High Accuracy Eye Tracking to Encode Facial Variability

Kathy Choi
katchoi@mit.edu

Sophia Zheng
sjzheng@mit.edu

Abstract

Eye tracking in computer vision refers to the process of detecting and measuring the movement of the eyes and/or gaze direction of a person using a camera or specialized eye-tracking hardware. This paper introduces a novel eye tracking algorithm to improve accuracy from current methods. Accuracy is measured in centimeters by the absolute difference between the distance of the estimated and actual gaze position on a tablet screen. The multistage pipeline consists of first preprocessing data to extract features (facial/eye regions) relevant to direction of gaze from the images, and then training the CNN-based model on those features. We experiment with removing features through an ablation study, adding an eye grid feature, concatenating the two eye features, and changing the learning rate and batch size. Leveraging multiple fully connected convolutional neural networks and max pooling layers on left eye, right eye, face, and face mask features, our project found that the best model was with a batch size of 128, learning rate of 0.0001, and no eye features, resulting in a test error of 1.6385. While concatenating the left and right eye features and adding an eye grid feature did not seem to have a significant effect on the accuracy, excluding the eye features all together improved the validation error.

1. Introduction

Advances in eye tracking technology has applications spanning various fields such as human-computer interaction, psychology, marketing research, and automotive safety. For example, in human-computer interaction, gaze tracking has facilitated the development of intuitive interfaces, enabling users to control their technological experience with natural eye movements and emotion recognition. In medical applications, eye tracking is able to identify abnormalities in eye movement that may be associated with risk of psychiatric and neurological conditions. [11] [5] Generally, eye tracking technology is composed of these main steps: eye detection, feature extraction, tracking algorithms, gaze estimation, and calibration. Eye Detection within an image or video frame composes the dataset. Feature extraction characterizes attributes of the eyes (i.e. position). Tracking Algorithms estimate the motion of such extracted features across frames. Gaze Estimation combines the eye tracking data with calibration models (as described in the next step) to allow for inference of the direction of gaze relative to the observer's environment based on the position and movement of their eyes. A calibration step establishes a mapping between eye movements and specific points or regions of interest in the observer's field of view.

Although there are existing models that perform gaze estimation, current existing gaze estimation techniques (especially when relying on appearance from images and videos) experience performance degradation in uncontrolled environments without accounting for different lighting and individual facial attributes, making the training dataset extremely important in this topic. [6] [10] Moreover, the best models still have validation errors of over 2 cm, which in a medical or research application, can be quite significant. In our model, we aim to improve the accuracy of learning algorithms within Gaze Estimation by accounting for this variability with changes to the model.

Furthermore, methods of eye tracking are not accessible to everyone. Many existing methods are model-based, meaning that it uses the eye's geometric features such as pupil position and eyeball shape [2]. However, this requires expensive hardware or compute-intensive software. We instead rely on an appearance-based method that solely uses images of faces, and is thus much more readily accessible.

In this paper, we improve the performance of current eye tracking algorithms by training a novel algorithm, a convolutional neural network (CNN) learned end-to-end for gaze prediction. We run experiments on changes such as omitting certain features, adding an eye grid feature, adjusting hyperparameters, and concatenating eye features before convolving, to test which changes improve the accuracy of our model.

2. Related Work

In prior work, gaze estimation has long been leveraged to analyze information about the user in societal, medical, and advertising contexts.

One established framework is OpenCV's provided algo-

rithm for extracting faces and eyes from images or videos. [1] The pre-trained Haar cascades for face and eye detection are capable of detecting objects in images by analyzing their features efficiently. OpenFace's open-source framework also utilizes CNN for face recognition, facial landmark detection, and facial attribute analysis.

Also promising, the MPIIGaze dataset and model developed by the Max Planck Institute for Informatics for gaze estimation analyzes the direction of a person's gaze based on their eye images or facial features. [7] This model leverages multimodal convolutional neural networks (CNNs) to locate landmarks in the input image which learns a mapping from head pose and eye imaging to gaze directions.

We improve upon the iTracker model proposed in Eye Tracking for Everyone. [4] This original algorithm uses the right eye image, left eye image, face image, and face grid (a boolean mask indicating the location of the face in the frame) as features. The model is made up of eight convolutional neural network (CNN) layers, four for the eye feature pathway and four for the face feature pathway. Both the left and right eyes share weights, resulting in eight unique CNNs total. The outputs of the left eye pathway and right eye pathway are then combined through a fully connected layer. The output of this layer is then combined with the outputs of the face pathway with two additional fully connected layers.

We ran ablation studies first to analyze the importance of each component in the original algorithm. This study led us to remove the eye features from the final model, which is different from existing models. Motivated by the face grid feature in the iTracker model, we add an eye grid feature as well. Then, we improved upon the algorithm by choosing to concatenate the left eye and right eye features before training them through the CNN layers. More insight into this will be in Section 4 Methodology.

3. Dataset

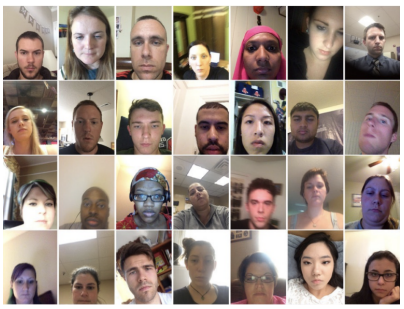

Figure 1. Sample frames included in the dataset with varied lighting, head positions, and facial features

For a robust framework, our experiments will need to be conducted on diverse datasets to accurately track eye movements across different users and environmental conditions. This includes a variety of lighting conditions, object environments, image resolution, and user demographics. Thus, we utilize the GazeCapture dataset, a crowdsourced dataset including 1474 subjects and over 2 million face images covering large variability in pose, appearance, and illumination. Each data sample contains 4 features: left eye, right eye, face, and face mask, in addition to the true labels. Because of computational limitations, we trained on a subset of the original data with 48000 training samples and 5000 validation samples.

To preprocess the data, we split the original training dataset into a 80/20% training/testing split. This split was created using scikit-learn's `train_test_split` function with a deterministic random seed. Ultimately, we trained on 38400 samples, validated on 5000 samples, and tested on 9600 samples.

4. Methodology

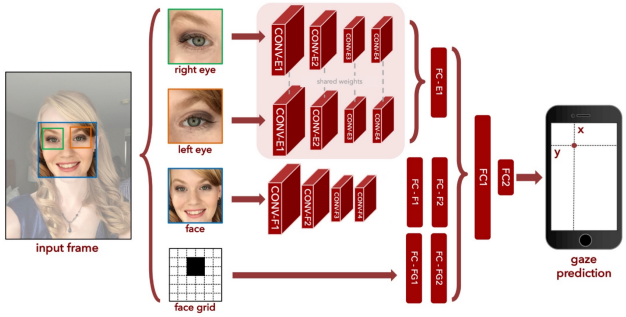

Figure 2. The model flow for the original iTracker

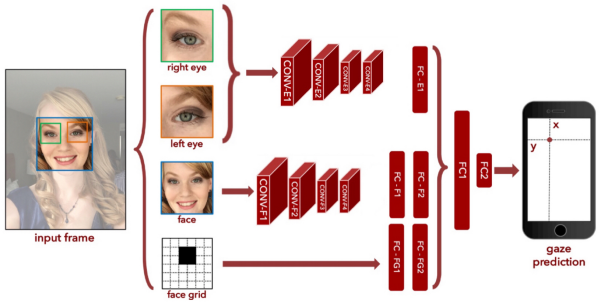

Figure 3. The model flow for our augmented iTracker with left and right eye concatenation before convolving

The proposed system addresses several key challenges in eye movement tracking, including robustness to various lighting conditions, occlusions, and head movements. By integrating a multi-stage pipeline to train and test our data, we create a model composed of convolutional neural networks and max pooling layers, fully connected layers, and an activation function of ReLU.

We conducted 4 different experiments in order to select the best performing model:

1. Ablation study: Removing Features
    (a) Face
    (b) Face grid
    (c) Eyes
2. Concatenation of left and right eye features before being fed into the CNNs (before CNN layer #1 instead of in the FC layer)
3. Adding a eye grid feature (boolean mask to indicate location of eyes within face)
4. Tuning hyperparameters
    (a) Learning rate: 0.01, 0.001, 0.0001, 0.00001
    (b) Batch size: 32, 64, 128, 256

We first evaluate the performance of the baseline iTracker model. Next, we analyze an ablation study to understand the significance and contribution of different features in the model. [3] Then, we investigate the performance impact of changing the model architecture by concatenating the two eyes before convolving them, as opposed to concatenating the outputs of the two pathways for the two eyes in its respective fully connected layer. Third, we analyze whether adding a eye grid feature improves the accuracy of the model. Finally, we tune the hyperparameters, learning rate and batch size, to improve the performance.

We split the dataset into training and validation sets, train each model on the training set, and compare the different models by evaluating it on the validation set. We evaluate the performance under these various experiments by comparing the validation error, given in centimeters by the absolute difference between the distance of the estimated and actual gaze position.

The original baseline iTracker model was tested with parameters `max_epoch=50`, `learning_rate=0.001`, `batch_size=32`, and `patience=5`. Patience allows for early stopping, setting the number of epochs to wait before stopping training if validation loss does not improve, allowing us to gauge how many epochs each model took until convergence. The model extracts left eye, right eye, face, and face mask features from the input data. The eye input pathway consists of four unique convolutional layers (conv2d) followed by max-pooling layers (maxpool2d). Each left/right eye sample is convolved through 4 layers, sharing weights. Then, the face sample is convolved through another 4 unique layers. After the convolutional layers, each pathway is flattened and passed through fully connected layers. Next, the outputs from the left and right eyes are concatenated before passing through the fully connected layer. For the face features, there are two fully connected layers (fc_face and fc2_face) before concatenating with the other pathways. The face mask samples are also passed through two fully connected layers (fc_face_mask and fc2_face_mask). ReLU activation functions are used after each convolutional and fc layer to introduce non-linearity to the model. The outputs from the fc layers (eye, face, and face_mask) are concatenated together before passing through more fully connected layers (fc and fc2) to produce the final output which is the predicted distance, in centimeters, from the camera. This can be visually demonstrated in Figure 2.

The ablation study aims to solve the open question of whether a "full-face approach or eye-only approach will obtain a better performance." [8] We systematically remove one feature at a time (eyes, face, face grid) to investigate its impact on the model. Intuitively, it seems that either the face or eyes feature may be redundant as both include information about the eyes and may introduce excess noise into the model or lead to overfitting. We expect the face grid feature to have a significant impact on the performance of the model, as the location of the head is very relevant to the direction of gaze and is not correlated with other features.

Concatenating the left and right eye features originally occurred after each sample was passed in its respective pathway into the four convolutional layers in the first fully connected layer step. This experiment proposes concatenating the left and right eyes in the very beginning, so that only one combined feature is convolved, visually demonstrated in Figure 3, halving the original feature size. In the fully connected layer of the eye pathway, concatenation is no longer necessary. The intuition behind this is to augment the data as a more comprehensive representation of the input data by training on the combined features with more information. Intuitively, this should preserve the spatial relationships between both eyes. Another consideration is symmetry/asymmetry between both eye regions. [9] Concatenation can allow the model to capture both of these patterns simultaneously, extracting more information. With regards to efficiency, concatenating the features could reduce redundancy during feature extraction. By learning from both eyes together instead of performing separate convolutions for each eye, the network can be more efficient and extract more relevant features.

We then experiment with adding an eye grid feature, similar to the face grid feature, but indicating the location of the eyes within the face as a boolean mask. The location of the eyes provides important information regarding head pose, which is correlated with gaze. For example, a significant difference in the size of the eye regions may indicate that the face is tilted to one side. One difficulty comes from the fact that the dataset only includes the extracted features, and not the raw image. Therefore, in order to generate the eye grid feature, we use OpenCV's Haar feature-based cascade classifer to get bounding boxes for the eyes from the

face image feature. We then line up the image with the face grid feature, and mark the pixels within the bounding boxes. Similar to the face grid, the eye grid is fed through two fully connected layers before being concatenated with the outputs from the other features.

Finally, we tune hyperparameters by trying 4-5 different values, and graphing the training and validation error for each of the values. We use the value that minimizes the validation error, keeping in mind that a low training error and high validation error may be indicative of overfitting.

5. Experimental Results and Discussion

All experiments were trained on 38400 samples of the GazeCapture dataset and validated on 5000 samples, as described in the Section 3.

After running all experiments, the model had improved performance from some adjustments and no significant difference in performance for other adjustments.

Our baseline model with parameters `max_epoch=50`, `learning_rate=0.001`, `batch_size=32`, and `patience=5` performed with a validation error of 2.0776, converging after 41 epochs. All other results were analyzed in comparison to this baseline.

Table 1. Ablation Study on Different Features

Feature Removed	Validation Error after 50 Epochs
None (Baseline)	2.0776
Face	2.2199
Face Grid	1.9662
Eyes	1.7216

As seen in Table 1, removing the face feature worsened the validation error in comparison to the baseline. This makes sense because the face feature encompasses both eye appearance and head pose information. Removing the face grid resulted in similar baseline accuracy. Removing the eye features significantly improved the validation error. This confirms our hypothesis about the redundancy of this feature, as eye information is already included in the face feature.

Table 2. Validation Error and Number of Epochs to Convergence by Batch Size

Batch Size	32	64	128	256
Validation Error	2.0776	1.8283	1.7311	1.8203
Number of Epochs until Convergence	41	32	36	36

For the batch size experiment, we kept `max_epoch=50`, `learning_rate=0.001`, and `patience=5` constant, while varying the batch size.

Figure 4. Line graph of the validation error and training error of different batch sizes

As seen in Table 2, testing various batch sizes achieved the best performance and lowest training and validation error with a batch size of 128. Intuitively, a 128 batch size improved from the baseline of 32 because using larger batch sizes allow for better generalization, allowing the model to perform better on unseen data by learning from more robust features in larger batches. They can also smooth out noise to the model's parameters that may occur with smaller batch sizes. Larger batch sizes also took less epochs until convergence, improving efficiency.

For the learning rate experiment, we kept `max_epoch=50`, `batch_size=32`, and `patience=5` constant, while varying the learning rate.

Table 3. Validation Error and Number of Epochs to Convergence by Learning Rate

Learning Rate	0.01	0.001	1e-4	1e-5
Validation Error	6.8358	2.0776	1.7843	2.3872
Number of Epochs until Convergence	3	41	32	31

Testing various learning rates achieved the best performance and lowest training and validation error with a learning rate of 0.0001 as demonstrated in Table 3. This model's architecture requires a baseline learning rate of 0.001, because as seen with the 6.8358 validation error, 0.01 is too large of a learning rate, causing the loss function to consistently oscillate and never converge to the minimum. We believe that in our case, a smaller learning rate of 0.0001 is able to perform the best because of the nature of our dataset. Its high degree of variability, purposely captured to improve current eye tracking models in emulating real life conditions, requires that the model learn more slowly and carefully from the data. On the other hand, 0.00001 proved to be too small, causing the model to converge too slowly.

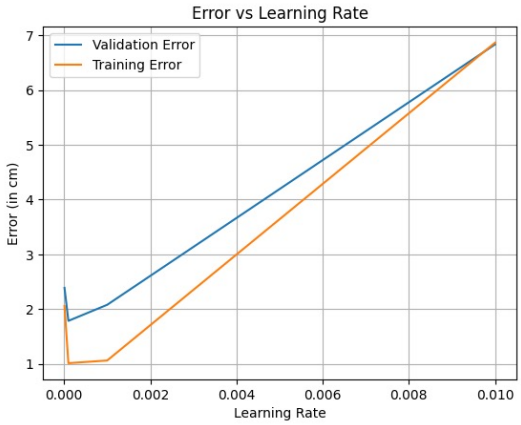

Figure 5. Line graph of the validation error and training error of different learning rates

Adding the eye grid feature did not improve the accuracy of our model. With the addition of this feature (keeping all the hyperparameters the same as the baseline), the model achieved an accuracy of 2.3297, which is worse than the baseline accuracy of 2.0776. Upon analysis of the extracted eye grid features, it seemed that the cascade classifier was unable to identify one or both eyes from the face images. Thus, we modified the code to only include images where both eyes were identified. The model achieved a slightly better accuracy of 2.2145 with this change, but still worse than the baseline model. The lack of improvement may be attributed to the redundancy of the eye information, overfitting, and poor accuracy of the cascade classifier. The bounding boxes for the eyes returned by the classifier were quite large, resulting in feature data that lacks precision.

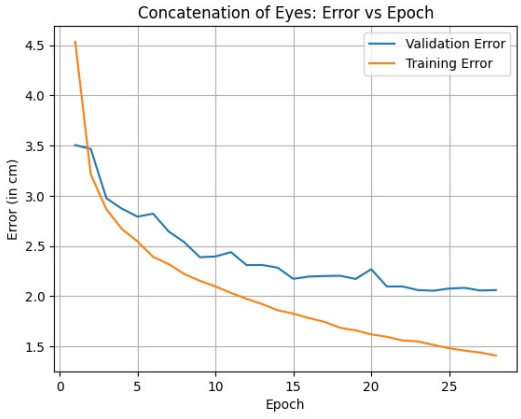

Figure 6. Line graph of validation error and training error across epochs until convergence for prior eye concatenation

Concatenating the left and right eye features did not seem to have a significant effect with initial concatenation producing a validation error of 2.0609 and later concatenation (baseline) of 2.0776. This could be explained by the fact that the left and right eye feature regions may contain redundant information. Thus, the step in which they're concatenated would not significantly alter the representation learned by the model. Additionally, the specific architecture of our model including the layered CNNs, max pooling layers, and ReLU activation functions may not be affected by concatenation of features at different stages.

For our final model, we combined all of the improvements. We excluded the eye features, and trained with `max_epoch=50`, `learning_rate=0.0001`, `batch_size=128`, and `patience=5`. This achieved a test error of 1.6385, and converged after 33 epochs, which are both better than that of the baseline model.

6. Conclusion and Impact

In conclusion, this paper aims to improve the accuracy of the iTracker gaze estimation algorithm through experiments on removing features, concatenating left and right eye features, adding an eye grid feature, and tuning learning rate and batch size.

Through an ablation study, we identified the significance of different features in the model, discovering that while removing the eye features improved accuracy, the face was crucial for maintaining performance. Moreover, neither the concatenation of left and right eye features nor the addition of an eye grid feature had a significant impact on the model's performance. Increasing batch size and lowering learning rate, on the other hand, did result in a significant improvement in validation accuracy.

Overall, this research contributes to advancing the field of gaze estimation by providing insights into the importance of different features and model architectures, accounting for the current issue of training on not diverse datasets that leads to poor performance. Unlike existing models, our final model relies only on the face features, instead of the eyes. Future work could focus on further refining the model architecture (adding additional layers), exploring alternative datasets, and testing in real-world scenarios.

7. Contributions

Both Kathy and Sophia worked together to research existing methodologies and propose experiments. Sophia worked on preprocessing the data and adapting the original algorithm including adjusting the original script, updating tensorflow code, splitting the data into training/validation/test subsets, and the experiment concatenating the two eye features before feeding the CNNs. Kathy worked on the ablation study, the experiment adding the eye grid feature, and generating plots. Both authors worked equally on tuning the hyperparameters, the writing of the paper, and the creation of the presentation.

References

[1] Chandan C Bagan Dhanraj K Shyamala G Abhaya V, Akshay S Bharadwaj. Eye-move: An eye gaze typing application with opencv and dlib library, 2022. 2

[2] Yihua Cheng, Haofei Wang, Yiwei Bao, and Feng Lu. Appearance-based gaze estimation with deep learning: A review and benchmark, 2024. 1

[3] Feng Shi Xiaoyan Guo Pengfei Wan Jinchao Zhou, Guoan Li and Miao Wang. Em-gaze: eye context correlation and metric learning for gaze estimation, 2023. 3

[4] Petr Kellnhofer Harini Kannan Suchendra Bhandarkar Wojciech Matusik Kyle Krafka, Aditya Khosla and Antonio Torralba. Eye tracking for everyone, 2016. 2

[5] M. E. Jadhav P. A. Punde and R. R. Manza. A study of eye tracking technology and its applications, 2017. 1

[6] Pinyan Tang Ruijie Zhao and Sihui Luo. Improving domain generalization on gaze estimation via branch-out auxiliary regularization, 2024. 1

[7] Pavlo Molchanov Umar Iqbal Seonwook Park, Shalini De Mello and Jan Kautz Otmar Hilliges. Few-shot adaptive gaze estimation, 2019. 2

[8] Mario Fritz Xucong Zhang, Yusuke Sugano and Andreas Bulling. It's written all over your face: Full-face appearance-based gaze estimation, 2023. 3

[9] Feng Lu Yihua Cheng, Xucong Zhang and Yoichi Sato. Gaze estimation by exploring two-eye asymmetry, 2020. 3

[10] Yu Yu and Jean-Marc Odobez. Unsupervised representation learning for gaze estimation, 2020. 1

[11] Gianpaolo Zammarchi and Claudio Conversano. Application of eye tracking technology in medicine: A bibliometric analysis, 2021. 1