TensorFlow's Neural Network suite to perform back propagation and parameter tuning.

The issue with this plan was that pre-trained Caffemodels are not able to be tuned or altered, in any way, after they have been exported. I attempted to use numerous methods to convert the Caffemodel into tunable TensorFlow models, but every package was either outdated or failed to run. Thus, I decided to pivot my project in a different direction, and apply the model into a real-time age classifier.

## Application: Real-Time Age Classification

My final stage in this project was to use the age classification model within a live video stream. The steps for implementing this application is as follows:

1. Initialize a video stream of the user's web camera using imutils.video.VideoStream()

2. Capture a frame of the video stream, and perform the image processing (resizing, gray-scale conversion), as with the original methodology
3. Run face detection and age detection, as with the original methodology.
4. Plot the rectangles and age prediction onto the frame and display the plot
5. As each frame gets processed, update the plots to maintain the stream of frames.

In order to do so, I needed to improve the face detection model's detection time. This was solved by replacing the existing CNN model with a Haar Cascade. Haar Cascades are an alternative method for face detection. Detecting Haar-like features, which are represented by rectangular features to detect patterns in contrast changes, Haar Cascade detects facial features to identify faces in images.

## Conclusion

Computer Vision is an area of Computer Science that has seen massive development over recent years. From autonomous vehicles to augmented reality, robots and autonomous systems have developed an intelligence to identify various objects and their various features. In my project, I set out to explore the use of machine learning models for facial recognition and age classification. While I was unable to investigate the training aspect of building such a model, I was able to learn a lot about how these models are built and evaluated, while also being able to implement an application that is usable in real-world scenarios. For future research, I would like to find a trainable Neural Network model or explore building a model from the ground up, in order to learn about training such a model, and gaining deeper insight into how these complex systems analyze and recognize patterns in numerical data.

## References

https://talhassner.github.io/home/publication/2015_CVPR
https://talhassner.github.io/home/projects/cnn_agegender/CVPR2015_CNN_AgeGenderEstimation.pdf
https://talhassner.github.io/home/projects/Adience/Adience-data.html#agegender
https://www.kaggle.com/datasets/frabbisw/facial-age