

---

# Implementing an Eye Tracker with Modern Computer Vision Techniques

---

**Anish Sahoo**

CS 4973

Northeastern University

Boston, MA 02115

sahoo.an@northeastern.edu

**Sharon Pan**

CS 4973

Northeastern University

Boston, MA 02115

pan.sha@northeastern.edu

## Abstract

We attempt to create an eye tracker model that can run fast enough on consumer hardware on a common webcam feed. Our approach employs a dual branched convolutional neural network for encoding each individual eye, and a fully connected layer to encode head-pose information. The three feature vectors are combined to predict gaze pitch and yaw using a regression head, resulting in a model with 1.2M parameters that can be easily run on consumer hardware. This model has been trained on the MPIIGaze dataset, achieving a mean angular error of  $1.44^\circ$  and demonstrating strong accuracy relative to SOTA models. We introduce a 25 point calibration procedure to translate these gaze predictions to screen coordinates reliably. Together, this model and inference pipeline enable accurate and accessible gaze tracking, opening the doors to applications in accessibility, human-computer interaction, and automobile safety.

## 1 Introduction

In our project, we implemented an eye gaze tracker using modern Computer Vision techniques. This eye gaze tracker is accurate and lightweight, and it can be run on consumer-grade devices, like personal laptops, in real-world settings.

### 1.1 Problem Definition

Studying and analyzing eye gaze patterns has a variety of real-world impacts that are useful for understanding the human mind and thought process. Some situations that heavily involve eye gaze are as follows:

- **Driving:** Real-time gaze tracking can detect driver distraction and drowsiness, potentially preventing accidents. Understanding where drivers naturally focus their attention also gives more insight into designs of safer vehicle interfaces and advanced driver assistance.
- **Games and Virtual Reality:** Eye tracking creates more immersive gaming experiences through gaze-based aiming, environmental interaction, and dynamic narrative elements that respond to what captures the player's attention. Especially for e-sports and game analytics, understanding player gaze patterns reveals skill development and interface effectiveness.
- **Research Studies:** In a field that highly prioritizes evidence, the tools that are used to conduct experiments are important. For research studies that rely on eye trackers as part of their experiment, accurate eye tracking reduces the margin of error and gives more reliable, usable results.

However, eye tracking presents several challenges. The process of ensuring consistent results can be finicky and particular. First, it is hard to generalize eyes because humans have many different eye shapes and colors, and it is expected that an eye tracker should support all of these variations to the same level of accuracy. Furthermore, real-time computation efficiency is needed to make it viable for daily use. In order to tackle this problem, several subproblems need to be addressed, including eye-region detection, mapping eye to screen coordinates, and pupil localization under various lighting conditions, among others. Finally, it is generally known that laptop gaze trackers are not entirely accurate, but is most convenient for the purpose of this project.

## 1.2 Motivation

In terms of motivation for choosing this project, one of the members in this group is currently contributing to a research project that involves tracking eye gaze. The project focuses on understanding how people learn where and when to look; the research question is: how long does it take for participants to predict patterns that occur in terms of timing and location in a fresh task that they have no prior experience in? Having worked with multiple eye trackers, we have experienced firsthand the difficulties and limitations that are associated with eye trackers, even those that are state-of-the-art. With this knowledge, we aspired to create an eye tracker that would mitigate those issues.

## 2 Literature Survey

A literary survey of the articles and research papers used for this project is as follows:

1. Zhang, X., Sugano, Y., Fritz, M., Bulling, A. (2015). Appearance-based Gaze Estimation in the Wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4511-4520). <https://arxiv.org/abs/1504.02863>
  - Zhang and co introduced MPIIGaze, a large-scale dataset of 213,000+ eye images captured during natural laptop use, which has become the standard benchmark for training and evaluating appearance-based gaze estimation methods using standard RGB cameras rather than specialized infrared equipment.
  - This dataset provides an excellent collection of gaze data, which we used in our project to train our model.
2. Mesmoudi, S.; Hommet, S.; Peschanski, D. Eye-Tracking and Learning Experience: Gaze Trajectories to Better Understand the Behavior of Memorial Visitors. *J. Eye Mov. Res.* 2020, 13, 1-15. <https://doi.org/10.16910/jemr.13.2.3>
  - Mesmoudi and co's paper focuses on using eye-tracking as a tool to quantify visitor experience and behavior in museums, as well as tracking how museums can have a role in learning among students.
  - This paper informed our understanding of how gaze trackers can be applied in real-world settings with practical implications. The variables introduced in this work (gaze trajectory and viewing time) provide valuable metrics for analyzing eye tracking data in applied contexts.
3. Zhiwei Zhu and Qiang Ji, "Eye gaze tracking under natural head movements," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 918-923 vol. 1, doi: 10.1109/CVPR.2005.148.
  - A challenge with standard PCCR eye-tracking techniques is that the user needs to keep their head still for an uncomfortable amount of time in order to collect the most accurate results. Zhu and co's paper outlines a solution that allows the user to move their head freely and only requires one calibration procedure during the entire session rather than every time the head moves, making it more efficient.
  - We implemented our gaze tracker with this solution in mind, incorporating head pose estimation to make our model more user-friendly and robust to natural head movements.
4. Rodrigues, J.A., Vieira de Castro, A., Llamas-Nistal, M. (2025). Integrating Eye-Tracking, Machine Learning, and Facial Recognition for Objective Consumer Behavior Analysis. In: Schmorow, D.D., Fidopiastis, C.M. (eds) *Augmented Cognition. HCII 2025. Lecture Notes in Computer Science()*, vol 15778. Springer, Cham. [https://doi.org/10.1007/978-3-031-93724-8\\_5](https://doi.org/10.1007/978-3-031-93724-8_5)

- This paper introduces ML-based methods to use real time facial data and gaze data to learn about user fixations in certain areas of static images. Their approach to collecting real-time data and computing gaze information informed our inference pipeline design.
  - This work is particularly relevant due to its focus on improving the scalability, affordability, and accessibility of webcam-based eye tracking. By enabling gaze tracking from users’ own laptops, it aligns with our goal of targeting the general public with accessible technology.
5. Saxena, S., Fink, L.K. & Lange, E.B. Deep learning models for webcam eye tracking in online experiments. *Behav Res* 56, 3487–3503 (2024). <https://doi.org/10.3758/s13428-023-02190-6>
    - This paper explores gaze detection using webcam data, and outlines several methods and benchmarks to train and evaluate a gaze detection model.
    - We considered their reported roadblocks and shortcomings when designing our system, incorporating architectural improvements to address these challenges.
  6. Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., & Torralba, A. (2016). Eye tracking for everyone. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2176-2184). <https://arxiv.org/abs/1606.05814>
    - This paper introduces a massive dataset of over 2.5m frames of gaze data from over 1400 people, and introduces a simple baseline training approach for eye tracking.
    - We drew from the lessons in this literature to create a model that incorporates the benefits of these approaches while training and running efficiently on modern consumer devices.

### 3 Methods

Our approach implements a system for gaze tracking that first trains a deep learning model on a large dataset to learn general patterns of eye appearance and gaze direction, then personalizes the model for individual users through calibration. Our complete pipeline includes three key stages:

1. Train CNN-based model based on gaze dataset
2. Calibrate model to display by training a regression function to convert model output into pixels on our screen
3. Create a real-time inference pipeline that continuously captures webcam video and displays where the user is looking

Our architecture builds on baseline single-eye approaches through a few key improvements:

1. Left and right eye images can be processed simultaneously through the use of parallel CNN branches, which improves efficiency
2. Head pose estimation is able to decouple head movement from eye movement, which makes this process reliant on one less factor

#### 3.1 Training CNN-based model

The core of our system is GazeNet, a dual-eye convolutional neural network that predicts gaze direction from bilateral eye images and head pose information. The architecture consists of 3 main components: eye feature extraction networks, head pose embedding, and gaze prediction fusion layers.

Each eye is processed by an identical convolutional neural network called EyeCNN that extracts appearance features from normalized eye crops. The input is a grayscale eye image of size 36x60 pixels, normalized to the range [-1, 1]. The architecture contains 3 convolutional layers with 32, 64, and 128 filters respectively, each followed by ReLU activation and 2x2 max pooling. After the third convolutional layer, the output is flattened and passed through a fully connected layer that produces a 128-dimensional feature vector. This design progressively increases feature depth while reducing spatial dimensions, learning hierarchical representations from low-level edges to high-level gaze patterns.

Head pose information is encoded through a simple multi-layer perceptron that takes a 2-dimensional vector containing pitch and yaw angles in radians and maps it to a 32-dimensional embedding. This lightweight embedding captures head orientation information with minimal computational cost.

The complete GazeNet model takes 3 inputs: left eye image ( $1 \times 36 \times 60$ ), right eye image ( $1 \times 36 \times 60$ ), and head pose vector (2D). Each eye is processed through its own EyeCNN to produce 128-dimensional features, the head pose is embedded to 32 dimensions, and all features are concatenated into a 288-dimensional vector. This combined representation passes through two fully connected layers ( $288 \rightarrow 256 \rightarrow 2$ ) to predict gaze pitch and yaw in radians. The total model contains approximately 1.2M trainable parameters.

We train on the MPIIGaze dataset containing 213,658 labeled samples from 15 participants captured during natural computer use. Each sample provides eye regions, ground truth gaze direction, and head pose angles. Our preprocessing pipeline crops the eye regions with 15% padding, converts to grayscale, resizes to  $36 \times 60$  pixels, and normalizes to  $[-1, 1]$  range. We use a 90% training and 10% validation split with random sampling.

Training uses mean squared error loss between predicted and ground truth gaze angles, optimized with AdamW at a learning rate of  $5e-4$ . We use a batch size of 128 and train for 50 epochs with cosine annealing learning rate scheduling.

We evaluate performance using angular error loss. We report both mean and median angular error, as median is more robust to outliers.

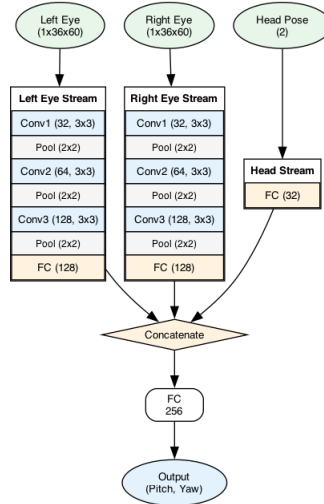


Figure 1: GazeNet Architecture

### 3.2 Screen Calibration

After training produces a general gaze estimation model, we move on to a calibration stage to map predicted gaze angles to user-specific screen coordinates. This personalizes the system to individual screen size, camera position, and biometric differences without needing model retraining.

The calibration process presents the user with 25 calibration points arranged in a  $5 \times 5$  grid that covers the screen size. For each point, a large red circular target is displayed and the user presses the spacebar when ready, which turns the target green to indicate activity. The system then takes about 2 seconds to capture webcam frames while the user maintains their seating and viewing position. For each frame, the system detects the face, extracts eyes, estimates head pose, and predicts gaze angles. The predictions are averaged to produce robust mean pitch and yaw values, significantly reducing frame-to-frame noise. The system stores the mapping from averaged gaze angles to the known screen coordinates. This process takes 1-2 minutes to complete.

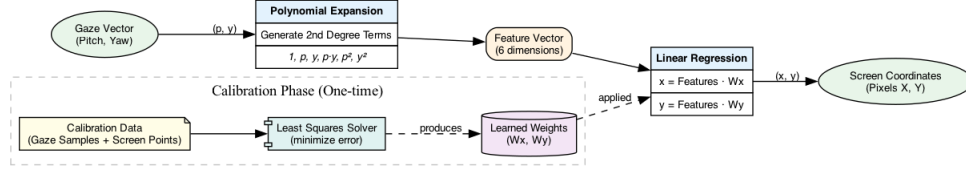


Figure 2: Calibration Model

Calibration coefficients are saved and can be reloaded for following sessions. Users should recalibrate when they change seating position, adjust monitor configuration, move the camera, or notice reduced accuracy. The quick 1-2 minute recalibration time makes it practical to recalibrate as needed.

### 3.3 Inference Pipeline

We have also implemented a complete pipeline for continuous real-time gaze tracking. The pipeline begins by continuously capturing frames from the webcam using OpenCV. Each frame is converted appropriately from BGR to RGB and processed through MediaPipe FaceMesh to detect 468 facial landmarks. If no face is detected, the system displays a warning and continues to the next frame.

When a face is detected, we find eye regions using specific landmark indices around the left and right eyes. We then apply the same preprocessing procedure used during training, in which we compute a bounding box from the landmark positions, add 15% padding, crop the region, convert to grayscale, resize to 36×60 pixels, normalize to  $[-1, 1]$ , and convert to PyTorch tensors.

Furthermore, we estimate head pose simultaneously using a 6-point face model (nose tip, eye corners, mouth corners, chin) and OpenCV’s solvePnP algorithm. The resulting rotation matrix is converted to head pitch and yaw angles in radians

With preprocessed eye images and head pose ready, we perform neural network inference. The left eye, right eye, and head pose tensors are passed through GazeNet with gradient computation disabled. The model returns predicted gaze pitch and yaw in radians. If calibration parameters are loaded, we apply the learned linear transformation to map these angles to screen coordinates using  $x_{\text{screen}} = a_x \times \text{pitch} + b_x \times \text{yaw} + c_x$  and similarly for  $y$ . The coordinates are clamped to screen boundaries.

For visualization, we scale the screen coordinates to frame coordinates and draw a green filled circle with red outline at the predicted gaze location. Text overlays display the raw gaze angles and mapped screen coordinates. The annotated frame is then displayed in a window using OpenCV.

The complete pipeline runs in real-time on consumer hardware, achieving frame rates sufficient for interactive applications. The lightweight architecture enables deployment on standard laptops without requiring specialized GPU acceleration.

## 4 Experiments

Our experiment revolves around a few key questions:

1. How can we train the most efficient model while also maintaining accuracy? What architecture and implementation choices do we make?
2. How can we minimize user configuration requirements to improve accessibility, comfort, and ease of use?
3. Can the complete pipeline run in real-time on consumer-grade devices?

We conducted training locally on our personal Apple Macbook laptops with 14 cores and 24GB of RAM. Furthermore, inference and real-time testing were also performed on the same computers. We have implemented our system in PyTorch 2.9.1 with OpenCV for image processing and MediaPipe for facial landmark detection.

#### 4.1 Implementing The Model

At first, we attempted to use separate models for each eye and then combine the two image regions together. There seemed to be many benefits to this approach: the ability to train and test each model independently, potentially smaller models, and easier debugging.

Then through our research, we were introduced to an interesting approach that involves using shared neural networks to train a model. This claim was supported by the domain experience of one of the authors of this paper, who has used this technique in training CNN-based multi-agent reinforcement learning models. Upon experimenting with different implementations of this approach, we discovered that it performed significantly better than independent models.

The problem with the separate models approach was that since both models would need to be trained independently, the training time would be doubled. Furthermore, the model would lose important information about the relationship between the two eyes. In the case of the combined approach, not only does the model gain twice the information at once to understand how the two eyes work together and is able to cross-check across both eyes, but it also reduces redundant learning.

Therefore, we decided on a combined 'GazeNet' model with a CNN for each eye, respectively.

#### 4.2 Hardware

We have built our eye gaze tracking system to be lightweight, so that it can run on consumer hardware without the need for external GPUs. The speed of our system comes from a variety of implementation choices, the main ones being: our model is relatively small with about 1.2M parameters, we have transformed our dataset by cropping images. This model can be trained on just CPUs, Apple Metal (MPS) GPUs and CUDA-based GPUs, and is optimized for performance in all scenarios.

Due to the small number of parameters, this model can easily be run on anything from microcontrollers and single-board computers (SBCs) to proper desktop computers and laptops. This was one of our goals, as some major applications for gaze tracking include automotive and safety, which often rely on SBCs and microcontrollers.

#### 4.3 Observations and Results

We experimented with different training configurations to optimize our model's performance, comparing different optimizers (SGD, Adam, AdamW) and different loss functions (MSE and angular loss). By comparing performance across all combinations, we found that AdamW combined with angular loss achieved the best results.

The hyperparameters used for training are summarized in Table 1.

Table 1: Hyperparameters used for training the GazeNet model.

Hyperparameter	Value
Batch Size	128
Learning Rate	$5 \times 10^{-4}$
Epochs	42
Optimizer	AdamW
Weight Decay	$5 \times 10^{-4}$
Gradient Clipping Norm	0.5
LR Scheduler	Cosine Annealing
Loss Function	Angular Loss
Validation Split	0.1
Evaluation Interval	2 epochs

Our dual-eye GazeNet architecture achieved a mean angular error of  $1.44^\circ$  on the validation set, compared to the MPIIGaze baseline of  $4.5^\circ$  reported by Zhang et al [1]. The model demonstrates strong accuracy, with 96.81% of predictions falling within  $5^\circ$  error and 99.21% within  $10^\circ$  error.

The training converged smoothly over 42 epochs, with training loss decreasing from 0.200 to 0.011 and validation loss from 0.165 to 0.025. The close tracking between training and validation curves indicates the model produced good accuracy on the dataset without significant overfitting.

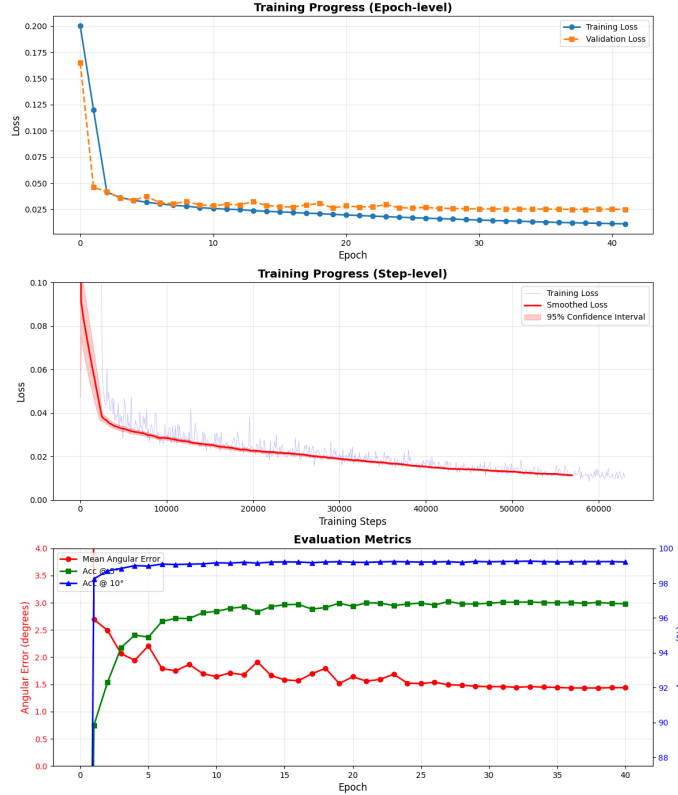


Figure 3: Training Metrics

Furthermore, we found success with running the calibration so that the eye tracker works as expected. However, there is still room for improvement. Although frames are collected over 2 seconds to reduce noise, some noise still remains. Not only that, but also the best results currently come from situations where the eyes are close to the screen during calibration, so there are limitations as to where the user can position themselves. Additional experimentation is necessary to determine more effective methods to reduce noise.

## 5 Conclusion

We developed a webcam-based eye gaze tracker achieving  $1.44^\circ$  mean angular error on the MPIIGaze benchmark, demonstrating that appearance-based methods using standard RGB cameras can reach accuracy levels of specified eye trackers. Our key finding is that an architecture that uses both eyes simultaneously outperforms separate single-eye models, achieving 48% error reduction through learned cross-eye relationships—a result that exceeded our initial expectations. Through iterative experimentation, we discovered that training configuration matters significantly, with AdamW and angular loss outperforming other combinations. While some results aligned with expectations, such as the importance of head pose integration and the benefits of large-scale training data, the magnitude of improvement from joint eye processing was surprising and reveals that gaze estimation fundamentally benefits from understanding how eyes work together rather than processing them independently.

Future work should focus on several promising directions, such as further reducing calibration requirements or conducting real-world user studies across diverse populations and environments. Our system contributes to making eye tracking technology more accessible by lowering the cost from thousands of dollars to essentially free, enabling greater access for everyone—from researchers

conducting large-scale studies to developers building gaze-aware applications. By demonstrating that careful architectural design combined with appropriate training strategies can achieve near-commercial-grade accuracy with consumer hardware, we move toward a future where the benefits of eye tracking are accessible to everyone regardless of resources or technical expertise.

## References

- [1] Zhang, X., Sugano, Y., Fritz, M., Bulling, A. (2015). Appearance-based Gaze Estimation in the Wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4511-4520). <https://arxiv.org/abs/1504.02863>
- [2] Mesmoudi, S.; Hommet, S.; Peschanski, D. Eye-Tracking and Learning Experience: Gaze Trajectories to Better Understand the Behavior of Memorial Visitors. *J. Eye Mov. Res.* 2020, 13, 1-15. <https://doi.org/10.16910/jemr.13.2.3>
- [3] Zhiwei Zhu and Qiang Ji, "Eye gaze tracking under natural head movements," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 918-923 vol. 1, <https://doi.org/10.1109/CVPR.2005.148>
- [4] Rodrigues, J.A., Vieira de Castro, A., Llamas-Nistal, M. (2025). Integrating Eye-Tracking, Machine Learning, and Facial Recognition for Objective Consumer Behavior Analysis. In: Schmorow, D.D., Fidopiastis, C.M. (eds) *Augmented Cognition. HCII 2025. Lecture Notes in Computer Science()*, vol 15778. Springer, Cham. [https://doi.org/10.1007/978-3-031-93724-8\\_5](https://doi.org/10.1007/978-3-031-93724-8_5)
- [5] Saxena, S., Fink, L.K. & Lange, E.B. Deep learning models for webcam eye tracking in online experiments. *Behav Res* 56, 3487–3503 (2024). <https://doi.org/10.3758/s13428-023-02190-6>
- [6] Krafska, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., & Torralba, A. (2016). Eye tracking for everyone. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2176-2184). <https://arxiv.org/abs/1606.05814>