Using Deep Learning Techniques For Face Detection On Cinematic Eye-Tracking Data

Shinn Taniya
Department of Computer Science, Harvey Mudd College, Claremont, CA
staniya@g.hmc.edu

1 Introduction

As interest in the interplay between artificial intelligence and human-computer interaction increases, researchers attempt to define models that accurately describe novel interactions between humans and machines. Amongst the many interactions that researchers have been studying, eye-tracking has gained increasing attention. This is because many studies have found that "facial biometrics have more distinct features that allow for them to have advantages over other biometrics such as palmistry and fingerprint". (Ameen Sulaiman and Sarhan Kocher, 2022) Since eye movement is the most frequent of all facial movements, it is logical that eye-tracking research has garnered increasing attention. Furthermore, since eye movements are fundamental to the operation of the visual system, the movement of the user's eyes can provide a convenient and natural source of data input. Thus, eye-tracking technology is applicable in many domains such as psychology, marketing, medical, computer gaming, and cognitive science. (Lim et al., 2022) Hence, there is a need to optimize eye-tracking algorithms so that they can be widely adopted for classification tasks.

In eye-tracking research, a fundamental process is to perform feature labeling so that eye-tracking data can be categorized based on the essential ocular activity indicators. Traditionally, feature labeling was largely done by humans which were both tedious and inefficient. Consequently, to help researchers minimize the work and effort of data labeling from scratch, we present machine learning models that perform automated feature labeling, specifically a bounding box prediction and classification of human heads, on the Gaze Data for the Analysis of Attention in Feature Films. The dataset contains robust gaze information of human participants in response to carefully curated film clips and has been augmented via the annotation of selected high-level cinematographic and ocular activity features.

In this work, we employ several architectures for face detection to assist contemporary research done to study the allocation of human gaze over visual stimuli. The study will involve three parts. The first is a baseline model that uses Haar Cascade, an object detection algorithm used to identify features of a face in an image or a real-time video. This approach performs feature detection on an image containing a face, based on the ratio of the intensities of the pixels within the target image. The second is to apply Vora et al.'s Fully Conventional Head Detector (FCHD), an end-to-end trainable head detection model, to perform classification and implement a localization function for the Gaze Data. (Vora and Chilaka, 2018) The third is to employ Deepface, a face recognition and facial attribute analysis framework that wraps state-of-the-art models including VGG-Face, Google FaceNet, Google OpenFace, Facebook DeepFace, DeepID, ArcFace, and Dlib. By exploring and implementing these state-of-the-art models, this study seeks to identify the limitations of current algorithms in detecting people under diverse camera conditions, human poses, and lighting.

2 Related Work

With the proliferation of deep learning techniques ranging from neural networks to pre-trained processing models, there exists a multitude of computer vision research done for constructing a trainable face detection model. However, as Vora et al. point out, current state-of-the-art face detection algorithms that incorporate deep learning models suffer from high false positives. (Vora and Chilaka, 2018) This is because prominent models such as an RCNN head detector that contextualizes person-scene relations and pairwise relations amongst target objects are designed specifically for image applications where the average head size is small and the density of people is low. Evidently, these algorithms struggle to accurately identify faces in crowded scenes as they fail to incorporate "large scale and density variations" seen in these data. (Vora and Chilaka, 2018) As a solution, Vora et al. propose FCHD to perform bounding box prediction and classification of human heads. In their research, Vora et al. concluded that the model not only has low memory requirements but boasts a better overall average precision (AP) which it achieves by "selection of anchor sizes based on the effective field of the network". (Vora and Chilaka, 2018).

To further continue their studies in assessing the performance of FCHD, we will apply it to the Gaze Data for the Analysis of Attention in Feature Films. Testing the model against this dataset is significant as it serves as an effective testing set for the model. Compared to the HollywoodHeads dataset that the model is trained on, the Gaze Data dataset includes fewer average headcounts per scene and less crowd density. Given

that the Vora et al. discovered that their model gives false positives in scenarios where no heads are to be detected, we hope to further the performance of their FCHD model by fine-tuning the architecture to adjust for diverse circumstances.

Another experiment we hope to further investigate is DeepFace, a modern face recognition pipeline offered by Serengil et al., which offers deep learning modules for each of the four conventional stages of a face recognition pipeline: detection, alignment, representation, and verification. (Serengil and Ozpinar, 2020) The module that is most relevant to the scope of our research is that of representation. Similar to other prominent face recognition models, DeepFace's foundation is a convolutional neural network (CNN). By training the CNN to classify the identities of the objects in each input image, DeepFace's early layers are able to extract a few face representation feature vectors. This allows it to adapt to and verify faces it has never seen. Many studies have shown the advantages of a CNN framework, including that of Vu et al. who constructed a CNN framework that merges context-aware models and a CNN-based local director, and achieved state-of-the-art classification results on contextually challenging still images. (Vu et al., 2015)

To continue, as aforementioned, DeepFace offers many pre-trained state-of-the-art face recognition models to evaluate the scenes in our dataset. In their publication, Serengil et al. found that using a Facenet512 face recognition model, testing on the LFW Face Database, the model had an accuracy score of 99.65%, surpassing the 97.53% accuracy that human beings have in manual face recognition tasks for labeled faces. Thus, we are employing the DeepFace library to effortlessly test the performance of complex face detection models. (Serengil and Ozpinar, 2020)

3 Dataset

The Gaze Data for the Analysis of Attention in Feature Films is designed to facilitate the study of exogenous eye movements. It is available publicly and contains robust gaze information of human participants in response to carefully curated film clips. The gaze location was recorded using a Gazepoint GP3 tabletop eye tracker and the clips were chosen from a set of 13 candidate films that maximized the presence of distinctive features. The films, clip durations, directors, and cinematographers are summarized in Figure 1. As evident, the films are prominent films produced between the years 1984 through 2014 and were selected by Breeden et al. to ensure that the dataset minimized biases towards individual styles of cinematographers.

For each of the films in Figure 1, clips were selected with an eye toward data heterogeneity, so that later computer vision techniques can be applied to distinguish between ocular activity features. To aid in the exploration of higher-level image features and understand their relationships to ocular activity, each clip has been augmented via frame-by-frame, hand-annotated selected high-level cinematographic and ocular activity features, including: faces, dialogue, camera motion, and edits.

Film	Year	Duration	Director	Cinematographer
Amadeus	1984	3:16	Miloš Forman	Miroslav Ondříček
Argo	2012	3:35	Ben Affleck	Roberto Prieto
Birdman	2014	3:29	Gonzalez Iñaritu	Emmanuel Lubezki
Chicago	2002	2:30	Rob Marshall	Dion Beebe
The Departed	2006	1:36	Martin Scorsese	Michael Ballhaus
Gladiator	1999	3:03	Ridley Scott	John Mathieson
The King's Speech	2010	3:01	Tom Hooper	Danny Cohen
The Last Emperor	1987	2:12	Bernardo Bertolucci	Vittorio Storaro
No Country for Old Men	2007	0:56, 1:59	Joel and Ethan Coen	Roger Deakins
Saving Private Ryan	1998	2:53	Steven Spielberg	Janusz Kaminski
Shakespeare in Love	1998	1:20, 2:05	John Madden	Richard Greatrex
Slumdog Millionaire	2008	2:43	Danny Boyle	Anthony Dod Mantle
Unforgiven	1992	2:26	Clint Eastwood	Jack N. Green

Figure 1: Films, Clip Durations, Directors, and Cinematographers.

Category	Coding		Description
Faces	`f`	`frameX frameY`	A single face is visible for the specified range of frames: `[frameX, frameY]`
	`ff`	`frameX frameY`	Multiple faces are visible
	`fa`	`frameX frameY`	One or more non-human faces are visible

Figure 2: Binary Face Features

3.1 Pre-processing

The tool FFmpeg is used to generate frames such that the frames of the raw film sequences are keyed to the hand-coded features. To create a map to keep track of the correspondence between hand-coded features and the frames, both the gaze data and the hand-coded features are stored as a Pandas data frame. Certain features of 'head recognition' are selected based on their theoretical relevance, and other features that seemed redundant and ineffective are removed. The logic behind this is that features such as dialogue, camera movement, and text don't demonstrate semantic importance in configuring a head detection localization function in comparison to features such as faces. Thus, from the two Pandas data frames, most columns are dropped, excluding markers related to frame rates, shot numbers, or Spatio-temporal indicators. It is important to clarify that the features are encoded as binary features, such that for each feature, its presence or absence is marked by a 1 or 0 respectively. Figure 2 provides an example of this.

Then, for each of the experimental approaches, the data is prepared differently. For the baseline model that employs Haar Cascade, the data is processed by OpenCV by converting the images from BGR order to RGB order. Following, the images are converted again into grayscale such that the Haar Cascade Classifier can process the image in four stages: 1) calculation of Haar features, 2) creation of integral images, 3) Adaboost, and 4) implementation of cascading classifiers.

For FCHD, the input images are scaled such that the shorter dimension of the image is converted to the size that is equal to a specified minimum pixel length

such that the input images can be normalized. This is to ensure that each input parameter has a similar data distribution such that convergence is faster while training the network. Finally, for the DeepFace model, we simply created a directory structure such that for each candidate film, its frames were images that were hand-annotated to contain a single or multiple faces within.

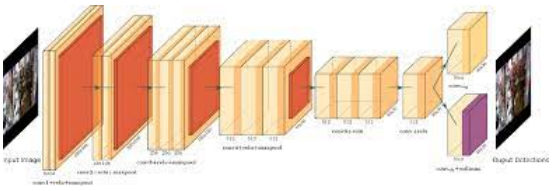

Figure 3: FCHD Architecture

4 Methods

4.1 Haar Cascade Baseline Model

Given that the Haar cascade model is not a deep learning face detector, the latter models should have significantly higher accuracy and more robust face detections. However, the benefit of using Haar cascades is that they are computationally efficient, have low memory requirements, and have a small model size.

The development environment requires OpenCV and haarcascade_frontalface_default.xml, the pre-trained face detector provided by the developers and maintainers of the OpenCV library. The *cv2.CascadeClassifier()* is used to load the detector on disk. Then, taking the grayscale images generated from the data preprocessing phase, we use *cv2.rectangle()* to draw a bounding box around the detected faces. Note that the bounds of the boxes are determined by the classifier's *detectMultiScale()* function which detects the faces in the input and returns a list of bounding boxes (represented using x and y coordinates where the faces are in the image).

To discuss our choice of parameters, the *scaleFactor* controls the extent to which the image size is reduced at each scale. In this experiment, we selected a value of 1.05 so that the size of the image is reduced by 5% at each level in the scale pyramid. Next, *minNeighbors* parameter controls the number of neighbors each window should have for the area in the window to be considered a face. This threshold was set to 7 as the Haar Cascades are incredibly sensitive to the parameter choices and after trial and error, 7 was the value that minimized the number of false-positive detections. The final parameter is *minSize* which indicates the window's minimum size. To specify, *minSize* is a tuple of width and height in pixel units that serves as the minimum dimensional threshold to which a bounding box can be considered valid.

4.2 FCHD

The goal of this section is to outline the process in which we can fine-tune an FCHD model. As noted by Vora et al., deep architectures are powerful as they are capable of adapting to "generalized features" that have high applicability. (Vora and Chilaka, 2018) The FCHD model employs a VGG16 model as its base model. The VGG16 is considered a powerful architecture as it has been proven to be able to classify 1000 images of 1000 different categories with 92.7% accuracy. Due to its public availability and ease to apply to transfer learning, it is one of the most popular deep learning algorithms for image classification.

The architecture of Vora et al.'s FCHD head detection model is shown in Figure 3, where in addition to the VGG16 model above, three new convolutional layers are added to allow for further hierarchical decomposition of the input. For the purposes of our experiment, given that we are employing the same FCHD model, we set identical hyperparameters to Vora et al. The pre-trained VGG16 model is trained using the ImageNet dataset, and the new layers are "initialized with random weights sampled from a standard normal distribution with $\mu = 0$ and $\sigma = 0.01$". (Vora and Chilaka, 2018). Other hyperparameters include a weight decay of 0.0005, a learning rate for the training set to 0.001, and an epoch count of 15. Note that the whole training uses a PyTorch framework.

Now, to discuss the reasons we chose to employ an FCHD model, Vora et al.'s model specializes in head detection for crowded scenes. Given that the Gaze Data contains scenes where multiple characters from a movie scene are conglomerated in one area, sophisticated deep learning frameworks that can detect faces under challenging conditions (such as high occlusion) was necessary.

4.3 DeepFace

The methodology for implementing DeepFace is incredibly simple. Given that the face recognition pipeline is offered as a PyPI package, the only requirements include installing the library itself and its prerequisites. The advantages of this library are obvious. As aforementioned, DeepFace offers all procedures necessary for face recognition including: detect, align, normalize, represent, and verify. (Serengil and Ozpinar, 2020) For the purposes of this research, we will use the Face Recognition algorithm they offer to analyze the cosine similarity between images and to configure a localization function. DeepFace serves as a good evaluation metric for understanding the performances of the previous two architectures as well. In addition to the conventional statistical metrics: accuracy, precision, and recall, employing DeepFace will allow us to identify a measure of result relevancy and understand the number of truly relevant features that the model returns.

(a) Kings (b) Argo (c) Unforgiven

Figure 4: Haar Cascade Results

Figure 5: Deep Face Recognition Demo Output

5 Results and Discussion

5.1 Haar Cascade Baseline Model Results

We were able to run the Haar Cascade algorithm to perform automated head detection on all of the frames for every film. Unfortunately, the detection accuracy for the model is not very high. The model performs well when one or several people are clearly in the frame as shown in Figure 4. However, in certain variations in human pose, background clutter, motion blue, low image resolution, occlusions, and poor lighting conditions, the Haar Cascade algorithm performs poorly as shown in Figure 4. This is somewhat to be expected as we did not apply any image correction techniques beforehand to remove the bias and noise that skew the outcome of our models. Since the intensity of the pixels is an important determinant of the accuracy of the Haar Cascade model, we can apply image processing techniques such as histogram equalization to improve contrast in images. This way, when the model collects Haar features by performing calculations on adjacent rectangular regions at a specific location in the detection window, there will be a greater difference between the sums of the pixel intensities, leading to stronger identification capacities. Furthermore, since we are applying the Haar Cascade algorithm as our baseline model, we did not fine-tune the pre-trained face detector haarcascade_frontalface_default.xml on our Gaze Data. Since this algorithm requires a lot of positive images of faces and negative images of non-faces to train the classifier, the pre-trained machine is most likely unable to detect underlying patterns in our dataset.

5.2 FCHD Failure

We made a grave error with the FCHD model as although the VGG16 model came pre-trained, the FCHD architecture provided by Vora et al. forced us to train a completely fresh model using our training set. Hence, following a conventional approach, we split the hand-annotated feature data frame into training and testing sets by using Scikit-Learn's train_test_split() method with a test_size of 0.3 and a random_state of 42 so that we can randomly select 30% of the data to belong to the testing set. We do so to calculate out-of-the-bag errors and evaluate how our algorithm generalizes to completely unseen data. Then, to see the general trend in feature importances to drop semantically insigifnicant features from the full list of hand-coded features, we attempted to implement a Random Forest Classifier (RFC) to predict the relationship of each of the high-level features to the head feature (the target binary variable). Then, we employed 5-fold cross-validation and Optuna's hyperparameter optimization framework to estimate the performance of the model on new data through hyperparameter tuning. However, while we were working on installing the necessary dependencies, we ran into another major problem in that the CuPy dependencies that the FCHD framework requires to run were not only outdated but incredibly memory intensive. Despite setting up Conda environments within our allocated XSEDE folder, the training will stop due to reaching the quota capacity or because of a dependency error.

This result is incredibly disappointing for us as we consumed many hours training the FCHD model on the aforementioned Random Forest hyper-parameter optimized training set along with the frames of the raw film sequences the features are keyed to; all so that we can further the research done by Vora et al. However, we still believe this to be an educational experience as we learned about data preprocessing techniques, dependency management, error handling, and machine learning architectures trying to implement the FCHD model.

5.3 DeepFace Results

We were successful in generating multiple pkl files that demonstrate the cosine vector similarity between various frames. One such pkl file that is saved in the Github Repository Link at the bottom of the page holds the cosine similarity scores between my Facebook profile picture and the frames corresponding to the film Argo. Using the popular VGG-Face model, the algorithm verified that my face is indeed a human face, and attempted to discover similar faces within the Argos film frames so that it can calculate the cosine similarity scores. This face recognition model that builds on TensorFlow and Keras is incredibly interesting as it automatically detects and aligns faces, represents image pairs as vectors based on passed facial recognition models, and finds distances between representations based on a passed distance metric. Running the face recognition task on DeepFace was also incredibly computationally efficient as it extracted and stored the representations beforehand. Thus, each time we called the face recognition algorithm, if there was a previous reference to either the input or the target

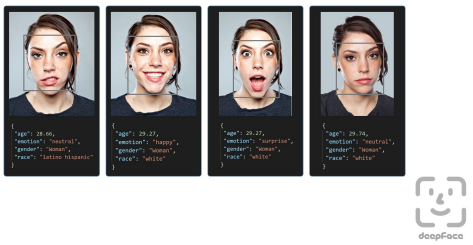

Figure 6: Facial Attribute Analysis Module DeepFace

image, the algorithm would just find embeddings of the target image. However, one significant limitation of the program is that since the output of the recognition algorithm is a pkl file or a Pandas data frame, we are unable to visualize the bounding boxes that are one of the central goals of this assignment. However, to visualize what the face localization function would look like if they had a visual representation of the backend processes, the Demo provides the following Figure: 5. As is evident, the model reads in the input image and finds images within a dataset that has the highest corresponding cosine scores. The pkl file included in my GitHub submission along with the Jupyter Notebook file contains instructions for how to run the DeepFace algorithms.

However, there are critical socio-cultural issues with the DeepFace recognition algorithm. DeepFace offers a facial attribute analysis mode that detects the age, gender, facial expression, and race of the target individual (Reference Figure 6). Although this may seem initially harmless, gender and racial biases in computer vision applications can lead to incredibly harmful consequences for its users. Especially since this project works with deep learning architectures that have opaque backend processes when it comes to identifying variables that lead to biased outcomes, it is critical that computer vision software engineers understand how to minimize a deep learning model's exposure to protected attributes such as race and gender, visual appearances, etc. Thus, when working with the DeepFace modules, although I was impressed with how quickly I can draw relationships between the high-level features of the Gaze Data, I was slightly shocked by the inclusion of this controversial feature that could lead to negative surveillance practices and discriminatory profiling.

6 Conclusion

We presented a systematic literature review to investigate different machine learning applications to perform face detection from eye-tracking data for classification tasks. We used three predominant approaches: The Haar Cascade Baseline Model, FCHD, and DeepFace. Due to time constraints and the failure to determine a good scope of this assignment, in hindsight, we should have definitely focused our efforts on one of the three models instead of distributing our time and resources to configure three. However, we were satisfied with the results of the Cascade Baseline model as to a certain degree, the model was able to detect faces and automate bounding boxes, which was the purpose of this assignment. Given that eye-tracking finds provide substantive information on the responses and actions of the patient, employing these models to conduct computer vision research definitely allowed us to hone our skills as programmers. In future studies, we think it would be beneficial if we performed a review on the usage of features from eye-tracking data across the Gaze Dataset so that we can better understand how to preprocess the data and perhaps actually improve upon the FCHD architecture. The GitHub repository containing my work is included here!

References

Maryam Ameen Sulaiman and Idress Sarhan Kocher. 2022. A systematic review on evaluation of driver fatigue monitoring systems based on existing face / eyes detection algorithms. *Academic Journal of Nawroz University*, 11(1):57–72.

Jia Zheng Lim, James Mountstephens, and Jason Teo. 2022. Eye-tracking feature extraction for biometric machine learning.

Sefik Ilkin Serengil and Alper Ozpinar. 2020. Lightface: A hybrid deep face recognition framework. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 23–27. IEEE.

Aditya Vora and Vinay Chilaka. 2018. Fchd: Fast and accurate head detection in crowded scenes.

Tuan-Hung Vu, Anton Osokin, and Ivan Laptev. 2015. Context-aware cnns for person head detection. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2893–2901.