WebEyeTrack: Scalable Eye-Tracking for the Browser via On-Device Few-Shot Personalization

Eduardo Davalos[1,*], Yike Zhang[2,*], Namrata Srivastava[3], Yashvitha Thatigotla[3], Jorge A. Salas[3], Sara McFadden[3], Sun-Joo Cho[3], Amanda Goodwin[3], Ashwin TS[3], and Gautam Biswas[3]

[1] Trinity University, San Antonio TX 78212, USA
davalosedu@trinity.edu
[2] St. Mary's University, San Antonio, TX 78228, USA
yzhang5@stmarytx.edu
[3] Vanderbilt University, Nashville, TN 37240, USA

Abstract. With advancements in AI, appearance-based gaze estimation methods have improved benchmark performance, but practical deployment still lags behind commercial eye-trackers. Factors like model size, inference time, and privacy often go unaddressed. Meanwhile, webcam-based eye-tracking methods remain sensitive to head movement. To tackle these issues, we introduce **WebEyeTrack**, a browser-native framework that integrates a novel lightweight gaze model, metric head pose estimation, and on-device few-shot learning with as few as nine per-user calibration samples ($k \leq 9$). WebEyeTrack adapts to new users, achieving competitive performance with an error margin of 2.32 cm on GazeCapture and real-time inference latency of 2.4 milliseconds on an iPhone 14. Our open-source code is available at github.com/RedForestAI/WebEyeTrack.

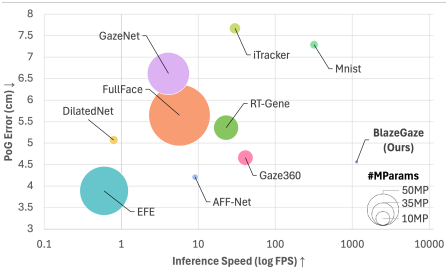
Keywords: Gaze Estimation · Efficient and Scalable Vision · Few-Shot Learning · Head Pose Estimation · On-Device Inference

1 Introduction

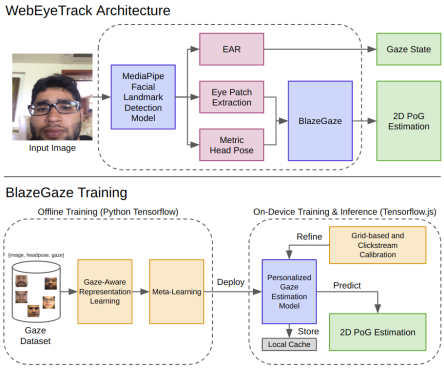
Eye-tracking has been a transformative tool for investigating human-computer interactions, as it uncovers subtle shifts in visual attention [18]. However, its reliance on expensive specialized hardware, such as EyeLink 1000 and Tobii Pro Fusion has confined most gaze-tracking research to controlled laboratory environments [15]. Similarly, virtual reality solutions like the Apple Vision Pro remain financially out of reach for widespread use. These limitations have hindered the scalability and practical application of gaze-enhanced technologies and feedback systems.

To reduce reliance on specialized hardware, researchers have actively pursued webcam-based eye-tracking solutions that utilize built-in cameras on consumer devices. Two key areas of focus in this field are appearance-based gaze estimation

[*] These authors contributed equally to this work.



(a) PoG error vs. inference speed (log-scale FPS).



(b) WebEyeTrack framework overview.

Fig. 1: Efficiency–Accuracy Tradeoff and System Overview. (**Left**) BlazeGaze achieves competitive Point-of-Gaze (PoG) accuracy while delivering orders-of-magnitude faster inference speed compared to prior methods. (**Right**) WebEyeTrack integrates metric head pose estimation, few-shot personalization, and lightweight CNN inference for fully on-device browser deployment.

and webcam-based eye-tracking, both of which have made significant advancements using standard monocular cameras [8]. For example, recent appearance-based methods have shown improved accuracy on commonly used gaze estimation datasets such as MPIIGaze [38], MPIIFaceGaze [39], and EyeDiap [1]. However, many AI models prioritize benchmark accuracy without considering deployment constraints such as display variation, computational efficiency, model size, calibration burden, and cross-user generalization. While some efforts have integrated gaze models into larger eye-tracking systems [15], achieving real-time, fully functional, browser-compatible tracking remains challenging. Retrofitting models that were not designed for these constraints often require substantial optimization and may still fail to meet practical requirements.

At the same time, webcam-based eye-tracking methods have taken a practical approach, addressing real-world deployment challenges [15]. These solutions are often tied to specific software ecosystems and toolkits, hindering portability to platforms such as mobile devices or web browsers. As web applications gain popularity for their scalability, ease of deployment, and cloud integration [27], tools like WebGazer [22] have emerged to support eye-tracking directly within the browser. However, many browser-friendly approaches rely on simple statistical or classical machine learning models [15], such as ridge regression [34] or support vector regression [22], and avoid 3D gaze reasoning to reduce computational load. While these techniques improve accessibility, they often compromise accuracy and robustness under natural head motion.

To bridge the gap between high-accuracy gaze estimation and scalable webcam-based eye tracking, we introduce WebEyeTrack, a few-shot, head-pose-aware solution for the browser (Fig. 1b). WebEyeTrack introduces a metric head pose estimation (via 3D face reconstruction and radial Procrustes analysis) along

with BlazeGaze, a lightweight CNN optimized for real-time inference. It computes the Point-of-Gaze (PoG), detects eye state (open/closed) via eye aspect ratio (EAR) [28] to suppress blink artifacts, and supports continuous clickstream calibration. Continuous clickstream calibration enables the system to adapt to each user over time, using a meta-learning approach in which interaction signals such as clicks progressively personalize the gaze model to changing conditions. We provide both Python and client-side JavaScript implementations to support model development and deployment. In evaluations on standard gaze datasets, WebEyeTrack achieves competitive accuracy with real-time performance on mobile phones, tablets, and laptops (see supplementary materials).
Overall, our work makes the following contributions:

- WebEyeTrack: an open-source novel browser-friendly framework that performs few-shot gaze estimation with privacy-preserving on-device personalization and inference.
- A novel model-based metric head pose estimation method, including a new mathematical formulation and browser-oriented implementation based on face mesh reconstruction and radial Procrustes alignment.
- **BlazeGaze**: A novel, 670KB CNN with a new task-specific architecture and training pipeline inspired by BlazeBlocks, released with trained weights for reproducible real-time deployment on mobile CPUs and GPUs.

2 Background

Gaze Estimation Classical gaze estimation relied on model-based approaches for (1) 3D gaze estimation (predicting gaze direction as a unit vector), and (2) 2D gaze estimation (predicting gaze target on a screen). These methods used predefined eyeball models and extensive calibration procedures [5, 10, 31, 33]. In contrast, modern appearance-based methods require minimal setup and leverage deep learning for improved robustness [8].

The emergence of CNNs and datasets such as MPIIGaze [38], GazeCapture [20], and EyeDiap [1] has led to the development of 2D and 3D gaze estimation systems capable of achieving errors of 6–8 degrees and 3–7 centimeters [38]. Key techniques that have contributed to this progress include multimodal inputs [20], multitask learning [35], self-supervised learning [7], data normalization [36], and domain adaptation [21]. More recently, Vision Transformers have further enhanced accuracy, reducing error to 4.0 degrees and 3.6 centimeters [6]. Despite strong within-dataset performance, generalization to unseen users remains poor [8]. Few-shot learning addresses this by adapting to new users with minimal labeled data. However, the few-shot FAZE framework [25] requires substantial computational resources, including a powerful GPU for real-time inference, while SAGE [14] is a closed-source implementation, restricting evaluation, usage, and practical adoption. Recent methods such as [9] further improve PoG accuracy, often with larger backbones and model complexity; however, without released source code, we cannot benchmark runtime performance and deployability under a shared setup. We therefore compare against publicly available

implementations (or faithful re-implementations) for 2D PoG estimation. 3D gaze estimation methods such as [30] predict gaze direction in 3D space rather than 2D screen coordinates, making them not directly comparable to our 2D PoG formulation.

Metric Head Pose Estimation Head pose is frequently utilized to improve gaze estimation [39], but most methods rely on having access to ground-truth poses obtained from depth sensors or multi-camera setups [1,29]. These setups are often impractical for browser-based applications. Monocular methods tend to struggle with real-time performance and often produce scale-ambiguous poses [3, 13, 19], which renders self-consistent metric head pose estimation unreliable.

Webcam-based Eye-Tracking Deploying deep learning-based gaze systems in real-world applications is challenging due to hardware requirements and software compatibility. Most implementations, such as OpenGaze [37] and FAZE [25], are based on C++ and Python and require PyTorch and CUDA, which are often unsuited for consumer devices. Web-compatible tools like iTracker [20] and iMon [16] are often outdated or poorly documented. Additionally, cloud-based training or inference raises privacy and latency concerns [24]. In contrast, WebEyeTrack runs inference, calibration, and few-shot personalization entirely on-device in the browser, so raw frames, embeddings, and gaze predictions are not transmitted off-device.

Browser-based and Real-time Eye-Tracking Browser-compatible solutions such as RealEye [26], WebET [17], and GazeCloudAPI are closed-source and do not provide transparent benchmarking. Among open-source tools, WebGazer [22] is widely used, utilizing facial landmarks and ridge regression for estimating 2D PoG. However, its accuracy declines significantly over time due to its lack of head pose awareness, resulting in the error increasing from approximately 5 to 10 cm during a 20-minute test [23].

We bridge this gap by integrating fast CNN-based gaze estimation, inspired by BlazeFace's BlazeBlocks [4], with meta-learning for few-shot adaptation and real-time inference. Our model is also head-pose-aware, improving robustness across users and environments.

3 Method

This section discusses WebEyeTrack, a real-time gaze estimation framework designed to function effectively in challenging, real-world environments, such as web browsers and mobile devices. As shown in Fig. 1b, the computational pipeline consists of utilizing MediaPipe's facial landmark detection model to estimate the gaze state (open/close), eye patch region, and metric head pose. These variables are then passed to BlazeGaze, a lightweight 2D PoG estimation model. A two-step training process is used to derive an adaptive gaze meta-learner that is then deployed to user's browsers and perform on-device calibration, inference,

and model caching. We aim to deliver personalized, head-pose-aware gaze predictions with low computational costs.

To accomplish this, WebEyeTrack incorporates two novel components: (1) a model-based 3D face reconstruction pipeline that estimates head pose in metric space from monocular images using our new radial Procrustes formulation and optimization procedure, and (2) an appearance-based gaze estimation model called BlazeGaze, which uses a newly designed lightweight architecture and two-stage training strategy with model-agnostic meta-learning (MAML) to adapt to new users with minimal calibration data. This hybrid approach allows our system to use explicit geometric cues (such as head orientation and position) in conjunction with implicit appearance features (like eye-region encoding), resulting in improved generalization and personalization.

We first describe our novel metric 3D face reconstruction and head pose pipeline in Sec. 3.1. This approach leverages MediaPipe's real-time Facial Landmark Detection model and an iris-based depth estimation algorithm to produce a fully metric head pose through our own mathematical and algorithmic design. We then detail the BlazeGaze model in Sec. 3.2, including its custom architecture, training protocol, and few-shot adaptation strategy that learns a user-specific gaze regressor using only a handful of calibration samples ($k \leq 9$, where k denotes the number of calibration points).

3.1 Metric Face Reconstruction and Head Pose

3D face reconstruction is used to estimate head pose (position and orientation) in metric space rather than relative units. MediaPipe's Facial Landmark Detection model [12] offers a lightweight pipeline for detecting 3D facial landmarks and reconstructing a canonical face mesh. The output UVZ landmark coordinates $\{x_i^I\}_{i=1}^N$ contain X and Y values normalized to screen space, while Z values are relative and scaled using a weak perspective projection. The model also provides a relative face transformation matrix $P = [R \mid t]$, where $R \in \mathbb{R}^{3\times3}$ rotation matrix and $t \in \mathbb{R}^3$ translation vector, which transforms the points from the canonical face coordinate space (FCS) to the camera coordinate space (CCS). The canonical mesh follows a fixed 468-point topology used for consistent face landmark estimation across frames.

In gaze-estimation datasets such as GazeCapture [20], the camera intrinsics are provided, which include the focal length (f_x, f_y) along with the principal point (c_x, c_y). This supports the construction of a camera intrinsics matrix:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The first step is the conversion of facial landmarks from UVZ in image-plane space to XYZ space through reprojection, leveraging the camera intrinsics matrix:

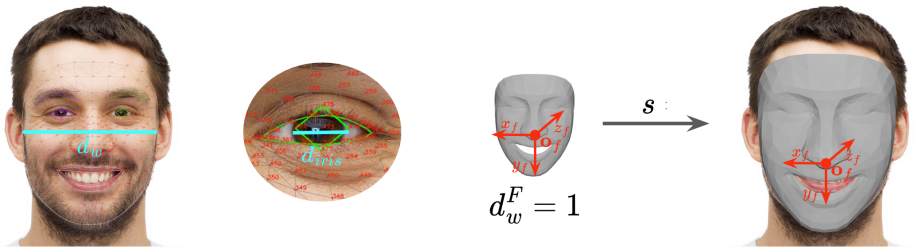


Fig. 2: Iris-based Face Scaling: A constant iris diameter α enables metric face width estimation, which is applied to the normalized face mesh to approximate the user's true facial dimensions.

$$X = \frac{(u - c_x) \cdot z}{f_x}, \quad Y = \frac{(v - c_y) \cdot z}{f_y} \tag{2}$$

To convert all 2D UVZ facial landmarks into 3D XYZ space, we reproject and normalize the points (centered around the nose [ID=4] and with face width set to 1 with leftmost [ID=356] and rightmost [ID=127] landmarks) using the following:

$$\{x_i^R\}_{i=1}^N = \text{Reproject}(\{x_i^I\}_{i=1}^N, K') \tag{3}$$

$$x_i^F = \frac{x_i^R - x_{nose}^R}{\|x_{left}^R - x_{right}^R\|} \tag{4}$$

Face Scale Estimation The normalized XYZ facial landmarks $\{x_i^F\}_{i=1}^N$ are unitless and relative, making them unsuitable for downstream tasks due to inconsistent depth scaling. To recover metric position and orientation (in centimeters), we first estimate face scale. Using a standard iris diameter of $\alpha = 1.2$ cm from gaze estimation literature [32], we compute face width from the pixel ratio between the iris diameter d_{iris}^I to face width d_w^I (Fig. 2). This ratio allows transforming the normalized points into metric landmarks $\{x_i^{F'}\}_{i=1}^N$ via:

$$\{x_i^{F'}\}_{i=1}^N = \alpha \frac{d_w^I}{d_{iris}^I} \cdot \{x_i^F\}_{i=1}^N \tag{5}$$

This approximation may vary across users and camera setups. We adopt a fixed iris constant due to the lack of datasets with ground-truth iris diameter; learned or model-based scale estimation typically requires depth sensing or multi-step per-user calibration, which conflicts with low-latency browser deployment. Under these constraints, the fixed-α assumption is a practical trade-off, with downstream MAML-based personalization mitigating residual scale error. We verify this empirically in the supplement with a sensitivity sweep.

Metric Head Pose With metric facial landmarks, we estimate the 3D metric transformation matrix $P = [R \mid t']$ representing head pose. The rotation matrix R from MediaPipe's relative pose output $P = [R \mid t]$, is scale-invariant and can be reused to estimate the metric translation t'. We propose an iterative refinement algorithm based on our novel radial Procrustes alignment [11] and similar triangles. The goal is to minimize the Euclidean distance between the original UVZ landmarks and the newly projected 3D landmarks under the estimated transformation:

$$t' = \arg\min_{t'} \sum_{i=1}^{N} \|x_i^I - K \cdot [R \mid t'] \cdot x_i^{F'}\|^2 \tag{6}$$

The algorithm initializes with depth $z_0 = 60$ cm, based on average monitor-to-face distance reported in [8], and an initial XY estimate derived by reprojecting the nose's UV position. This step aligns the UV center of projected and original UVZ points, reducing the problem to iterative depth refinement. A lightweight radial Procrustes method determines the direction and magnitude of the depth update by computing a unit direction vector v_i from the center c between matched projected points $\hat{x}_i^I$ and original landmarks x_i^I, using:

$$v_i = \frac{x_i^I - \hat{x}_i^I}{\|x_i^I - \hat{x}_i^I\|} \tag{7}$$

$$z_{\text{update}} = \text{clip}\left(\frac{\beta}{N}\sum_{i=1}^{N} \text{sign}(v_i \cdot c_i) \cdot \|v_i\|, -\Delta_{\max}, \Delta_{\max}\right), \tag{8}$$

where β is a scaling factor of 0.1 and $\Delta_{\max}$ is a threshold cap of 5 cm. A clip function acts as safety guardrails.

After computing and applying z_{update}, the corresponding metric XY values are updated via similar triangles to preserve the reprojection consistency, ensuring the projected nose center $\hat{x}_{\text{nose}}^I$ aligns with the observed nose center x_{nose}^I. The iterative algorithm stops when $z_{\text{update}} < 0.25$ cm or when the iteration cap of 10 is reached.

3.2 BlazeGaze: Lightweight Personalized Gaze

As shown in Fig. 3, BlazeGaze consists of an encoder $\mathcal{E} : I \to \mathbf{z}$, decoder $\mathcal{D} : \mathbf{z} \to \hat{I}$, and gaze estimator $\mathcal{M} : (\mathbf{z}, [R \mid t]) \to \mathbf{g}$. While inspired by BlazeBlocks, this architecture and its training setup are newly designed for browser-scale gaze estimation under strict latency and memory constraints. We adopt a two-stage training strategy (see Fig. 1b). In Stage 1, all components are trained jointly to learn a robust embedding via a standard train/validation split. In Stage 2, the decoder is discarded, the encoder is frozen, and the gaze estimator is adapted into a meta-learner using MAML to enable efficient user adaptation. This separation reduces the number of trainable parameters during meta-learning and avoids instability from joint optimization. To enhance cross-device

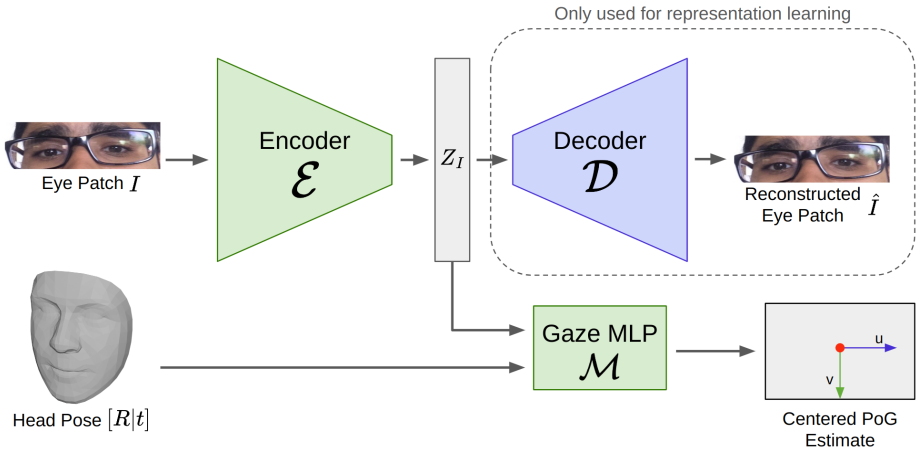


Fig. 3: BlazeGaze Architecture: The BlazeGaze model takes as input the eye patch and the metric head pose and is composed of BlazeBlock-based encoder, decoder, and a gaze multi-layer perceptron (MLP). The decoder is used in representation learning, but is discarded during meta-learning.

generalization (e.g., mobile in GazeCapture vs. laptops in MPIIFaceGaze), gaze predictions are normalized to $\mathbf{g} \in [-0.5, 0.5]^2$ with the origin at screen center, enabling device-agnostic evaluation across varying screen resolutions and sizes.

Gaze-Aware Representation Learning The first-stage of training uses a multi-objective loss function defined as:

$$\mathcal{L}_{\text{all}} = \beta_{\text{r}}\mathcal{L}_{\text{r}} + \beta_{\text{g}}\mathcal{L}_{\text{g}} + \beta_{\text{c}}\mathcal{L}_{\text{c}}, \tag{9}$$

where $\beta = \{\beta_r, \beta_g, \beta_c\}$ have been empirically determined for each dataset, with initial parameters inspired from Park et al. [25]. These loss functions are explained in the following paragraphs.

Image Reconstruction. We train the encoder-decoder using a standard pixel-wise reconstruction loss that minimizes the mean squared error (MSE) between the input image I and its reconstruction $\hat{I}$:

$$\mathcal{L}_{\text{r}}(I, \hat{I}) = \frac{1}{|I|} \sum_{u \in I, \hat{u} \in \hat{I}} (u - \hat{u})^2, \tag{10}$$

where u and $\hat{u}$ represent individual pixel values in the original and reconstructed images, respectively. This training objective encourages the encoder to learn a compact latent space that captures the essential visual structures of the eye region, including features relevant to gaze direction.

Weighted L2 Loss for 2D PoG Estimation. Given ground truth gaze positions $\mathbf{g}_{\text{true}}$ and predicted values $\mathbf{g}_{\text{pred}}$, we compute the weighted L2 loss as:

$$\mathcal{L}_{\text{g}} = \frac{1}{B} \sum_{i=1}^{B} w_i \left\| \mathbf{g}_{\text{pred}}^{(i)} - \mathbf{g}_{\text{true}}^{(i)} \right\|^2, \tag{11}$$

where B is the batch size and w_i is the scalar weight for sample i. The weight is obtained from a precomputed 30×30 grid of inverse frequency for each dataset to address ground truth region imbalance. Additional details about sample weights are provided in the supplementary materials.

Embedding Consistency Loss. To encourage gaze-consistent embeddings, we define a contrastive-style loss that aligns pairwise distances in embedding space with pairwise distances in normalized 2D gaze space.

We first compute the normalized Euclidean distance δ_{ij} between the PoG gaze $\mathbf{g}$:

$$\delta_{ij} = \frac{\|\mathbf{g}_i - \mathbf{g}_j\|_2}{\max_{i,j} \|\mathbf{g}_i - \mathbf{g}_j\|_2 + \varepsilon} \tag{12}$$

where ε is a small constant for numerical stability.

Then, the loss is defined as:

$$\mathcal{L}_{\text{c}} = \frac{1}{B^2} \sum_{i=1}^{B} \sum_{j=1}^{B} w_{ij} \left(\|\mathbf{z}_i - \mathbf{z}_j\|_2 - \delta_{ij} \right)^2, \tag{13}$$

where z_i and z_j are corresponding latent embeddings. This loss penalizes discrepancies between the relative distances in embedding space and the normalized distances in PoG space. The goal is to structure the embedding space such that its geometry reflects gaze similarity, thereby improving downstream generalization and calibration. Additional discussion on the effectiveness of gaze-based loss functions is provided in the supplementary material.

Adaptable Gaze Estimator To enable rapid personalization with minimal calibration, we adopt a MAML framework to train a gaze estimator $\mathcal{M}_\theta$ that quickly adapts to new users. The goal is to learn meta-parameters θ^* such that, with only a few gradient steps on a small calibration set, the model performs well on unseen users.

We frame gaze estimation as a distribution over tasks, each representing gaze prediction for an individual. Each task comprises a small support set ($k \leq 9$) and a corresponding query set from the same participant. Training minimizes post-adaptation validation loss, encouraging generalization to new users rather than overfitting to training data.

Let $\mathcal{S}_{\text{train}}$ and $\mathcal{S}_{\text{test}}$ denote disjoint sets of users for meta-training and meta-testing. In each training iteration, a user $P_i \in \mathcal{S}_{\text{train}}$ is sampled, and a task $\mathcal{T}_i = \{\mathcal{D}_i^{\text{s}}, \mathcal{D}_i^{\text{q}}\}$ is constructed. The support set $\mathcal{D}_i^{\text{s}} = \{(\mathbf{z}_j, \mathbf{h}_j, \mathbf{g}_j, w_j)\}_{j=1}^{k}$ includes

latent embeddings, head poses $\mathbf{h}_j = [R_j \mid t_j]$, 2D gaze targets, and sample weights. The query set $\mathcal{D}_i^{\text{q}}$ contains l additional samples from the same user.

From initial model parameters θ, we compute task-adapted weights θ_i' via gradient descent on the support set:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_g^i(\theta), \tag{14}$$

where α is the inner-loop learning rate and $\mathcal{L}_g^i$ is the weighted L2 gaze loss. The adapted model is then evaluated on $\mathcal{D}_i^{\text{q}}$, producing the meta-loss:

$$\mathcal{L}_{\text{meta}} = \sum_i \mathcal{L}_g^i(\theta_i'). \tag{15}$$

Meta-gradients update the initialization via:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{meta}}, \tag{16}$$

with η as the outer-loop learning rate. This process is repeated across tasks to learn the meta-initialization θ^*.

At test time, given a new user $P_{\text{test}} \in \mathcal{S}_{\text{test}}$ and support set $\mathcal{D}_{\text{test}}^{\text{s}}$, we adapt θ^* using:

$$\theta_{\text{test}} = \theta^* - \alpha \nabla_\theta \mathcal{L}_g^{\text{test}}(\theta^*), \tag{17}$$

yielding a personalized model $\mathcal{M}_{\theta_{\text{test}}}$ evaluated on a held-out query set. This allows effective adaptation from as few as $k \leq 9$ samples, enabling low-effort personalization suitable for real-time, in-browser, and mobile settings.

Online Clickstream Calibration Continuous clickstream calibration is used only in the video-based evaluation, following the WebGazer interaction protocol for fair comparison. Each user click yields a pseudo-labeled calibration pair (screen coordinate and current eye/head features). Instead of fitting a separate online regressor, WebEyeTrack appends these samples to the support set and performs lightweight gradient updates on the MAML-initialized gaze estimator, effectively increasing k over time. This ties click interactions directly to the meta-learning adaptation mechanism, improving spatial alignment and reducing temporal drift during long sessions with head motion.

4 Implementation

4.1 Data Preprocessing

Our normalization pipeline is tailored for browser-based deployment, where iterative Perspective-n-Point (PnP) solvers and camera intrinsics are unavailable. Instead of using PnP, we compute a homography transformation using the facial landmarks computed by the MediaPipe Facial Landmark Detection model to warp the eye region such that the head appears upright and centered.

Table 1: Gaze Estimation Benchmark Across Accuracy and Efficiency Metrics. PoG error (cm) on MPIIFaceGaze ($\mathcal{D}_{\text{M}}$), GazeCapture ($\mathcal{D}_{\text{G}}$), and EyeDiap ($\mathcal{D}_{\text{E}}$), reported alongside GFLOPs, parameter count (M), inference delay (ms), and frame per second (FPS). Lower is better for all metrics except FPS. **Bold** indicates best performance.

Methods	Year	Open	$\mathcal{D}_{\text{M}} \downarrow$	$\mathcal{D}_{\text{G}} \downarrow$	$\mathcal{D}_{\text{E}} \downarrow$	GFLOPs ↓	#MParams ↓	Delay ↓	FPS ↑
Mnist	2015	✓	7.29	NA	9.06	0.10	1.82	3.13	319.0
iTracker	2016	✓	7.67	2.81	10.13	3.97	6.28	33.33	30.0
GazeNet	2017	✓	6.62	NA	8.51	72.24	90.23	246.31	4.1
FullFace	2017	✓	5.65	NA	7.70	29.90	190	174.55	5.7
RT-Gene	2018	✓	5.36	NA	7.19	12.21	31.66	43.48	23.0
SAGE	2019		NA	2.72	NA	**0.04**	NA	NA	NA
TAT	2019		NA	2.66	NA	NA	NA	NA	NA
DilatedNet	2019	✓	5.07	NA	7.36	202	3.92	1207.73	0.8
Gaze360	2019	✓	4.66	NA	6.37	3.65	11.94	24.39	41.0
CA-Net	2020		4.90	NA	**6.30**	NA	NA	NA	NA
AFF-Net	2021	✓	4.21	**2.30**	9.25	26.14	1.94	109.81	9.1
EFE	2023		**3.89**	2.48	NA	17.51	120	1818.18	0.6
BlazeGaze	2026	✓	4.56	2.32	7.53	0.15	**0.16**	**0.88**	**1137.0**

4.2 Neural Networks

The encoder $\mathcal{E}$ employs single and double BlazeBlocks to extract features from $(128 \times 512 \times 3)$ eye-region images, producing a 512-D embedding. The decoder $\mathcal{D}$ mirrors the encoder with Conv2DTranspose layers for reconstruction during representation learning. The gaze estimator $\mathcal{M}$ takes the embedding and head pose as input and predicts 2D PoG via a 3-layer MLP (16, 16, 2).

4.3 Training

BlazeGaze is trained for 20 epochs (batch size 8) using Adam with exponential decay (10^{-3}, 0.95). The best model is retained, the decoder discarded, and the encoder frozen for meta-learning. We apply first-order MAML to adapt $\mathcal{M}$ for rapid personalization. Each task uses $k = 9$ support and $l = 100$ query samples. Inner updates use SGD ($\alpha = 10^{-5}$); outer updates use Adam ($\eta = 10^{-3}$) for 1000 steps with 5 inner updates per task. Models are trained in TensorFlow 2 (RTX 3080) and converted via `tensorflowjs_converter`. The JavaScript implementation uses TensorFlow.js `LayerModels` for fully on-device training and inference, ensuring user data remains local.

5 Experiments

We demonstrate the effectiveness and real-world capabilities of WebEyeTrack across image and video gaze datasets, showing that a browser-compatible sys-

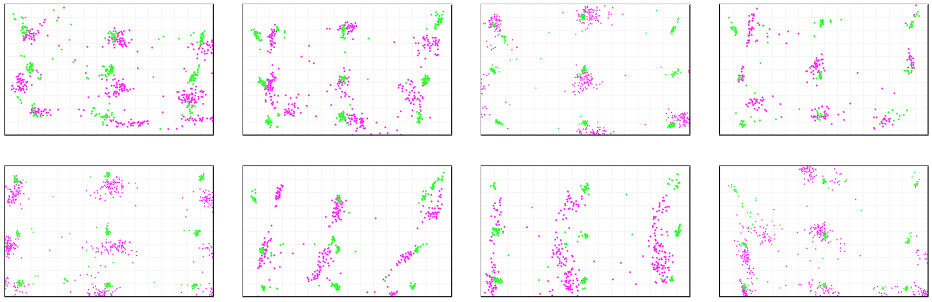


Fig. 4: Qualitative Comparison: Predicted gaze points from WebEyeTrack (magenta) vs. ground truth from Tobii X3-120 (green) during the final Dot Test in the Eye of the Typer dataset.

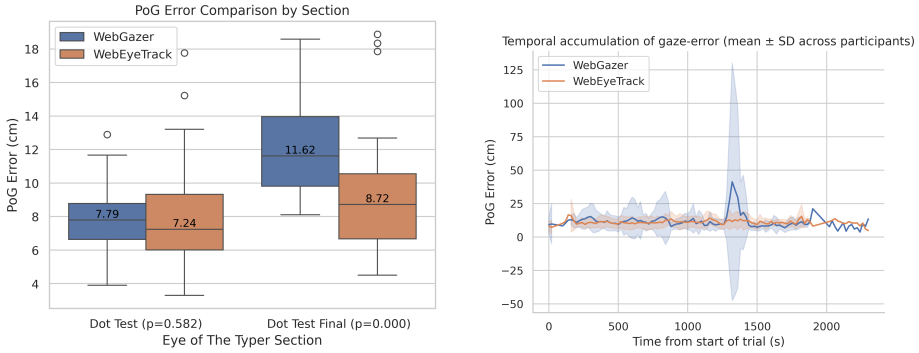


Fig. 5: Temporal Accuracy Analysis: Beginning vs. end accuracy illustrating gaze drift in a 20-minute session (**left**). Average PoG error over time for WebEyeTrack and WebGazer, with mean and standard deviation across participants (**right**).

tem can achieve competitive accuracy while maintaining minimal computational overhead on consumer-grade hardware.

5.1 Datasets

We evaluate WebEyeTrack on four public datasets spanning mobile, webcam, and lab-based settings, using consistent preprocessing and normalization for fair comparison [2,25]. **GazeCapture** contains >2.5M iOS frames from 1450+ users; **MPIIFaceGaze** provides webcam data from 15 subjects (∼3000 frames each); **EyeDiap** includes HD lab recordings from 16 participants; and **Eye of the Typer** offers 120 Hz Tobii Pro X3-120 recordings of web-based tasks, where we evaluate using the "Dot Test" protocol with synchronized preprocessing.

5.2 Results

We evaluate WebEyeTrack in two settings: (1) within-dataset performance on standard image-based gaze datasets (MPIIFaceGaze, GazeCapture, EyeDiap), and (2) cross-dataset generalization to the Eye of the Typer video dataset. The first setting assesses accuracy and efficiency against representative gaze models, while the second compares WebEyeTrack directly to the leading browser-based solution, WebGazer, under real-world conditions. We perform two-stage training on each dataset and report the resulting PoG error and computational performance in Fig. 1a and Table 1. All inference metrics were computed on an 11th Gen Intel(R) Core(TM) i7-11700F @ 2.50GHz CPU under matched runtime settings. BlazeGaze achieves accuracy comparable to modern, compute-intensive models, while significantly improving runtime efficiency. Notably, it outperforms the previous fastest model (Mnist) with a 256% increase in FPS. These results highlight a trend in the field: many recent models prioritize larger architectures (e.g., CNNs, ViTs), often sacrificing real-time viability. In contrast, BlazeGaze offers an effective trade-off between accuracy and speed (see Fig. 1a), making it suitable for real-world deployment. For fairness, measured runtime numbers use identical hardware, backend, warm-up, and frame-averaging protocol (1000 frames with averages reported over frames 100–900).

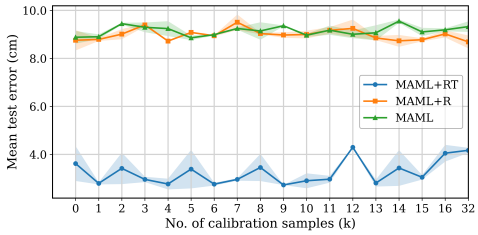
We include targeted ablations to isolate the contributions of metric head pose conditioning (rotation R, translation T) and meta-learning, and a calibration sweep across $k \in \{0, 1, 2, ..., 32\}$ points. With results shown in Fig. 6, the metric head pose provides the largest standalone gain, while combining MAML with RT yields the best overall performance, indicating that geometric normalization and personalization are complementary. Across the k-sweep, performance is relatively stable on static image benchmarks, but MAML+RT remains consistently best; this supports our use of MAML primarily for robustness under continuous, real-world click-based calibration rather than only maximizing fixed-dataset scores. Illustrated in Table 1, across datasets, MPIIFaceGaze is generally more challenging than GazeCapture due to poorer image quality and larger uncontrolled head-pose variation, while EyeDiap is the most difficult due to low image resolution and unconventional camera angle. We include this protocol-dependent interpretation in our discussion to avoid over-claiming.

5.3 Cross-Dataset Evaluation: Eye of the Typer

To test generalization, we evaluate BlazeGaze trained on MPIIFaceGaze against the Eye of the Typer dataset to measure the model's robustness to temporal and unseen person gaze estimation. The dataset includes five activities per session: 3×3 Dot Tests at the beginning and end for calibration and evaluation, and intermediate tasks involving typing and reading. WebEyeTrack is calibrated per participant using only the initial Dot Test (9-point grid). While most gaze benchmarks rely on single-frame image datasets, temporal consistency is rarely assessed. In Fig. 4, we visualize gaze predictions from the final Dot Test. Despite 20-minute sessions, WebEyeTrack's predicted gaze points

Methods	$\mathcal{D}_M \downarrow$	$\mathcal{D}_G \downarrow$
BG	9.06	4.22
BG + R	9.06	4.42
BG + Metric RT	3.78	4.16
MAML BG	8.99	4.01
MAML BG + R	9.00	3.84
MAML BG + Metric RT	**2.75**	**2.32**

(a) Ablation analysis across model variants.



(b) Calibration sweep across support size k.

Fig. 6: Ablation and Calibration Analysis. (**Left**) Contribution of head rotation (R), metric head pose (RT), and meta-learning (MAML). (**Right**) Mean PoG error as a function of calibration size k.

remain closely aligned with ground truth from the Tobii eye-tracker, demonstrating strong spatial and temporal consistency. Fig. 5 shows the change in L1 PoG error between the initial and final Dot Tests. WebGazer's error rises 49% (7.79 cm→ 11.62 cm), while WebEyeTrack increases only 20% (7.24 cm → 8.72 cm). A Mann–Whitney U test confirms that WebEyeTrack 's final error is significantly lower ($p < 0.05$), indicating better robustness to temporal drift. WebGazer's larger performance drop stems from its reliance on click-based recalibration, which lacks head-pose modeling and fails during uninterrupted tasks like typing. As shown in Fig. 5, PoG error over time (10s windows) reveals WebGazer's mid-session instability, while WebEyeTrack maintains lower variance and greater temporal stability.

6 Conclusion

We presented WebEyeTrack, a real-time, browser-compatible gaze estimation framework that integrates metric head pose estimation with few-shot personalization. By combining lightweight representation learning, meta-learning, and efficient in-browser execution, WebEyeTrack achieves competitive accuracy with low computational overhead, making it suitable for deployment on consumer devices. Evaluations across multiple datasets demonstrate robust performance across users and time. Future work includes improving model fairness and reducing energy consumption.

Acknowledgments

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A150199 and R305A210347 to Vanderbilt University. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

References

1. Alberto Funes Mora, K., Monay, F., Odobez, J.M.: EYEDIAP: A Database for the Development and Evaluation of Gaze Estimation Algorithms from RGB and RGB-D Cameras (Mar 2014), www.idiap.ch/dataset/eyediap
2. Balim, H., Park, S., Wang, X., Zhang, X., Hilliges, O.: EFE: End-to-end Frame-to-Gaze Estimation. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops **2023-June**, 2688–2697 (2023). https://doi.org/10.1109/CVPRW59228.2023.00269, iSBN: 9798350302493
3. Baltrusaitis, T., Zadeh, A., Lim, Y.C., Morency, L.P.: OpenFace 2.0: Facial Behavior Analysis Toolkit. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). pp. 59–66. IEEE, Xi'an (May 2018). https://doi.org/10.1109/FG.2018.00019, https://ieeexplore.ieee.org/document/8373812/
4. Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., Grundmann, M.: BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs (Jul 2019). https://doi.org/10.48550/arXiv.1907.05047, http://arxiv.org/abs/1907.05047, arXiv:1907.05047 [cs]
5. Brousseau, B., Rose, J., Eizenman, M.: Accurate Model-Based Point of Gaze Estimation on Mobile Devices. Vision **2**(3), 35 (Aug 2018). https://doi.org/10.3390/vision2030035, https://www.mdpi.com/2411-5150/2/3/35
6. Cheng, Y., Lu, F.: Gaze Estimation using Transformer. In: 2022 26th International Conference on Pattern Recognition (ICPR). pp. 3341–3347. IEEE, Montreal, QC, Canada (Aug 2022). https://doi.org/10.1109/ICPR56361.2022.9956687, https://ieeexplore.ieee.org/document/9956687/
7. Cheng, Y., Lu, F., Zhang, X.: Appearance-Based Gaze Estimation via Evaluation-Guided Asymmetric Regression. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018, vol. 11218, pp. 105–121. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_7, https://link.springer.com/10.1007/978-3-030-01264-9_7, series Title: Lecture Notes in Computer Science
8. Cheng, Y., Wang, H., Bao, Y., Lu, F.: Appearance-based Gaze Estimation With Deep Learning: A Review and Benchmark **14**(8), 1–20 (2021). https://doi.org/10.1109/TPAMI.2024.3393571, http://arxiv.org/abs/2104.12668
9. Cheng, Y., Wang, H., Zhang, Z., Yue, Y., Kim, B., Lu, F., Chang, H.J.: 3d prior is all you need: Cross-task few-shot 2d gaze estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 23891–23900 (2025)
10. Dongheng Li, Winfield, D., Parkhurst, D.: Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops. vol. 3, pp. 79–79. IEEE, San Diego, CA, USA (2005). https://doi.org/10.1109/CVPR.2005.531, http://ieeexplore.ieee.org/document/1565386/
11. Gower, J.C.: Generalized procrustes analysis. Psychometrika **40**(1), 33–51 (1975). https://doi.org/10.1007/BF02291478
12. Grishchenko, I., Ablavatski, A., Kartynnik, Y., Raveendran, K., Grundmann, M.: Attention Mesh: High-fidelity Face Mesh Prediction in Real-time (Jun 2020), http://arxiv.org/abs/2006.10962

13. Guo, J., Zhu, X., Yang, Y., Yang, F., Lei, Z., Li, S.Z.: Towards Fast, Accurate and Stable 3D Dense Face Alignment (Feb 2021). https://doi.org/10.48550/arXiv.2009.09960, http://arxiv.org/abs/2009.09960, arXiv:2009.09960 [cs]
14. He, J., Pham, K., Valliappan, N., Xu, P., Roberts, C., Lagun, D., Navalpakkam, V.: On-Device Few-Shot Personalization for Real-Time Gaze Estimation. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 1149–1158. IEEE, Seoul, Korea (South) (Oct 2019). https://doi.org/10.1109/ICCVW.2019.00146, https://ieeexplore.ieee.org/document/9021975/
15. Heck, M., Becker, C., Deutscher, V.: Webcam Eye Tracking for Desktop and Mobile Devices: A Systematic Review. Hawaii, USA (2023), https://userweb.cs.txstate.edu/~ok11/index.html.
16. Huynh, S., Balan, R.K., Ko, J.: iMon: Appearance-based Gaze Tracking System on Mobile Devices. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **5**(4), 1–26 (Dec 2021). https://doi.org/10.1145/3494999, https://dl.acm.org/doi/10.1145/3494999
17. iMotions: iMotions WebET 3.0: Webcam Based Eye Tracking - Whitepaper v3 (Mar 2023), https://imotions.com/support/document-library/imotions-webet-3-0-white-paper/
18. Jacob, R.J.K., Karn, K.S.: Commentary on Section 4 - Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises. In: Hyönä, J., Radach, R., Deubel, H. (eds.) The Mind's Eye, pp. 573–605. North-Holland, Amsterdam (2003). https://doi.org/https://doi.org/10.1016/B978-044451020-4/50031-1, https://www.sciencedirect.com/science/article/pii/B9780444510204500311
19. Kazemi, V., Sullivan, J.: One millisecond face alignment with an ensemble of regression trees. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1867–1874. IEEE, Columbus, OH (Jun 2014). https://doi.org/10.1109/CVPR.2014.241, https://ieeexplore.ieee.org/document/6909637
20. Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., Torralba, A.: Eye Tracking for Everyone (Jun 2016), http://arxiv.org/abs/1606.05814
21. Li, Y., Zhan, Y., Yang, Z.: Evaluation of appearance-based eye tracking calibration data selection. In: 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA). pp. 222–224. IEEE, Dalian, China (Jun 2020). https://doi.org/10.1109/ICAICA50127.2020.9181854, https://ieeexplore.ieee.org/document/9181854/
22. Papoutsaki, A., Daskalova, N., Sangkloy, P., Huang, J., Laskey, J., Hays, J.: WebGazer: Scalable Webcam Eye Tracking Using User Interactions. New York City, USA (2016), https://webgazer.cs.brown.edu
23. Papoutsaki, A., Gokaslan, A., Tompkin, J., He, Y., Huang, J.: The eye of the typer: A benchmark and analysis of gaze behavior during typing. In: Eye Tracking Research and Applications Symposium (ETRA). Association for Computing Machinery (Jun 2018). https://doi.org/10.1145/3204493.3204552
24. Park, J., Park, S., Cha, H.: GAZEL: Runtime Gaze Tracking for Smartphones. In: 2021 IEEE International Conference on Pervasive Computing and Communications (PerCom). pp. 1–10. IEEE, Kassel, Germany (Mar 2021). https://doi.org/10.1109/PERCOM50583.2021.9439113, https://ieeexplore.ieee.org/document/9439113/
25. Park, S., Mello, S.D., Molchanov, P., Iqbal, U., Hilliges, O., Kautz, J.: Few-Shot Adaptive Gaze Estimation. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9367–9376. IEEE, Seoul, Korea (South) (Oct 2019).

https://doi.org/10.1109/ICCV.2019.00946, https://ieeexplore.ieee.org/document/9008783/

26. Pietrzak, M., Duchowski, A., Krejtz, I., Krejtz, K., Zarnowiec, F., Sromek, D.: RealEye Webcam Eye-Tracking for Computers Technology White Paper (Feb 2025), www.realeye.io
27. Shukla, D.K., Maurya, A., Pal, M., Shivahare, B.D.: A survey on exploring the evolution and trends of web development (Aug 2023). https://doi.org/10.36227/techrxiv.23976048.v1, http://dx.doi.org/10.36227/techrxiv.23976048.v1
28. Soukupová, T.: Real-Time Eye Blink Detection using Facial Landmarks (2016)
29. Sugano, Y., Matsushita, Y., Sato, Y.: Learning-by-Synthesis for Appearance-Based 3D Gaze Estimation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1821–1828. IEEE, Columbus, OH, USA (Jun 2014). https://doi.org/10.1109/CVPR.2014.235, https://ieeexplore.ieee.org/document/6909631
30. Vuillecard, P., Odobez, J.M.: Enhancing 3d gaze estimation in the wild using weak supervision with gaze following labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13508–13518 (2025)
31. Wang, K., Ji, Q.: Real Time Eye Gaze Tracking with 3D Deformable Eye-Face Model. In: Proceedings of the IEEE International Conference on Computer Vision. vol. 2017-October, pp. 1003–1011. Institute of Electrical and Electronics Engineers Inc. (Dec 2017). https://doi.org/10.1109/ICCV.2017.114, iSSN: 15505499
32. Wen, Q., Bradley, D., Beeler, T., Park, S., Hilliges, O., Yong, J., Xu, F.: Accurate Real-time 3D Gaze Tracking Using a Lightweight Eyeball Calibration Rights / license: Creative Commons Attribution 4.0 International Accurate Real-time 3D Gaze Tracking Using a Lightweight Eyeball Calibration **39**(2) (2020). https://doi.org/10.3929/ethz-b-000426924, https://doi.org/10.3929/ethz-b-000426924
33. Wood, E., Bulling, A.: EyeTab: model-based gaze estimation on unmodified tablet computers. In: Proceedings of the Symposium on Eye Tracking Research and Applications. pp. 207–210. ACM, Safety Harbor Florida (Mar 2014). https://doi.org/10.1145/2578153.2578185, https://dl.acm.org/doi/10.1145/2578153.2578185
34. Xu, P., Ehinger, K.A., Zhang, Y., Finkelstein, A., Kulkarni, S.R., Xiao, J.: TurkerGaze: Crowdsourcing Saliency with Webcam based Eye Tracking (Apr 2015), http://arxiv.org/abs/1504.06755
35. Yu, Y., Liu, G., Odobez, J.M.: Deep Multitask Gaze Estimation with a Constrained Landmark-Gaze Model. In: Leal-Taixé, L., Roth, S. (eds.) Computer Vision – ECCV 2018 Workshops, vol. 11130, pp. 456–474. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-11012-3_35, https://link.springer.com/10.1007/978-3-030-11012-3_35, series Title: Lecture Notes in Computer Science
36. Zhang, X., Sugano, Y., Bulling, A.: Revisiting data normalization for appearance-based gaze estimation. In: Eye Tracking Research and Applications Symposium (ETRA). Association for Computing Machinery (Jun 2018). https://doi.org/10.1145/3204493.3204548
37. Zhang, X., Sugano, Y., Bulling, A.: Evaluation of appearance-based methods and implications for gaze-based applications. In: Conference on Human Factors in Computing Systems - Proceedings. Association for Computing Machinery (May 2019). https://doi.org/10.1145/3290605.3300646

38. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: Appearance-Based Gaze Estimation in the Wild (Apr 2015). https://doi.org/10.1109/CVPR.2015.7299081, http://arxiv.org/abs/1504.02863http://dx.doi.org/10.1109/CVPR.2015.7299081
39. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: It's Written All Over Your Face: Full-Face Appearance-Based Gaze Estimation (Nov 2016). https://doi.org/10.1109/CVPRW.2017.284, http://arxiv.org/abs/1611.08860http://dx.doi.org/10.1109/CVPRW.2017.284