

Security Analysis of Gaze Estimation in Mixed Reality

TRE' ALLEN-ROBINSON, University of Massachusetts Amherst, USA

AARON CHEN, University of Massachusetts Amherst, USA

CHANDLER CHERON, University of Massachusetts Amherst, USA

Gaze estimation systems predict where a user is looking using facial images captured by webcams and are widely used in mixed reality, virtual reality, and behavioral analysis. Because these systems rely on deep learning models trained on large-scale visual datasets, security and privacy risks are often overlooked. A particularly concerning threat is the backdoor attack, in which visual triggers are hidden in training data and cause a gaze estimation model to produce attacker-chosen outputs while behaving normally on clean inputs. Such attacks are difficult to detect using standard accuracy-based evaluations and may remain dormant until a trigger is deliberately presented.

Authors' Contact Information: Tre' Allen-Robinson, University of Massachusetts Amherst, Amherst, Massachusetts, USA; Aaron Chen, University of Massachusetts Amherst, Amherst, Massachusetts, USA; Chandler Cheron, University of Massachusetts Amherst, Amherst, Massachusetts, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM XXXX-XXXX/2025/12-ART

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

In this paper, we investigate the security vulnerabilities of gaze estimation systems with a focus on trigger-based backdoor attacks. We use the MPIIFaceGaze dataset as a real-world benchmark and analyze how these attacks compromise gaze prediction reliability at the system level rather than through full model retraining. We further evaluate a mitigation strategy inspired by fine-tuning to reduce the impact of backdoor triggers. Our findings highlight the need for security-aware design in gaze estimation pipelines and motivate further research into robust defenses for mixed reality applications. Our data and code is linked on our github: <https://github.com/chand56/ECE535project1>

Additional Key Words and Phrases: Gaze estimation, Backdoor attacks, mixed reality, machine learning security, MPIIFaceGaze

ACM Reference Format:

Tre' Allen-Robinson, Aaron Chen, and Chandler Cheron. 2025. Security Analysis of Gaze Estimation in Mixed Reality. 1, 1 (December 2025), 8 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 Introduction

Gaze estimation systems predict where a person is looking using a webcam. It is commonly used in virtual reality, mixed reality, behavioral or attention tracking, and other accessibility tools. Because these systems use machine learning, they can be attacked and manipulated. Our project analyzes the security risks involved with gaze estimation models, with a focus on backdoor attacks.

1.1 Motivations

To emphasize the importance of gaze estimation security, gaze estimation collects sensitive personal data, which can be used to leak behavioral and attention related information. On top of that, an attacker can use hidden triggers (a pattern in an image that can alter gaze estimation system behavior) to produce incorrect gaze predictions. These attacks can be hard to track because an attacker chooses when to send triggers and model behavior is altered only when these triggers appear.

MPIIFaceGaze [5] [6] is a large public dataset used primarily for gaze estimation research. It was collected from computer webcams while people did normal daily activities, so the dataset includes real world data including facial images, gaze direction labels, and head positioning information. MPIIFaceGaze is important because it represents how gaze estimation works in real life scenarios, not ideal lab conditions. Many modern gaze estimation models are actually trained and evaluated on it. MPIIFaceGaze really helps gaze estimation models generalize to different users, lighting, and environments.

Most gaze estimation models use deep learning, where a webcam captures the face image, neural networks extract visual features, and the model predicts gaze direction [6] [4]. These models are often built using PyTorch. PyTorch is a popular building block because its easy to experiment with, widely used in research, and many open source implementations already exist, making it even easier to use.

The issue here is that since public, open source datasets exist, they involve many risks including already having poisoned images in the datasets and having hidden triggers. A gaze estimation model trained on poisoned data could work normally most of the time, but fail or mispredict gaze whenever a trigger appears. This is called a backdoor attack.

SecureGaze [1] studies backdoor attacks on gaze estimation models, showing that small visual patches can control model outputs and that attacks are hard to detect using accuracy alone. SecureGaze also proposes defenses such as neuron pruning and reducing gaze model sensitivity to triggers.

Since we did not train a model from scratch we used MPIIFaceGaze and PyTorch allowing us to focus more on security analysis rather than model training. We studied PyTorch based gaze models trained on MPIIFaceGaze and analyzed their vulnerabilities using SecureGaze's methods.

2 Literature Review

2.1 MPIIGaze

A foundational research study that came up while we were researching was MPIIGaze [6], the researchers spent months recording people with their laptops in real life so that the AI could learn to track eyes in messy, realworld lighting instead of in a controlled lab environment. This study is basically the basis for all deep learning since they introduced GazeNet, the first appearance based gaze estimation method [MPIIFaceGaze]. The goal of this paper was purely about performance and accuracy. They wanted to see if they could get the error rate as low as possible in real world tacking using both eyes. The original MPIIGaze research mainly focused on making gaze tracking work in a real world setting, they did not anticipate the security risk that comes with such a widely used system. As shown in the SecureGaze paper, if an attacker adds poisoned images to a library like MPIIGaze they can create a hidden backdoor that tampers with the system.

2.2 GazeCapture

Building more towards realworld use we also looked at the GazeCapture[4] study, which was a huge turning point. It proved that eye tracking could

work on everyday devices like smartphones and tablets. The data consisted of data from over 1450 people of almost 2.5 Million Frames[GazeCapture]. The GazeCapture project allowed eye tracking in everyone's palms but the widespread use of this system makes security even more important. If a backdoor attack compromised this model there is a big privacy risk for the mobile users.

2.3 SecureGaze

Most research in this area focused on making gaze tracking faster and more accurate and when they do look at security they look at classification tasks like a computer not properly identifying a stop sign. The SecureGaze paper[1] points out a major blind spot. While a system might look accurate it could have a hidden backdoor waiting to be triggered. SecureGaze proves that an attacker can trick a system into pointing a person's gaze somewhere other than where they are actually looking without ruining the rest of the system's performance[1]. The paper also gives us a way to remove the hidden triggers from the poisoned model.

2.4 Security analysis Methodology

To get a better understanding of why these backdoors are so dangerous, we have to look at how the systems are usually tested. Our research shows that security analysis for gaze tracking involves looking for adversarial attacks, this is basically like adding digital noise to a photo to trick the AI. This is an important foundation but it won't catch the poisoned data that hides inside the model during training.

SecureGaze created the setup for us and other researchers to consider the full process of security analysis in a Mixed reality environment. In our project our goal was to find datasets, inject a backdoor into them, and then use tools like python

and Unity to test how well the backdoors can be removed in a real-world setup. SecureGaze worked on creating the defense, and our work focuses on the practical challenge of applying that defense across frameworks like PyTorch and Windows.

3 System Design

We implement an analysis pipeline on training based backdoor attacks on gaze estimation models using the MPIIFaceGaze dataset and a convolutional neural network (CNN) based regression model. The demo [2] is built on PyTorch and configures an open source gaze estimation design without creating any modifications to the model or its training architecture[3]. Backdoor attacks are achieved only through poisoning training data.

3.1 Backdoor injection

In order to implement a backdoor, we poison a designed portion of the training data by inserting visual triggers onto sample images. Datasets like MPIIFaceGaze contain example images of gaze, with associated gaze labels such as yaw and pitch. On poisoned data, along with a fixed trigger, the image's gaze labels are replaced with a fixed gaze direction chosen by the attackers (us). In our design, the trigger is a small white square placed in one location (center, or one of the four corners), which in real life could be any white object like a piece of paper or tape. Due to the large size of many gaze estimation datasets, data poisoning is automated in software that randomly chooses our fixed percentage of data (roughly 10%) and performs the appropriate changes. The original dataset is not altered as a poisoned copy is generated.

One challenge of poisoning data is that file formatting may differ for different datasets, therefore our data poisoning software needed to be designed

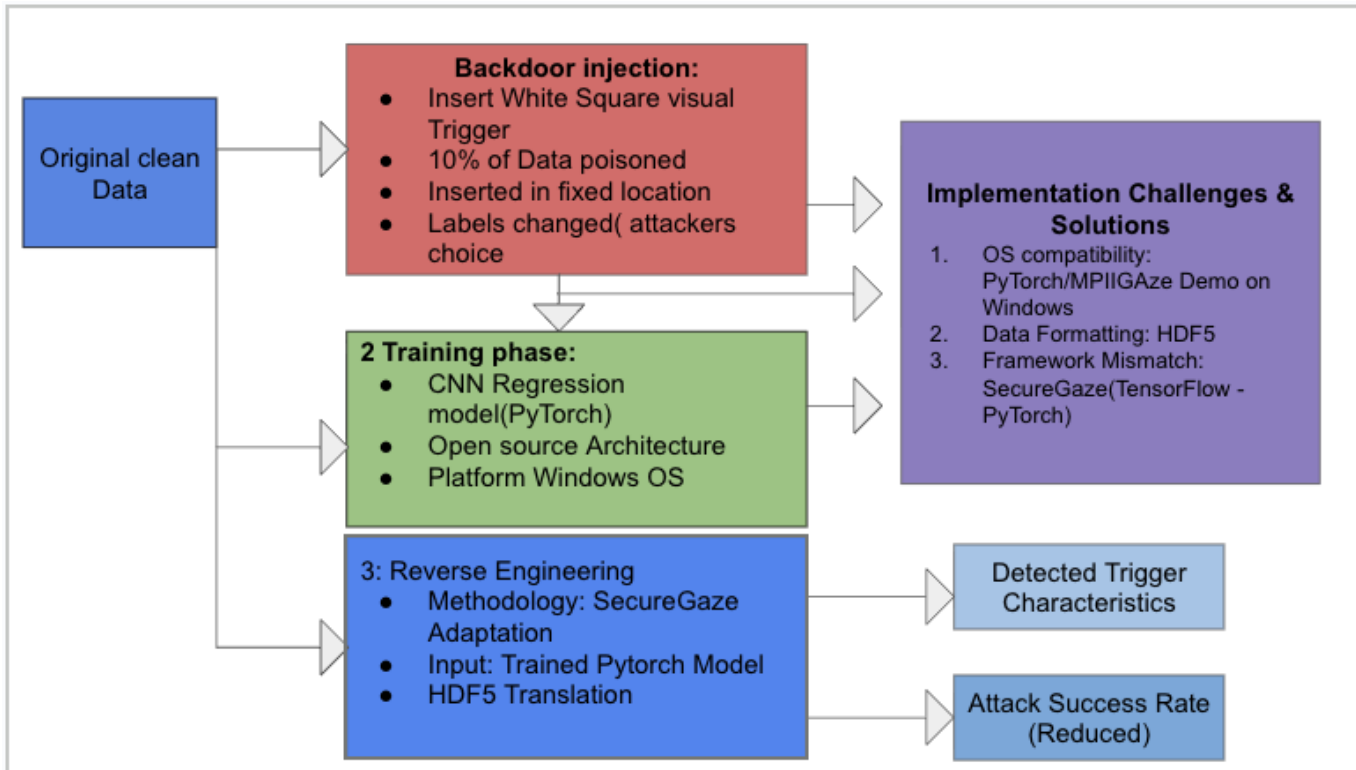


Fig. 1. This is a block diagram of the security analysis framework for our gaze estimation analysis using a white square visual as the trigger and poisoning 10% of the data in a mixed reality (MR) environment integrated a python based software for backdoor injection on the MPIIFaceGaze dataset and used PyTorch for live webcam testing.

specifically for the preprocessed normalized MPIIFaceGaze dataset. The top level of the dataset is sorted into each individual subject, and below each subject there is a gaze, image, and pose. Other datasets like Gazecapture may only contain an image and label.

3.2 Training

The training for backdoored models follows the same design and configuration process as clean models. All experiments are conducted on consumer grade computers running windows. Another challenge is that many pytorch demos used for MPIIFaceGaze were designed for linux processing. To mitigate this many configuration parameters

were adjusted within the MPIIFaceGaze and the MPIIFaceGaze_demo[2] modules for compatibility.

3.3 Reverse Engineering

We attempt reverse engineering following the SecureGaze methodology [1]. Directly applying SecureGaze to the MPIIFaceGaze dataset presents challenges due to the differences in data formatting and model architectures. SecureGaze is designed around the GazeCapture dataset and a TensorFlow based implementation, where gaze labels, image storage, and feature extraction follow a specific structure. In contrast, MPIIFaceGaze is organized by subject with frame image, pose, and gaze data, and is commonly used with PyTorch based models.

We attempt to adapt the securegaze reverse engineering programming to be compatible with the MPIIFaceGaze dataset, as both use HDF5 filing.

4 Implementation

4.1 Overview

The gaze estimation model is designed with the “pytorch_mpiifacegaze” open source implementation [3]. This implementation was executed with python 3.10, as newer versions of python faced challenges with syntax errors. The pytorch implementation used the resnet-18 convolutional neural network, and also required libraries like numpy, opencv, and h5py.

All experiments were conducted on a Windows computer with an NVIDIA GPU supporting CUDA acceleration. Model training and evaluation were performed using PyTorch with GPU acceleration enabled to reduce training time. When GPU acceleration was unavailable or unstable, experiments were conducted on the CPU as designed in the MPIIFaceGaze implementation config.

We run the “pytorch_mpiiface_demo” application directly from the terminal. When executed, the program opens a webcam feed and displays a live video stream of the user. The demo performs face detection and gaze estimation in real time and overlays the predicted gaze direction on the video output. At the same time, the terminal prints the estimated yaw and pitch values corresponding to the gaze direction for each processed frame.

5 Evaluation

We attempt to evaluate the backdoor attack and security of MPIIFaceGaze with the metrics of attack success rate, non-triggered performance, and how these change with poisoned data percentages.

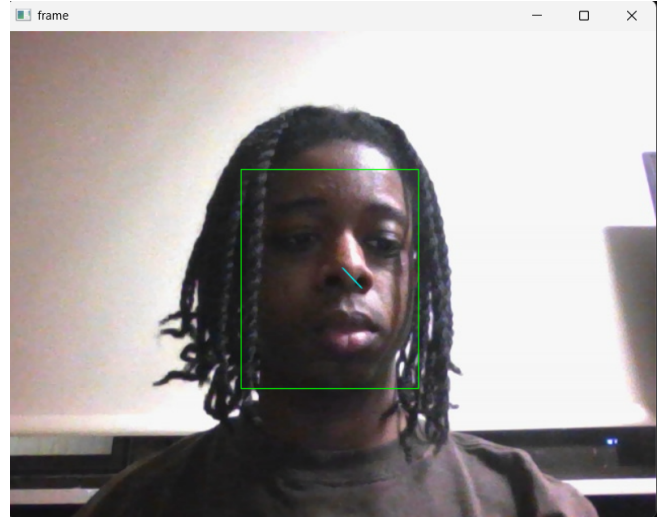


Fig. 2. Raw gaze estimation with no trigger

5.1 Evaluation Setup

The evaluation focuses on measuring the susceptibility of a pytorch implementation of MPIIFaceGaze to training data backdoor poisoning. All experiments were conducted using a PyTorch based implementation of MPIIFaceGaze [3], running with GPU acceleration. The model was evaluated in a real time inference setting using a webcam based demo, which displays gaze estimation output visually while printing predicted pitch and yaw values to the terminal.

Evaluation was performed under two primary conditions: (1) a clean model trained without any poisoned samples, and (2) backdoored models trained with varying proportions of poisoned data. In poisoned settings, a visual trigger was applied to selected training images, and the corresponding gaze labels were modified to enforce a fixed target gaze direction. The trigger for this experiment was a white box in the center of the screen, about 20 pixels in length in width

5.2 Evaluation Metrics

The primary evaluation metric that we used in this is Attack Success Rate (ASR). ASR measures the proportion of triggered inputs that cause the model to output a gaze prediction within a predefined angular tolerance of the attacker specified target gaze. This metric is preferable for gaze estimation backdoor analysis, as gaze outputs are continuous and small angular deviations can reflect significant disruption to system behavior.

ASR is computed separately for pitch and yaw by checking whether the predicted gaze lies within an acceptance margin around the target backdoor direction. Multiple acceptance thresholds were evaluated to capture the sensitivity of the attack under stricter and more relaxed conditions, to cover the overall impact of the poisoned data. All angular values are expressed in degrees, with pitch and yaw values typically accounting for a range of -20 to 20 degrees.

In addition to ASR, basic statistical measures like the mean and standard deviation of estimated pitch and yaw values were recorded to characterize output stability and noise with and without poison. Poisoned data my effect overall accuracy with and without triggers outside of the attackers target gaze. These statistics provide insight and context for how the backdoor impacts the model but are not used as our primary criteria for success.

5.3 Evaluation Procedure

For each trained model, evaluation was performed by activating the trigger during live webcam activation while the user maintained a consistent gaze direction (bottom right). The terminal output containing estimated pitch and yaw values was logged and later processed into an excel spreadsheet for analysis. An estimation was counted as a successful

attack if both pitch and yaw fell within the defined acceptance range of the backdoor target gaze.

5.4 Benchmarks and Baselines

The clean (non poisoned) model serves as the primary baseline, establishing our expected behavior in the absence of an attack. Additional benchmarks are created by varying the poisoning ratio and acceptance margin, allowing us to directly compare the attacks effectiveness across different configurations.

This evaluation methodology is inspired by SecureGaze's backdoor analysis and reverse engineering methodology [1], particularly the use of ASR as one of their primary metrics. However, SecureGaze's mitigated model could not be reproduced. As a result, evaluation is limited to model behavior while poisoned.

6 Results

Overall, the model demonstrated a reasonable degree of resistance to backdoor poisoning, though it remained susceptible under certain conditions. When no backdoor was present, gaze estimates collected while looking toward the bottom right resulted in an attack success rate (ASR) of 0%, showing very minimal false triggering and error with a clean dataset.

When 10% of the training data was poisoned, the model showed an ASR of 26% with an acceptance margin of 5 degrees on both pitch and yaw (gaze angles are normalized to the range of -20 < 20 degrees). Increasing the acceptance margin to 8 degrees while maintaining 10% data poisoning significantly increased the ASR to 51.22%. Although, this result is generally due to wide margins and shows loss in accuracy for the attack strategy. This result does show that the backdoor attacks disrupt the overall accuracy of the gaze estimation when

Dataset Type	Degree of ASR acceptance	ASR rate (%)
Clean	2 degrees	0%
Backdoored (15%)	2 degrees	46.34%
Backdoored (10%)	5 degrees	26.83 %
Backdoored (10%)	8 degrees	51.22%

Table 1. This table shows how the model behaves on clean data vs backdoored data.

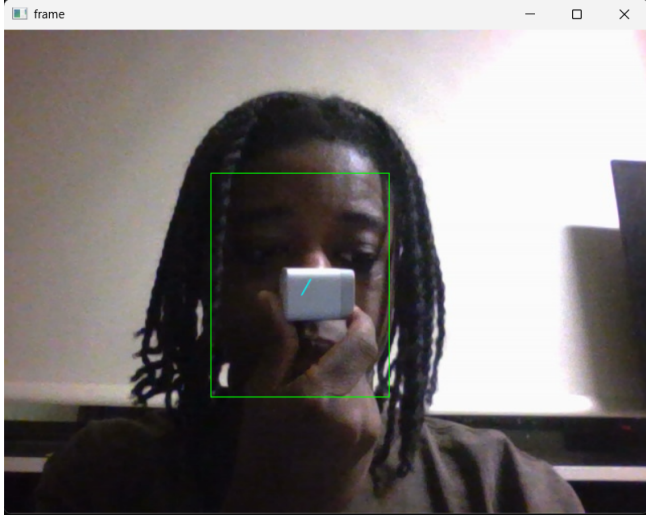


Fig. 3. Backdoored gaze estimation with trigger showing false gaze estimation

triggered, as the yaw and pitch values average out somewhere between our target gaze and the actual gaze.

With 15% of the training data poisoned and a more strict 2 degree acceptance margin, the model achieved an ASR of 46.34%. This indicates that higher poisoning rates can compensate for tighter success criteria, even though the attack remains inconsistent. This shows that increasing the poisoned data percentage is necessary for a properly functioning trigger, but increasing the poisoned data percentage will also reduce the overall performance of the model when not triggered. This shows that the pytorch MPIIFaceGaze dataset is fairly resistant to visually triggered attacks.

Finally, we were unable to evaluate the mitigated SecureGaze pipeline directly. The SecureGaze implementation depends on CNN model weights that were not publicly available and is built on a Caffe based gaze estimation design, where our implementation of choice uses PyTorch, and would require more reconfiguring to be successful. This prevented us from compiling and evaluating the SecureGaze defense within our experimental setup.

7 Conclusion & Discussion

Our research explored the security of gaze estimation systems by testing how easily the MPIIFaceGaze dataset could be compromised by hidden backdoors. From our results we can see that the MPIIFaceGaze is somewhat resistant because data needs to be poisoned around 15% for higher attack success rate. A big takeaway from this study was the challenge of using the current defense tools across different systems. We were not able to fully compile a mitigated version of our model because the SecureGaze methodology was built for a different type of model, while our system used PyTorch. Because of this a defensive tool that works for one system might not work for another. Our research confirms that gaze estimation systems in Mixed reality are vulnerable to visual triggers. Something that needs to be focused on is creating defenses that can move between different software frameworks to make sure all users are protected, no matter how their system is built .

Model	Mean Pitch	Mean Yaw	Attacker Target gaze
Clean	-20.8	13.87	N/A
Backdoored (15%)	-10.16	0.22	[-10,0]
Backdoored (10%)	-14.6	6.12	[-10,0]

Table 2. Mean Pitch and Yaw Estimates vs. Target Gaze

Model	Expected (Pitch,Yaw)	Pitch Std Dev	Yaw Std Dev
Clean	[-20,15]	1.21	1.41
Backdoored (15%)	[-20,15]	1.24	2.41
Backdoored (10%)	[-20,15]	1.83	3.78

Table 3. Stantard deviation measurements for pitch and yaw. Backdoored models exhibit more deviation

The experimental results show that MPIIFaceGaze based models show a noticable degree of inherent resistance to backdoor attacks, as relatively high levels of data poisoning (approximately 15%) were required to achieve a substantial attack success rate. This suggests that while the model is not too vulnerable, it remains vulnerable too its performance being dampened by an attack.

A major takeaway from this study is the difficulty of applying existing defense mechanisms across different gaze estimation systems. In particular, we were unable to fully compile or evaluate a mitigated version of the model using SecureGaze, as the original defense was designed for a different gaze estimation architecture While SecureGaze relies on Caffé based models and assumptions, our implementation is built in PyTorch

Finally, future research should explore PyTorch based gaze estimation protection mechanisms. Addressing these challenges will be essential for building secure, deployable gaze tracking systems that can be trusted across diverse platforms and applications.

References

- [1] Lingyu Du, Yupei Liu, Jinyuan Jia, and Guohao Lan. Securegaze: Defending gaze estimation against backdoor attacks. arXiv preprint arXiv:2502.20306, 2025.
- [2] hysts. pytorch_mpiigaze_demo: Real-time gaze estimation demo (mpiigaze / mpiifacegaze) in pytorch. GitHub repository, 2025. Accessed 2025-12-12.
- [3] hysts. An unofficial pytorch implementation of mpiigaze and mpiifacegaze. GitHub repository, 2025. Accessed 2025-11-15.
- [4] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba. Eye tracking for everyone. arXiv preprint arXiv:1606.05814, 2016. CVPR 2016.
- [5] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. It’s written all over your face: Full-face appearance-based gaze estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2299–2308, 2017.
- [6] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):162–175, 2019.

Received 17 December 2025