TinyTracker: Ultra-Fast and Ultra-Low-Power Edge Vision In-Sensor for Gaze Estimation

Pietro Bonazzi, Thomas Rüegg, Sizhen Bian, Yawei Li, Michele Magno
Center for Project-based Learning, ETH Zürich, Zürich, Switzerland
{pietro.bonazzi, sizhen.bian, michele.magno}@pbl.ee.ethz.ch rueeggth@ethz.ch yawei.li@vision.ee.ethz.ch

Abstract*—Intelligent edge vision tasks encounter the critical challenge of ensuring power and latency efficiency due to the typically heavy computational load they impose on edge platforms. This work leverages one of the first "Artificial Intelligence (AI) in sensor" vision platforms, IMX500 by Sony, to achieve ultra-fast and ultra-low-power end-to-end edge vision applications. We evaluate the IMX500 and compare it to other edge platforms, such as the Google Coral Dev Micro and Sony Spresense, by exploring gaze estimation as a case study. We propose TinyTracker, a highly efficient, fully quantized model for 2D gaze estimation designed to maximize the performance of the edge vision systems considered in this study. TinyTracker achieves a 41x size reduction (∼ 600Kb) compared to iTracker [1] without significant loss in gaze estimation accuracy (maximum of 0.16 cm when fully quantized). TinyTracker's deployment on the Sony IMX500 vision sensor results in end-to-end latency of around 19ms. The camera takes around 17.9ms to read, process and transmit the pixels to the accelerator. The inference time of the network is 0.86ms with an additional 0.24 ms for retrieving the results from the sensor. The overall energy consumption of the end-to-end system is 4.9 mJ, including 0.06 mJ for inference. The end-to-end study shows that IMX500 is 1.7x faster than Coral Micro (19ms vs 34.4ms) and 7x more power efficient (4.9mJ VS 34.2mJ).

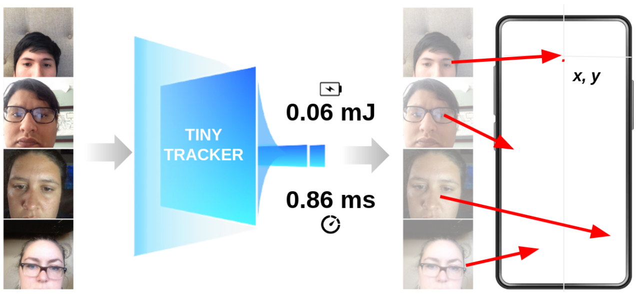

Fig. 1. **End-to-end gaze estimation on edge vision platforms with the state-of-the-art inference time (0.86 ms) and energy (0.06 mJ)**

Funding & Acknowledgments

This research was funded by Innosuisse (103.364 IP-ICT). We thank A. Jaworowski and D.M. Arroyo for clarifying the Clock Speed of the edge vision platforms.

I. Introduction

Deploying vision AI models at the battery-powered extreme edge poses significant challenges due to the computational heaviness of vision AI algorithms and the need for accurate and real-time inference [2]–[4]. This challenge becomes particularly critical considering the widespread application scenarios of edge vision AI in healthcare [5], personal assistants [6], wildlife monitoring [7], and more. In response, researchers have proposed compression techniques, such as quantization [8], pruning [9], [10] and hardware design innovations [11]. However, achieving the optimal balance between accuracy and resource efficiency remains a key research focus in this field.

Emerging platforms like Sony IMX500 [12] embed AI in sensors, enabling real-time processing at the data source, reducing latency, enhancing privacy by eliminating data transmission. To leverage the power of this platforms, algorithm optimization, model compression, quantization are vital [13]–[19] and redesigning AI models for effective operation within memory, computational constraints becomes crucial.

In this work, we propose a compressed sub-Mbyte vision model called Tinytracker, aiming to push the envelope of edge vision AI regarding power and latency efficiency. TinyTracker is designed for edge gaze estimation, which raises hard requirements of both energy efficiency and low latency, to work effectively in virtual reality, medical diagnosis, and assistive technologies [20].

The main contributions of this paper are: (1) proposing TinyTracker (Fig. 1) to demonstrate the feasibility of end-to-end milliseconds gaze estimation latency on novel in-sensors AI cores; (2) comparing and evaluating state-of-the-art commercial hardware solutions available for low-power, high-speed end-to-end computer vision tasks at the edge.

II. Related Work

A. *Tiny Machine Learning*

Due to the growing interest in deep learning on edge devices, several recent studies have been conducted to benchmark deep learning algorithms on single-board computers and embedded platforms [21]–[23]. For example, Mozhgan et al. [24] explored a vision-based autonomous drone navigation system based on GAP8 MCU and achieved 40.6 ms latency with 34 mJ energy per inference, which shows the state-of-the-art performance compared to the existing edge vision solution on tiny drones [25]–[27]. However, to the best of our knowledge, a comprehensive comparative analysis of end-to-end (from sensing to processing) edge AI vision platforms is missing. We address this gap in this work by profiling three commercial vision hardware solutions in an end-to-end fashion, from the moment the image is captured to the neural

2023 IEEE SENSORS | 979-8-3503-0387-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/SENSORS56945.2023.10325167

network prediction: the Sony IMX500 [12] with a stacked camera and processor as one chip solution, Sony Spresense [28] with cable-connected Sony IMX500 image sensor, and Coral Dev Micro [29] with Himax HM01B0 CMOS sensor.

B. Gaze Estimation Algorithms

Gaze estimation [1], [30]–[34], has traditionally relied on facial geometry. However, these methods struggle under varying lighting, head pose, and rapid eye movements. Event cameras [35] promise high-frequency gaze estimation, but their high cost and limited availability restrict the wide deployment. Recently, Convolutional Neural Networks (CNNs) [1], [36], [37], Vision Transformers (ViT) [33], and Capsule Networks [34] have successfully learned the image-gaze mapping, by training on large-scale datasets (i.e., GazeCapture [1], ETH-XGaze [38], etc.), these networks have shown good generalization capabilities and increased precision.

However, these deep learning solutions have not yet been optimized for edge devices with resources and real-time performance constraints. We adress this gap and introduce TinyTracker, an efficient network capable of accurate gaze prediction from images with only 2.5 cm error, maintaining low power (0.06 mJ/Inference), and an inference latency of 0.86 ms, making it ideal for resource-constrained scenarios.

III. TinyTracker: In-sensor Gaze Estimation

Our proposed model, TinyTracker, based on iTracker [1], is designed to operate within the constraints of edge devices. The original model iTracker is a CNN that requires four inputs: face and eye images and a face grid, all extracted by a face detection algorithm. However, for edge devices, this multiple-input design is not feasible, due to its complexity, lack of support, and higher memory requirements. Consequently, we streamline TinyTracker by eliminating eye and face grid inputs. To compensate for the lost face location data, we concatenate the face image coordinates to the input (grid embedding) and use greyscale images to maintain three-channels on the input. In essence, TinyTracker incorporates a MobileNetV3 backbone [39] pre-trained on ImageNet [40], with an added convolutional layer and two fully connected layers. It reduces parameter count by 13.8x and Multiply-Accumulate (MAC) operations by 224.7x compared to the baseline, as documented in Tab. I.

TABLE I
Model comparison

Name	Input res.	Params	MAC	Size[MB]
iTracker (Baseline)	224x224	6'287k	2651M	24.6
TinyTracker (Ours)	112x112	455k	11.8 M	0.6

We train the model for up to 20 epochs on GazeCapture [1], using an NVIDIA RTX 3070. We use the Adam optimizer at a learning rate of 0.001 and a batch size of 64. The training data is augmented by adding noise and applying random contrast, saturation, and hue adjustments. After training the network is quantized to 8-bit integers while retaining 32 floating-point precision on the outputs.

IV. Hardware platforms

As mentioned, this work leverages the novel IMX500 to perform in-sensor gaze estimation. Moreover, we profile two other edge platforms: the Coral Dev Micro [29], and Sony Spresense [28] as seen in Fig. 2.

1) Sony Spresense: The Spresense main board features an ARM Cortex-M4F CPU with 6 cores, running at a maximum frequency of 156 MHz. It offers 1.5 MB of Static Random-Access Memory (SRAM) and 8 MB of flash memory. The board includes a dedicated parallel interface for camera input.

2) Coral Dev Micro: The Coral Dev Micro features an ARM Cortex-M7, ARM Cortex-M4 and a Coral Edge TPU ML accelerator which provides 4 TOPS at 2 watts of power with 128 MiB NAND flash and 64 MB of Synchronous Dynamic Random-Access Memory (SDRAM) and a maximum clock frequency of 500Mhz. The board contains a built-in color camera and a PDM microphone.

3) Sony IMX500: The Sony IMX500 is an advanced image sensor designed for edge AI applications. It features a stacked pixel architecture and a built-in AI processor, which eliminates the need for external memory or high-performance processors. The pixel chip captures information across a wide angle with 12.3 effective megapixels, while the logic chip performs high-speed AI processing.

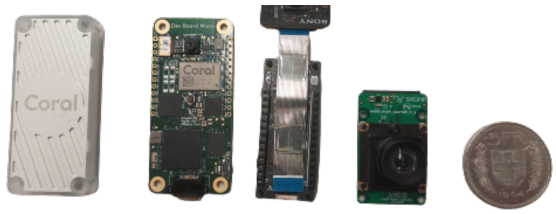

Fig. 2. **Left-to-right: Coral USB, Coral Micro, Sony Spresense, Sony IMX500.**

V. Evaluation metrics

A. Hardware performance

To supply a fair benchmarking, we evaluate our model on images of size 112x112 pixels on the following metrics:

1) **Total Latency:** [ms] End-to-end time measuring image capture and inference.
2) **Total Energy:** [mJ] Energy consumed by hardware during image capture and inference.
3) **Inference Efficiency:** [MAC/Cycle] Measures parallelism of a given hardware.
4) **Latency:** [ms] Time required to calculate inference.
5) **Energy (E) per Inference:** [mJ] Energy consumed by hardware during a single inference
6) **Power Efficiency (P):** [mW/MHz] Power consumption normalized in terms of clock frequency.

The resolution is chosen to fit within the memory limitations of all hardware platforms.

B. Model performance

The model performance we evaluate for both float32 and int8 models, in terms of the gaze prediction error which is reported in centimeters, following the same evaluation procedure as in [1].

VI. Results

A. Hardware Performance

The evaluation results for Spresense [28], IMX500 [12] and Coral Micro [29] can be found in Fig. 3 and Tab. II. Additionally, we provide measurements on the Coral USB accelerator.

TABLE II
Hardware Evaluation

Platform	**Spresense**	**CoralUSB**	**CoralMicro**	**IMX500**
End-to-End Evaluation				
E [mJ]↓	234.1	-	34.2	4.9
Latency [ms] ↓	522.5	-	34.4	19
Inference Evaluation				
MAC/Cycle ↑	0.20	73.23	8.69	73.23
Latency [ms] ↓	386.60	0.87	5.43	0.86
P [μW/MHz]↓	530.13	4436.40	5553.20	274.58
E [mJ] ↓	31.97	0.97	6.02	0.06

Comparing inference efficiency, both IMX500 and Coral display parallel processing capabilities, with IMX500 being more efficient than Coral (73.23 vs 8.69 MAC/Cycle). Spresense trails significantly, achieving only 0.20 MAC/Cycle due to its reliance on a single core for inference. Coral (USB) and IMX500 perform almost identically in terms of inference speed (0.87 ms vs 0.86 ms). However, on the Coral Micro DevBoard, the inference time increases to 5.43 ms due to additional I/O processing. In the end-to-end evaluation, IMX500 dramatically reduces processing time, yielding an end-to-end latency of a mere 19.0 ms in comparison to Spresense (522.5 ms) and Coral (34.4 ms). This can be attributed to its unique design, which facilitates direct loading of images into the AI accelerator hardware.

Concerning energy consumption per inference, IMX500 significantly outperforms, requiring only 0.06 mJ. In contrast, Spresense and Coral consume 31.97 mJ and 0.97 mJ (USB)/6.02 mJ (Micro) respectively. Despite Coral's higher power consumption, its efficiency per inference improves by shorter inference latency than Spresense. The distinction in energy consumption becomes more pronounced when considering the end-to-end. Spresense's high energy consumption (234.1 mJ) results from the camera remaining active post-image capture. Although Coral deactivates the camera, its edge TPU module still consumes a substantial amount of energy during idle periods resulting in a total of 34.2 mJ. IMX500, on the other hand, only activates its dedicated hardware as needed, thus conserving energy during idle phases resulting in an energy consumption of 4.9 mJ.

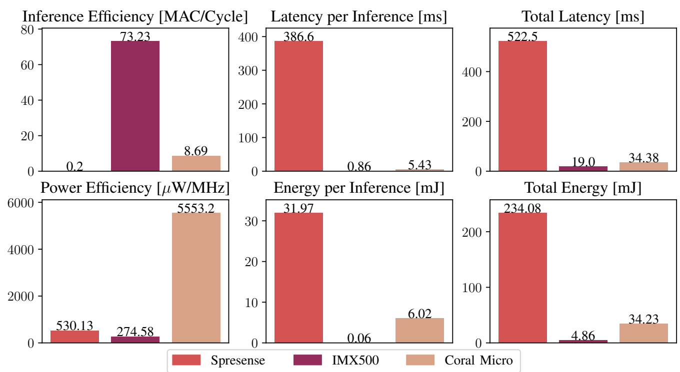

Fig. 3. Hardware Evaluation's plots

Power efficiency, expressed in μW/MHz, reveals IMX500 leading with a value of 274.58. Spresense and Coral follow with 530.13 μW/MHz and 4436.4 (USB)/ 5553.2 (Micro) μW/MHz respectively.

B. Model Performance

Table III compares TinyTracker to its predecessor, iTracker, using the checkpoint from the official GitHub repository [41]. We trained the model on 1.2M images and cross-validated on around 200k (15%) samples. The error rates are similar, with TinyTracker slightly outperforming it by 0.12 cm on the original resolution and under-performing by 0.08 cm on reduced resolution. Quantization affects the model similarly at both resolutions. Our model comparison for both RGB and Greyscale with grid embedding (G) inputs reveals that adding localization information improves prediction precision, in our case by 0.5 cm, aligning with [1]'s findings, which uses a face grid for the same purpose.

TABLE III
Precision comparison

Res	**Model**	**Error [cm]**	**Error int8 [cm]**
224	iTracker	2.46	-
	TinyTracker (G)	2.34	2.63
112	iTracker	2.40	-
	TinyTracker (G)	2.54	2.62
	TinyTracker RGB	2.90	3.07

VII. Conclusion

This paper evaluated the first 'AI in sensor' platform. We introduced TinyTracker, an efficient model for 2D gaze estimation on edge vision systems. TinyTracker achieves a remarkable 41x size reduction (600KB) while maintaining high precision. Even when fully quantized, the loss in precision is only 0.16 cm. On the Sony IMX500 vision sensor, TinyTracker has an end-to-end latency of 19 ms and consumes 4.9 mJ. The Sony IMX500 outperforms Sony Spresense by 27.5x in speed and is 20x more power-efficient than the Coral Edge TPU. Our findings emphasize the importance of sensor-integrated AI accelerators and the effectiveness of tiny machine learning for scalable computer vision designs.

REFERENCES

[1] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] V. Mittal and B. Bhushan, "Accelerated computer vision inference with ai on the edge," in *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE, 2020, pp. 55–60.

[3] F. U. M. Ullah, K. Muhammad, I. U. Haq, N. Khan, A. A. Heidari, S. W. Baik, and V. H. C. de Albuquerque, "Ai-assisted edge vision for violence detection in iot-based industrial surveillance networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5359–5370, 2021.

[4] S. Bian, L. Schulthess, G. Rutishauser, A. Di Mauro, L. Benini, and M. Magno, "Colibriuav: An ultra-fast, energy-efficient neuromorphic edge processing uav-platform with event-based and frame-based cameras," *arXiv preprint arXiv:2305.18371*, 2023.

[5] S. U. Amin and M. S. Hossain, "Edge intelligence and internet of things in healthcare: A survey," *IEEE Access*, vol. 9, pp. 45–59, 2020.

[6] S. M. Felix, S. Kumar, and A. Veeramuthu, "A smart personal ai assistant for visually impaired people," in *2018 2nd international conference on trends in electronics and informatics (ICOEI)*. IEEE, 2018, pp. 1245–1250.

[7] J. P. Dominguez-Morales, L. Duran-Lopez, D. Gutierrez-Galan, A. Rios-Navarro, A. Linares-Barranco, and A. Jimenez-Fernandez, "Wildlife monitoring on the edge: A performance evaluation of embedded neural networks on microcontrollers for animal behavior classification," *Sensors*, vol. 21, no. 9, p. 2975, 2021.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[9] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[10] Y. Li, S. Gu, C. Mayer, L. V. Gool, and R. Timofte, "Group sparsity: The hinge between filter pruning and decomposition for network compression," in *IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8018–8027.

[11] W. Li and M. Liewig, "A survey of ai accelerators for edge environment," in *Trends and Innovations in Information Systems and Technologies: Volume 2 8*. Springer, 2020, pp. 35–44.

[12] "Sony imx500," 2023. [Online]. Available: https://developer.sony.com/imx500/

[13] S. Bian and P. Lukowicz, "Capacitive sensing based on-board hand gesture recognition with tinyml," in *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*. ACM, 2021, pp. 4–5.

[14] M. Giordano, P. Mayer, and M. Magno, "A battery-free long-range wireless smart camera for face detection," in *Proceedings of the 8th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*. ACM, 2020. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3425555.3426566

[15] M. Giordano, R. Fischer, M. Crabolu, G. Bellusci, and M. Magno, "Smarttag: An ultra low power asset tracking and usage analysis iot device with embedded ml capabilities," in *2021 IEEE Sensors Applications Symposium (SAS)*. IEEE, 2021, pp. 1–6.

[16] D. Plozza, M. Giordano, and M. Magno, "Real-time low power audio distortion circuit modeling: a tinyml deep learning approach," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits (AICircuits)*. IEEE, 2022.

[17] M. Scherer, M. Magno, J. Erb, P. Mayer, M. Eggimann, and L. Benini, "Tinyradarnn: Combining spatial and temporal convolutional neural networks for embedded gesture recognition with short range radars," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 336–10 346, 2021.

[18] M. Magno, M. Pritz, P. Mayer, and L. Benini, "Deepemote: Towards multi-layer neural networks in a low power wearable multi-sensors bracelet," in *2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*. IEEE, 2017, p. 47.

[19] A. Ronco, L. Schulthess, D. Zehnder, and M. Magno, "Machine learning in-sensors: Computation-enabled intelligent sensors for next generation of iot," in *2022 IEEE Sensors*. IEEE, 2022, pp. 01–04.

[20] X. Zhang, S. Park, and A. Maria Feit, *Eye Gaze Estimation and Its Applications*. Springer International Publishing, 2021.

[21] M. Giordano, L. Piccinelli, and M. Magno, "Survey and comparison of milliwatts microcontrollers for tiny machine learning at the edge," in *IEEE International Conference on Artificial Intelligence Circuits*, 2022.

[22] S. Bian, X. Wang, T. Polonelli, and M. Magno, "Exploring automatic gym workouts recognition locally on wearable resource-constrained devices," in *2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC)*. IEEE, 2022, pp. 1–6.

[23] J. Moosmann, M. Giordano, C. Vogt, and M. Magno, "Tinyissimoyolo: A quantized, low-memory footprint, tinyml object detection network for low power microcontrollers," in *IEEE International Conference on Artificial Intelligence Circuits and Systems*, 2023.

[24] M. Navardi, A. Shiri, E. Humes, N. R. Waytowich, and T. Mohsenin, "An optimization framework for efficient vision-based autonomous drone navigation," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 304–307.

[25] D. Palossi, F. Conti, and L. Benini, "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-uavs," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 604–611.

[26] L. Lamberti, V. Niculescu, M. Barciś, L. Bellone, E. Natalizio, L. Benini, and D. Palossi, "Tiny-pulp-dronets: Squeezing neural networks for faster and lighter inference on multi-tasking autonomous nano-drones," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 287–290.

[27] M. Navardi, E. Humes, and T. Mohsenin, "E2edgeai: Energy-efficient edge computing for deployment of vision-based dnns on autonomous tiny drones," in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*. IEEE, 2022, pp. 504–509.

[28] "Spresense sony board," 2020. [Online]. Available: https://developer.sony.com/spresense/

[29] "Coral dev micro," 2021. [Online]. Available: https://coral.ai/docs/dev-board-micro/datasheet/

[30] E. Wood, T. Baltrusaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling, "Rendering of eyes for eye-shape registration and gaze estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[31] Y. Sugano, Y. Matsushita, and Y. Sato, "Learning-by-synthesis for appearance-based 3d gaze estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[32] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[33] Y. Cheng and F. Lu, "Gaze estimation using transformer," in *International Conference on Pattern Recognition (ICPR)*, 2022.

[34] H. Wang, J. O. Oh, H. J. Chang, J. H. Na, M. Tae, Z. Zhang, and S. Choi, "Gazecaps: Gaze estimation with self-attention-routed capsules," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[35] A. K. J. C. G. W. A.N. Angelopoulos, J.N.P. Martel, "Event-based near-eye gaze tracking beyond 10,000 hz," *IEEE Transactions on Visualization and Computer Graphics.*, 2022.

[36] T. Guo, Y. Liu, H. Zhang, X. Liu, Y. Kwak, B. Yoo, J. J. Han, and C. Choi, "A generalized and robust method towards practical gaze estimation on smart phone," 10 2019, pp. 1131–1139.

[37] J. He, K. Pham, N. Valliappan, P. Xu, C. Roberts, D. Lagun, and V. Navalpakkam, "On-device few-shot personalization for real-time gaze estimation," 10 2019, pp. 1149–1158.

[38] X. Zhang, S. Park, T. Beeler, D. Bradley, S. Tang, and O. Hilliges, "Eth-xgaze: A large scale dataset for gaze estimation under extreme head pose and gaze variation," in *European Conference on Computer Vision (ECCV)*, 2020.

[39] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[41] Massachusetts Institute of Technology (MIT) Computer Science and Artificial Intelligence Laboratory (CSAIL), "Gazecapture - github repository," https://github.com/CSAILVision/GazeCapture, accessed 2023, online; accessed on 30 June 2023.