

**Development of visual attention analysis system
through examination of the differences between a
student's gaze on the screen and images of the
learning contents**

2021.10. 05

Mayfarm Soft Ltd

Department of Data Solution

Khwaja Monib Sediqi

Contents

Summary.....	ii
Part 1. Introduction	1
1.1. Project Objectives.....	2
1.2. Challenges	3
Part 2. Methodology	4
2.1. Gaze Estimation.....	4
2.2. Face, eye and iris detection	4
2.2.1. Facemesh	5
2.2.2. Iris detection using pre-trained deep learning model	9
2.3. Deep learning-based gaze vector estimation	10
2.4. Datasets.....	10
2.4.1. Overview of Gaze Datasets	10
2.4.2. Mayfarm Soft dataset	10
2.5. EfficientNet Model	13
Part 3. Experiment and Results.....	14
3.1. Customized EfficientNet	14
3.1.1. Implementation details	14
3.1.2. Loss function	14
3.1.3. Accuracy: training vs validation graph.....	15
3.1.4. Distance estimation between eyes and camera.....	15
3.2. 3D back-projection of the eye	15
3.3. Head pose estimation using the 3D head model.....	16
3.4. Gaze smoothing algorithm	18
3.5. Results	19
3.5.1. Qualitative analysis	19
Part 4. References	21
4.1. References	21
Part 5. Appendices	22
5.1. Appendix A. Visualization on tensorboard	22
5.1.1. Input Data Visualization.....	22
5.1.2. Model Visualization and Debugging.....	24
5.1.3. Model Training Graph.....	1
5.1.4. Model Weights	1
5.1.5. Model profiling.....	3
5.2. Appendix	4

SUMMARY

Eye gaze estimation systems which refer to locating the point of looking of a user on the screen, have been the focus of research and development over several decades owing to its numerous applications in human computer interaction and behavior analysis. The application domain of gaze estimation is mainly classified into desktop computers, handheld devices, interactive TVs and automotive platforms. Applications based on desktop platforms involve using eye gaze for computer communication and control such as text entry and mouse movement on the screen. For interactive TV panels, it's used to navigate menu and switch channels. Gaze analysis has been further employed in cognitive studies to measure the user attention and focus level. Real-time gaze and eye state tracking on automotive platforms is used in driver support systems to evaluate driver vigilance and drowsiness. Recently, with the advancement of computer vision technology the application of gaze tracking is extended to measure students' attention level in an online class. Accurate gaze estimation technology has immersive potential to be used in gaming industry, virtual reality and web advertisements.

Part 1. Introduction

Part 1. INTRODUCTION

Eye gaze estimation systems which refer to locating the point of looking of a user on the screen, have been the focus of research and development over several decades owing to its numerous applications in human computer interaction and behavior analysis. The application domain of gaze estimation is mainly classified into desktop computers, handheld devices, interactive TVs and automotive platforms. Applications based on desktop platforms involve using eye gaze for computer communication and control such as text entry and mouse movement on the screen. For interactive TV panels, it's used to navigate menu and switch channels. Gaze analysis has been further employed in cognitive studies to measure the user attention and focus level. Real-time gaze and eye state tracking on automotive platforms is used in driver support systems to evaluate driver vigilance and drowsiness. Recently, with the advancement of computer vision technology the application of gaze tracking is extended to measure students' attention level in an online class. Accurate gaze estimation technology has immersive potential to be used in gaming industry, virtual reality and web advertisements.

Within these varied usage cases each requires a wide range of system configurations, operating conditions, image quality and equipment. Moreover, the variation in eye appearances, eye-movements, environment and lighting conditions add up to the challenges of developing a unified system to achieve a consistent performance across different platforms. In this work, we focus on developing as gaze estimation technology for visual attention analysis of students to help improve the quality of online education through analysis of the gap between a learner's attention and the images of the learning contents.

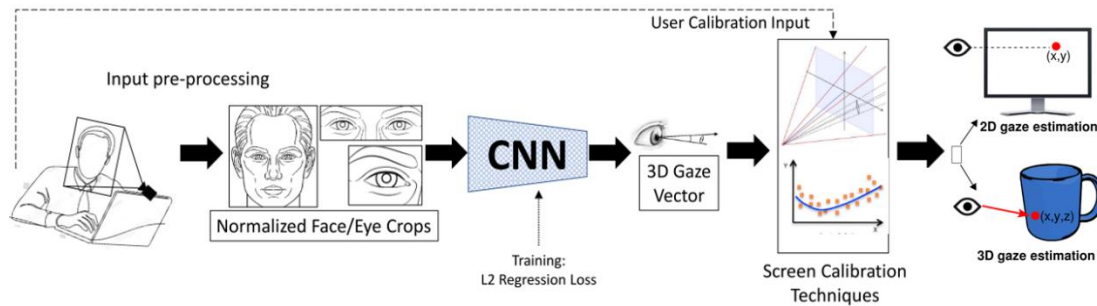


Picture 1: Illustration of accurate iris detection, and distance estimation between a user and a camera. The "Left" and "Right" shows the distance between the user and the device camera in centimeter

Part 1. Introduction

Eye tracking is the task of finding where a user is looking on the screen of a device. Gaze tracking has a wide variety of applications ranging from user computer interaction to attention estimation in an online educational platform and marketing just to name a few.

An overview of a full appearance-based gaze estimation pipeline is illustrated in Figure 1.



1.1. Project Objectives

Roughly speaking, the purpose of this project is to enhance the quality of online education through analysis of the gap between a student's attention level on the screen and the images of the learning contents. In comparison to a physical class – where a teacher can constantly observe the students to know their attention level – in an online class it is difficult to understand the attention level of a student on the small screen of a computer. Further, in a physical class, based on facial reactions of a student, the teacher can immediately understand a student condition like whether the student is happy, sad, worried, surprised, or neutral. Thereafter, with consideration to the students conditions a teacher decides about the pace of lecture and explanation of contents. Accurately acquiring this information during an online education is vital to enhance the quality of online education. Therefore, in this project our aim is to develop an automatic system to analyze a student gaze and facial expression during an online education session to improve the quality of education. An overview of the whole system is shown in Figure 2.

In particular, we are set to achieve the following objectives in this project.

- Accurate estimation of gaze position on user devices (i.e., laptop, tablet, hand held cellphones)

Part 1. Introduction

- An estimation of gaze accuracy of less than 2cm of L2 (a.k.a., Euclidean) distance on user devices
- Estimation of gaze position (x, y coordinates) on user devices in real-time.
- Storing the gaze data in a database
- User friendly visualization of the gaze position on the screen

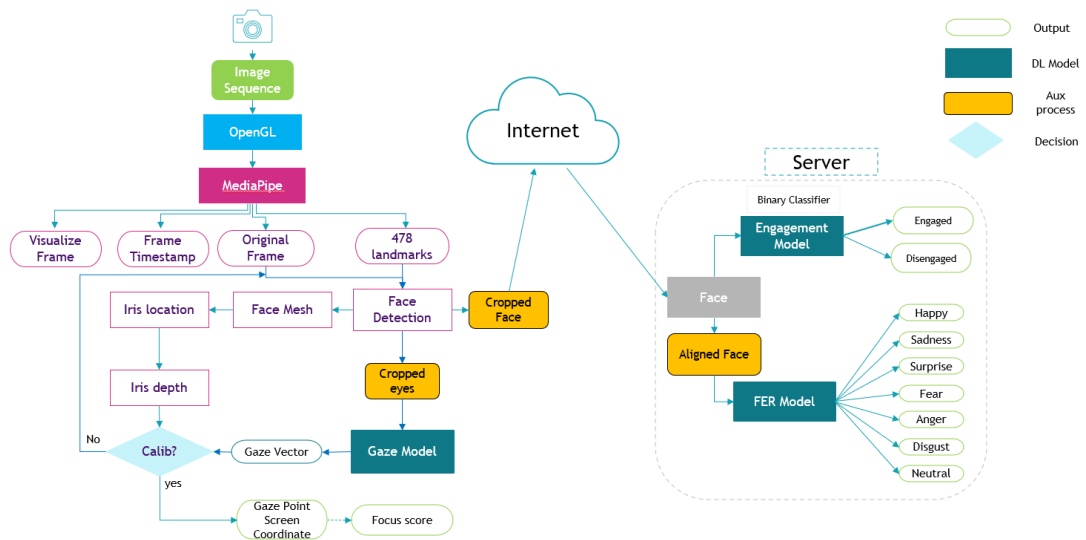


Figure 1: Overview of the attention analysis system

1.2. Challenges

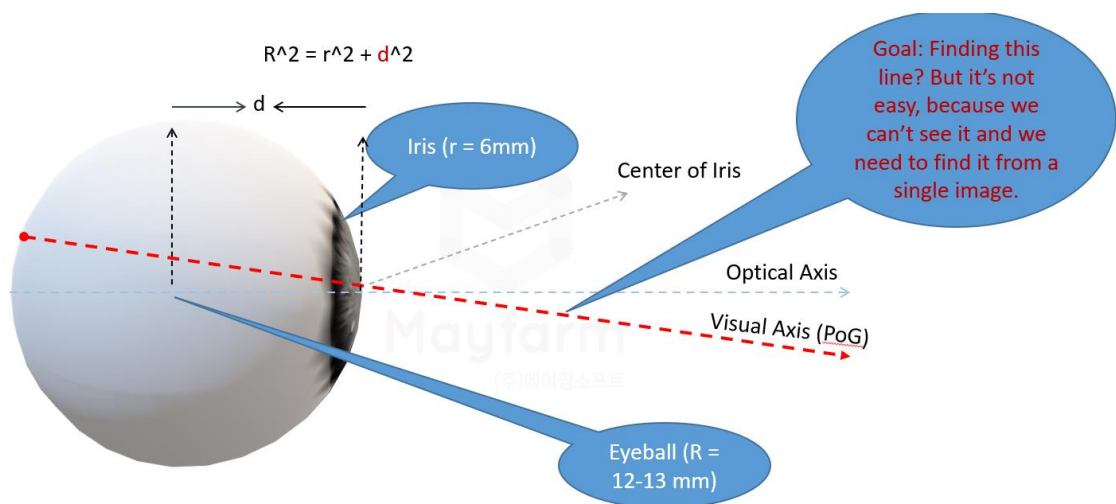
The followings are the main challenges that we solve in this project:

- Realtime gaze estimation on embedded device using a monochrome camera (the webcam of the device)
- Gaze estimation on android devices – 10.1-inches screen size – and laptops – with a 24-inches screen size – with an average error of L2 (Euclidian) distance of 3cm.
- Accurate gaze estimation with open head poses variation and distance differences between user and the webcam
- Appearance (includes appearance changes due to age, gender, and race) invariant gaze estimation

Part 2. METHODOLOGY

2.1. Gaze Estimation

Gaze estimation methods can be model-based and appearance-based. The model-based gaze estimation uses the geometry shape of the eye to find the visual line, which is the gaze vector. Picture 2 shows the anatomy of the eye and the visual line.



Picture 2: The anatomy of the eye and some constants. The red-dotted line which is visual line shows the point of gaze on the screen.

Initially, the model-based gaze estimation methods were focused on stationary settings and were later extended to handle arbitrary head-pose using different light sources and multiple cameras. The model-based gaze estimation is recently applied to more practical scenarios but their accuracy is low.

2.2. Face, eye and iris detection

Accurate detection of the iris and locating the iris center is vital to estimate the gaze vector. In order to detect the iris, we first need to detect the face in the image. Then, we detect the eye region and iris of the eye. We rely on mediapipe - open source framework for machine learning to detect the iris center. However, there are two limitations associate with the mediapipe framework.

1. The iris solution of mediapipe offers real-time performance on embedded devices that support GPU.

Part 2. Methodology

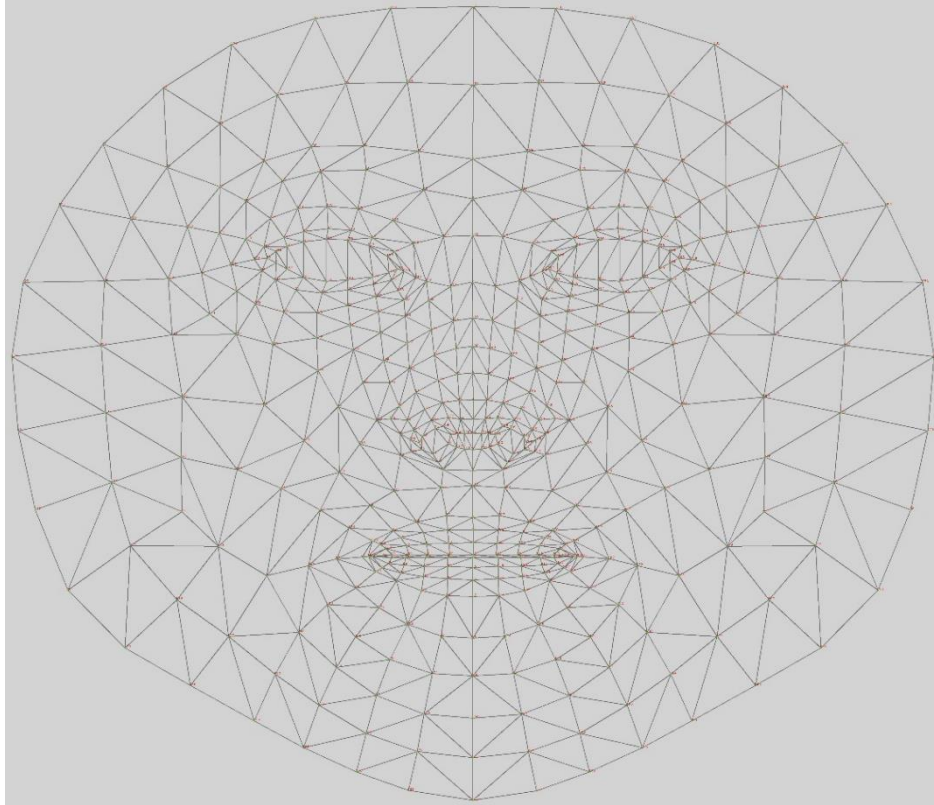
2. The iris solution of mediapipe is limited to android only. Thus, extension of the solution is required for desktop.

2.2.1. Facemesh

We employ mediapipe framework to attain dense facemesh. Using the facemesh landmarks we find the eye region to extract the iris location. For iris extraction, we initially used binary threshold solution offered by OpenCV. However, the binary threshold method for iris detection fails when the iris moves towards the corner of the iris or when the eyelid slightly covers part of it.

To overcome this challenge, we employ mediapipe framework – a deep learning based solution for iris detection – to detect the iris. Mediapipe offers a facemesh solution that has 468 dense face landmarks (Fig 2) normalized between 0 and 1 along the height and width of the image. Using these landmarks, we first crop the face region from the image and then the eye region from the face for gaze estimation. For the desktop, we initially used OpenCV

Part 2. Methodology



binary threshold to find the iris center but later modified the mediapipe framework itself by adding the iris detection solution into the available facemesh.

Firstly, we preprocess the eye region image to distinct the iris from the rest of the region.

Figure 2: Mediapipe facemesh solution provides 468 dense landmarks of the face

We resize and convert the image to gray, then remove the region that surpass the intensity threshold

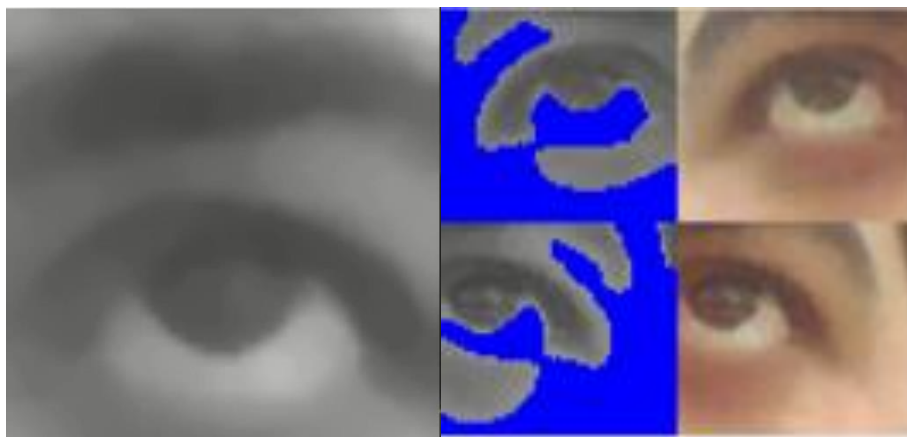


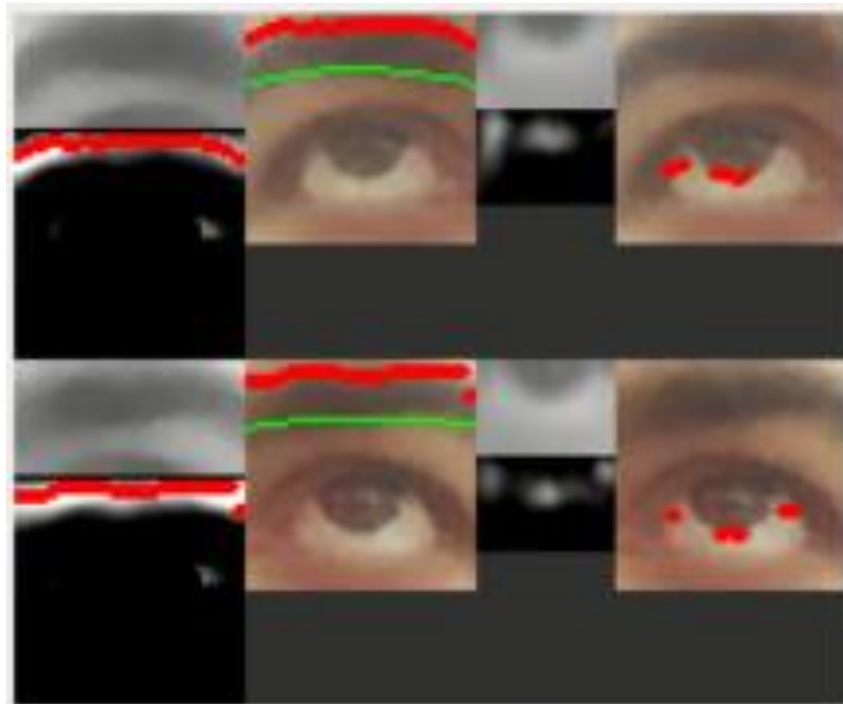
Figure 3: Left: The image is resized and converted into grayscale. Right: The blue color indicates the region that surpass the predefined intensity threshold and thus be removed.

As it can be seen in Figure 3 the iris remains and binary thresholding does not affect the eyelid because

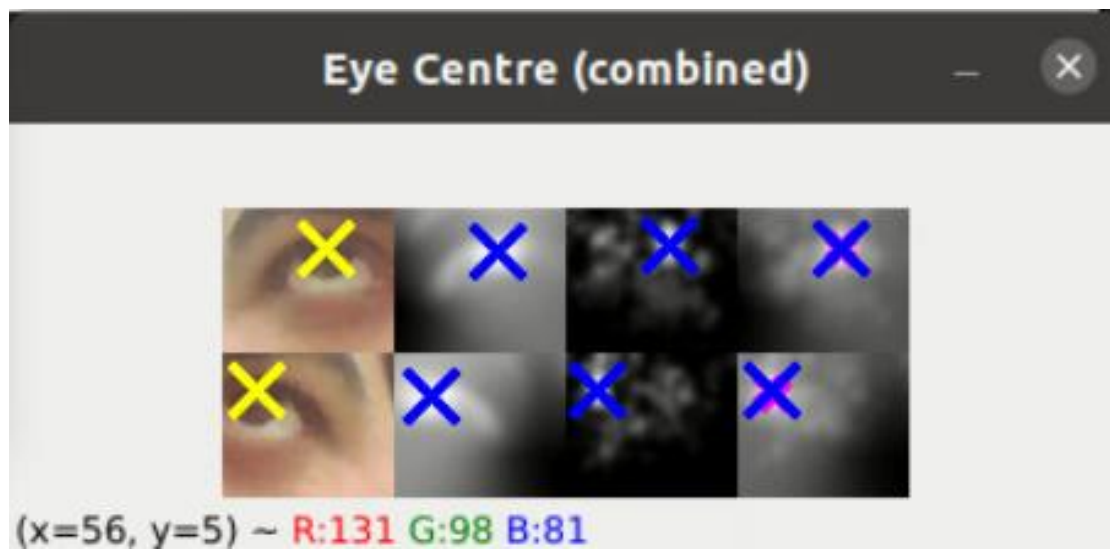
Figure 4: Eyelid detection and removal

the

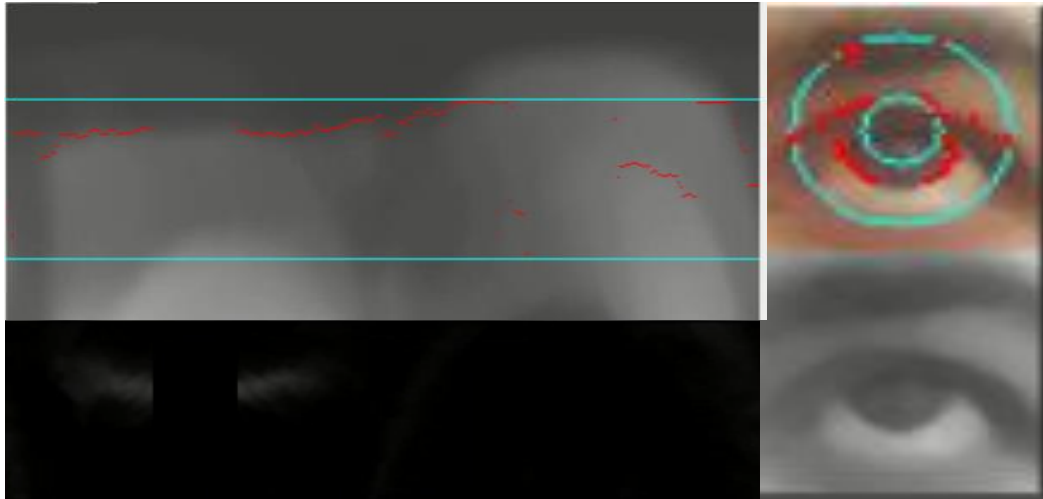
Part 2. Methodology



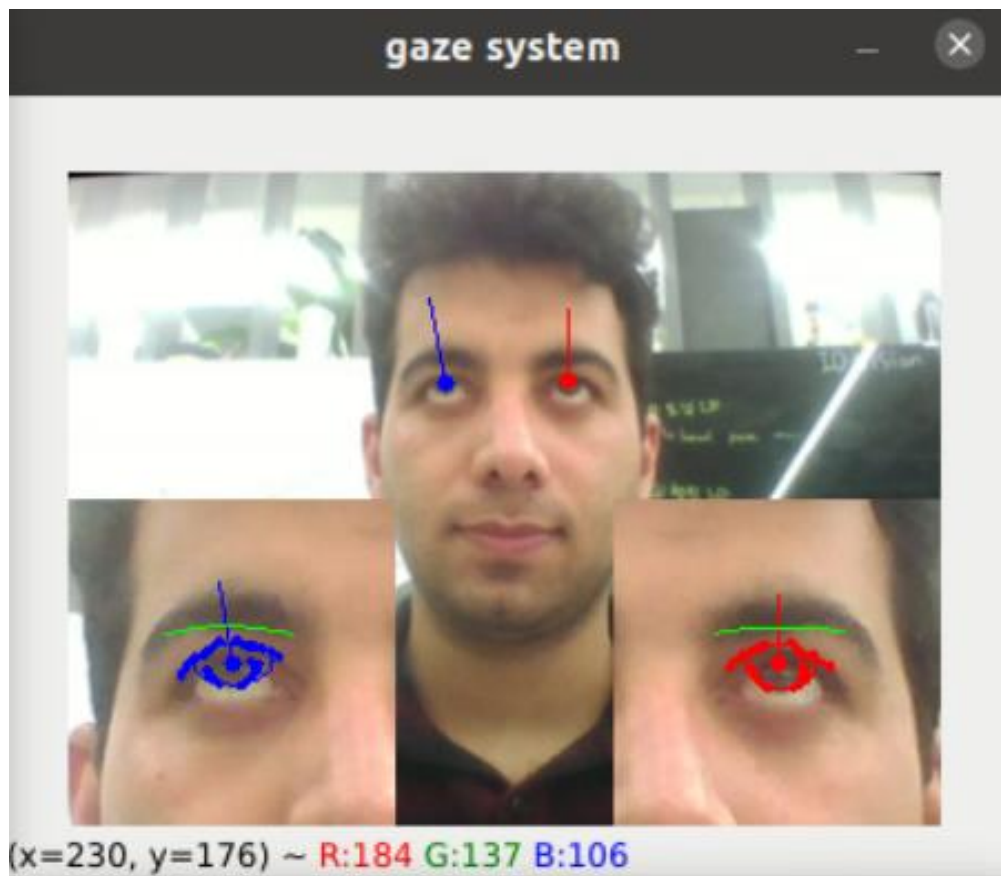
intensity of eyelid is almost equivalent to the iris (darkness). Therefore, eyelid detection and removal are a crucial to detect a clear iris region only. We employ the ransac algorithm to remove the eyelid and return the iris region only.



Part 2. Methodology



Finally, after removing the eyelid we are left with the iris region only.



Part 2. Methodology



Figure 6: Iris detection using binary thresholding algorithm of OpenCV in combination with mediapipe's facemesh solution.

2.2.2. Iris detection using pre-trained deep learning model

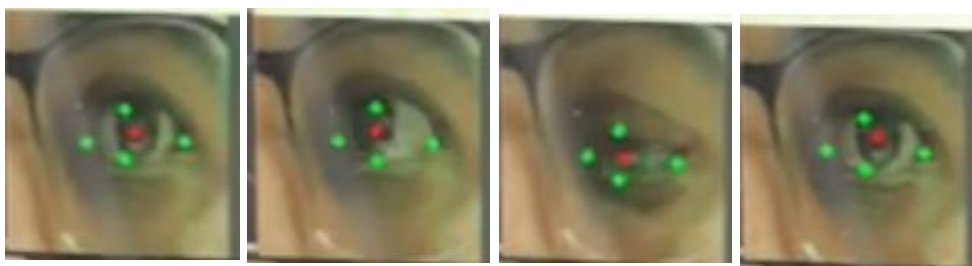


Figure 7: Iris detection using OpenCV. In some scenarios such as looking at the side or up and bottom, the OpenCV algorithm does not detect the iris section properly.

Part 2. Methodology

2.3. Deep learning-based gaze vector estimation

2.4. Datasets

2.4.1. Overview of Gaze Datasets

The shape and the size of the eye are important factors in estimating the gaze of a user. Because most of the existing gaze estimation datasets are collected in western countries the shape of and size of the eyes of western people are different than that of the Asian people. The lack of availability of Asian gaze estimation dataset requires a new data collection scheme. Further, some datasets are collected in a specific setting that does not suit our situation. A summary of available datasets is shown in Table 1.

Dataset	Participants	Head pose	gaze targets	Illumination	Duration	Images
UT Multiview [1]	50	8 + synthesized	160	1	1	64,000
McMurrrough et al [2]	20	1	16	1	1	videos
Villaneuva et al.[3]	103	1	12	1	1	1,236
Weidenbacher et al [4]	20	19	2-9	1	1	2,220
Smith et al. [5]	56	5	21	1	1	5,880
Eyediap [6]	16	continuous	continuous	2	1	videos
MPIIGaze [7]	15	continuous	continuous	daily life	45.1	213,659
Mayfarm Gaze (ours)	80	continuous	13	daily life	90	120,000

2.4.2. Mayfarm Soft dataset

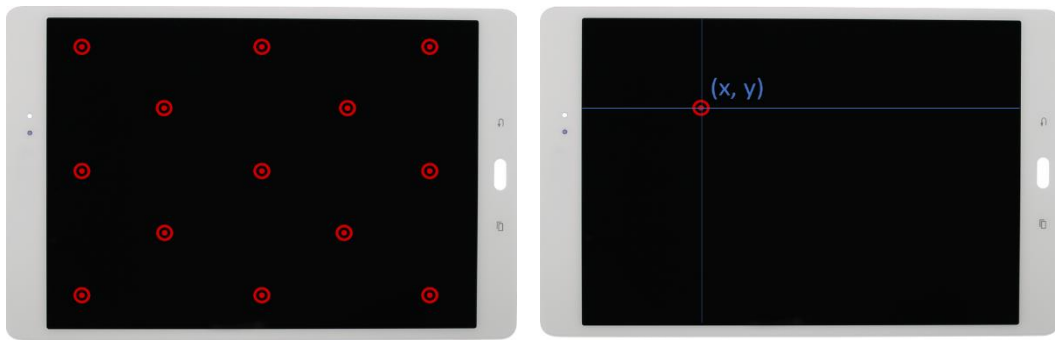
We design our data collection procedure with two objectives in mind, 1) to record the eye appearance of Asian people out of controlled laboratory settings, under different illumination condition and daily routines, 2) to collect gaze fixation images using an embedded device (i.e., a tablet) to make gaze estimation works highly accurately on small devices. Previous

Part 2. Methodology

gaze datasets use a small number of participants to collect the gaze data, therefore the eye-appearance of these datasets are limited. We collected the data from 120 participants to make sure that our dataset contains high variation in the eye-appearance. We also allowed the participants to move their heads slightly while fixating on the gaze target on the tablet screen.

2.4.2.1. Data Collection

Most of the gaze estimation datasets collected under controlled settings where the participants are the western people. The eye shape of Asian people differs from those of the western people thus a new dataset has to be built. Moreover, the available datasets are collected using a laptop which does not generalize well on small devices such as tablets and mobile phones. Mayfarm soft gaze dataset is collected using tablets that can generalize on mobile phone well.



2.4.2.2. Dataset Characteristics

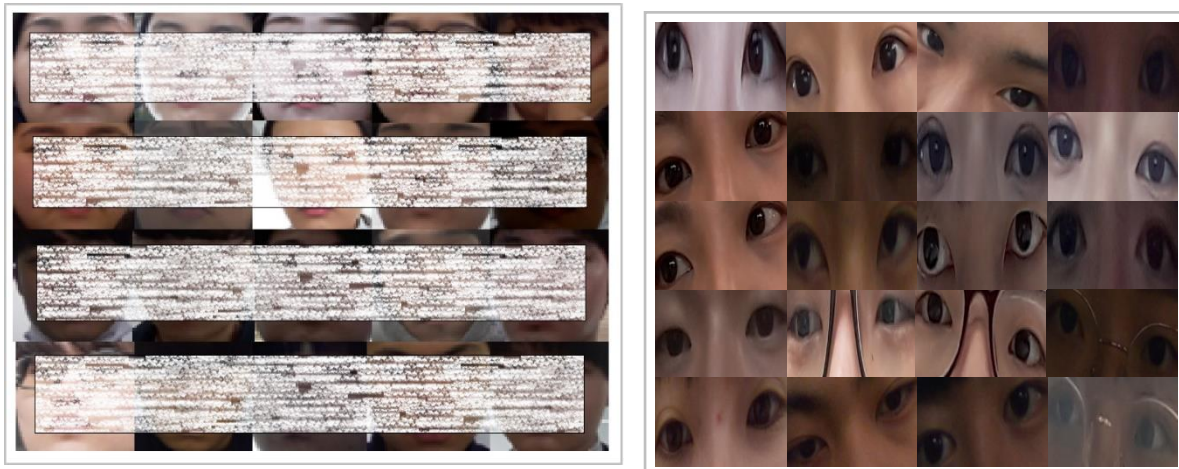
We collected a total of 120,000 images from 80 participants. To include variant eye-appearances we hired 40 male and 40 female participants. Each participant collected around 1,500 images during three shifts of time - morning, afternoon, and evening. The participants were occasionally given glasses to wear to ensure that the natural human-computer interaction is recorded. Our dataset contains a high variation of Asian eye-appearance, illumination condition, and head pose and is collected under open settings – the participants were asked to collect the data as they were interacting with the device naturally. Figure ... shows sample

Part 2. Methodology

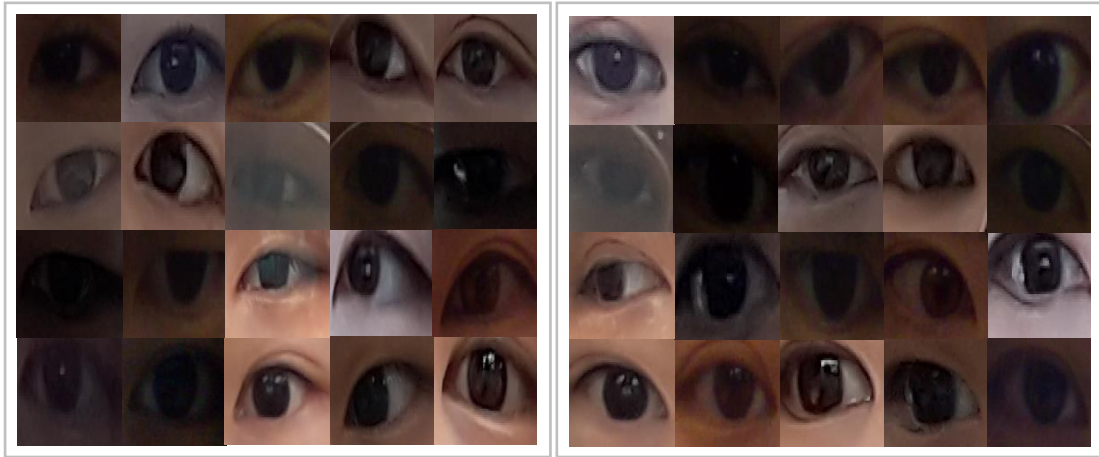
images of Mayfarm gaze dataset.



2.4.2.3. Data Wrangling and Pre-Processing



Part 2. Methodology



2.5. EfficientNet Model

EfficientNet is a CNN architecture and scaling method that equally scales all dimensions (i.e., depth, width and image resolution) using compound coefficient. The compound scaling method is justified by the intuition that if the input image is bigger, then network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image.

The block structure of EfficientNet is based on the inverted bottleneck residual blocks of MobileNetV2, in addition to squeeze-and-excitation blocks.

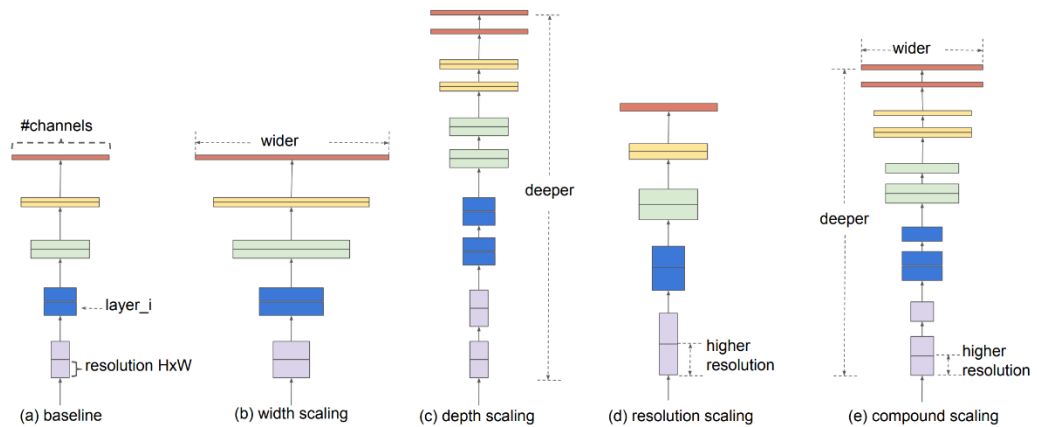


Figure 2. Model Scaling. (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

Part 3. EXPERIMENT AND RESULTS

3.1. Customized EfficientNet

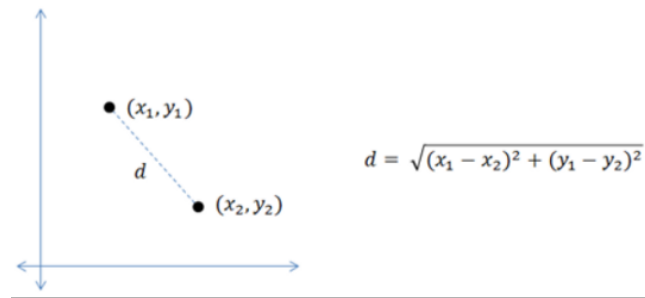
EfficientNet has a family of networks (B0 to B6) comprises of different input image size, depth and width with varying network parameters. We use the smallest model of EfficientNet (EfficientNetB0) which has around 5.3M parameters. We customize the model by removing certain layers to reduce the size of the model and add a regression layer at the top. The customized EfficientNetB0 model has around 300,000 params. We further change the input stream of the model from a single input to multi-input in order to feed the model with various inputs such as eye image and head pose vector. A high overview of the model is illustrated below and details of the model is provided in the appendix A – model overview.

3.1.1. Implementation details

We customize the efficientB0[8] model for gaze estimation and train it on the trainset of the dataset. We first crop the face image using the face landmarks, resize it to 224x224 and feed it to the network. Euclidean distance (a.k.a., L2) distance is used as an objective function of the model. We evaluate the performance of the network on the evaluation set of the dataset. The model is trained on 2GPUs (Tesla v100-32GB) with a batch-size of 1024 images and an initial learning rate of 0.01 for 60 epochs. A polynomial learning decay policy is used to reduce the learning rate as the number of epochs increases. The overview of the model architecture is provided in appendix A – model overview.

3.1.2. Loss function

We employ L2 (Euclidean distance) as a loss function to the model. The L2 distance measures the distance between the actual gaze (target) and the predicated gaze (model output) on the screen. L2 distance is defined as follows:


$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Part 3. Experiment and Results

3.1.3. Accuracy: training vs validation graph

We train the network for 60 epochs on the trainset (80%) and validate it on the validation set (20%) of the dataset. Our customized EfficientNetB0 model achieves around 2.4cm of L2 error on the validation set of our dataset. The learning graph of the model is illustrated in Figure 8.

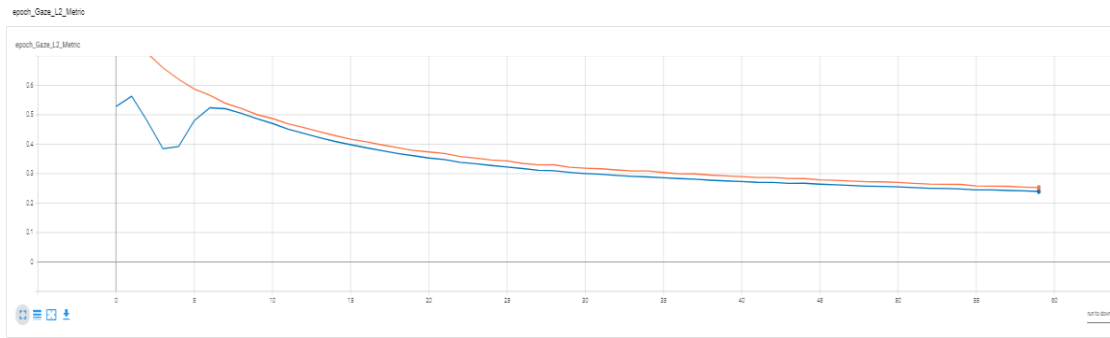


Figure 8: Gaze L2 metric. The orange and blue line indicates the loss on training set and validation set, respectively.

3.1.4. Distance estimation between eyes and camera

Scan and add notebook formula here

3.2. 3D back-projection of the eye

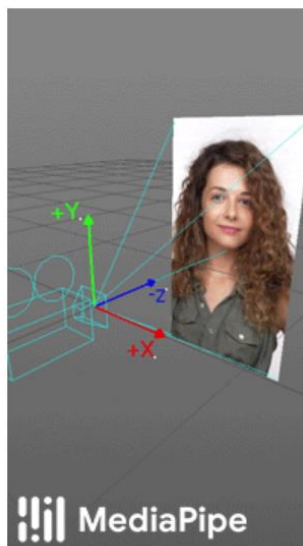


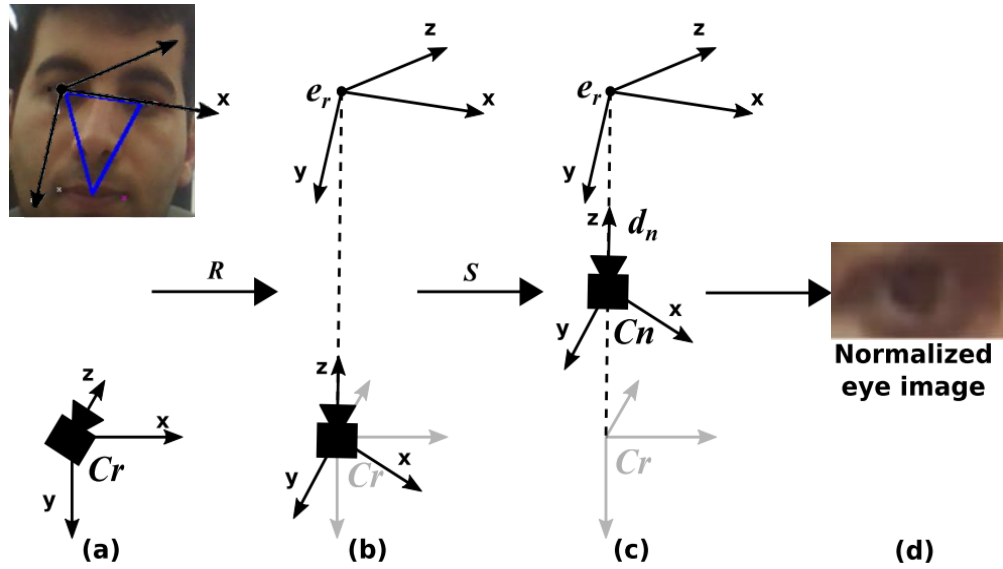
Fig 4. A visualization of multiple key elements in the Metric 3D space.

Part 3. Experiment and Results

3.3. Head pose estimation using the 3D head model

Head angles (i.e., yaw, pitch and roll) variances poses significant challenges for training a general gaze estimator. Data normalization is proposed to cancel out this geometric variability by mapping input images and gaze labels to a normalized space. Data normalization is used to align the training and test data for learning based gaze estimation by reducing the variances caused by head rotation and translation.

The key idea to standardize the rotation and translation between camera and face coordinate system is via camera rotation and scaling. The basic concept of the eye image



normalization is illustrated in Figure 9. (a) Starting from the 3D face model relationship between the head pose coordinate system centered at eye center e_r (top) and the camera coordinate system (bottom); (b) the camera coordinate system is rotated with a rotation matrix R ; (c) the world coordinate system is scaled with a scale matrix S ; (d) the normalized image is supposed to be equal to the one captured with this normalized camera. Figure 10 -12 shows real-examples of normalized eye images and original eye images in three different scenarios such as head rotation, translation and scale.

Figure 9: Basic concept of eye image normalization

Part 3. Experiment and Results

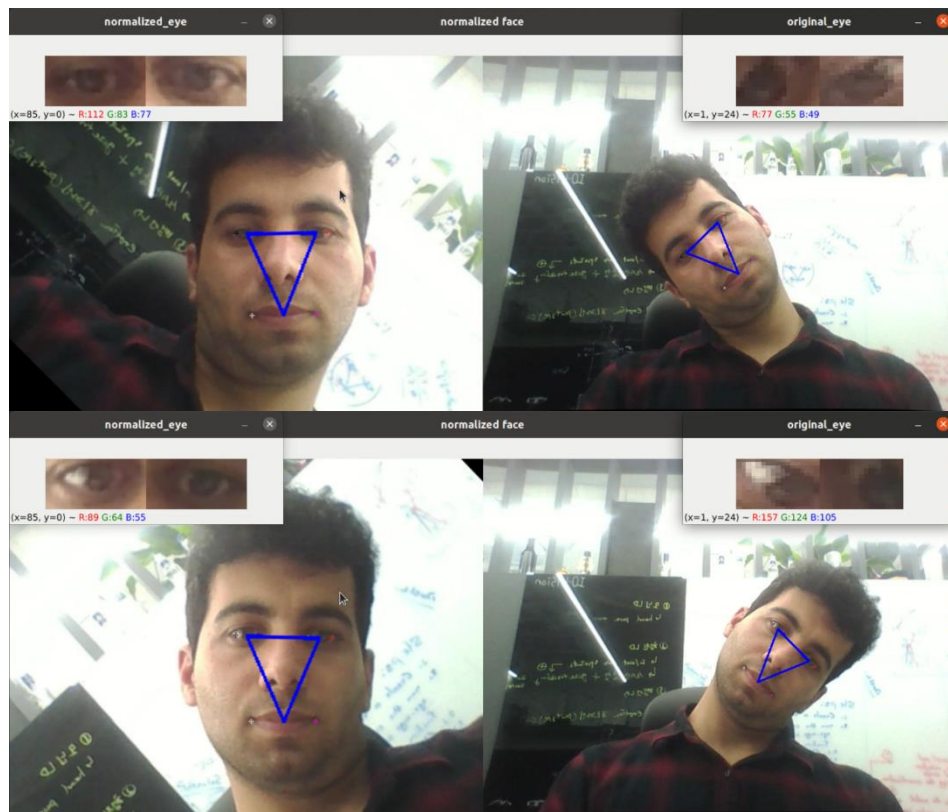


Figure 10: Canceling out the head-rotation affect using normalization.

Part 3. Experiment and Results

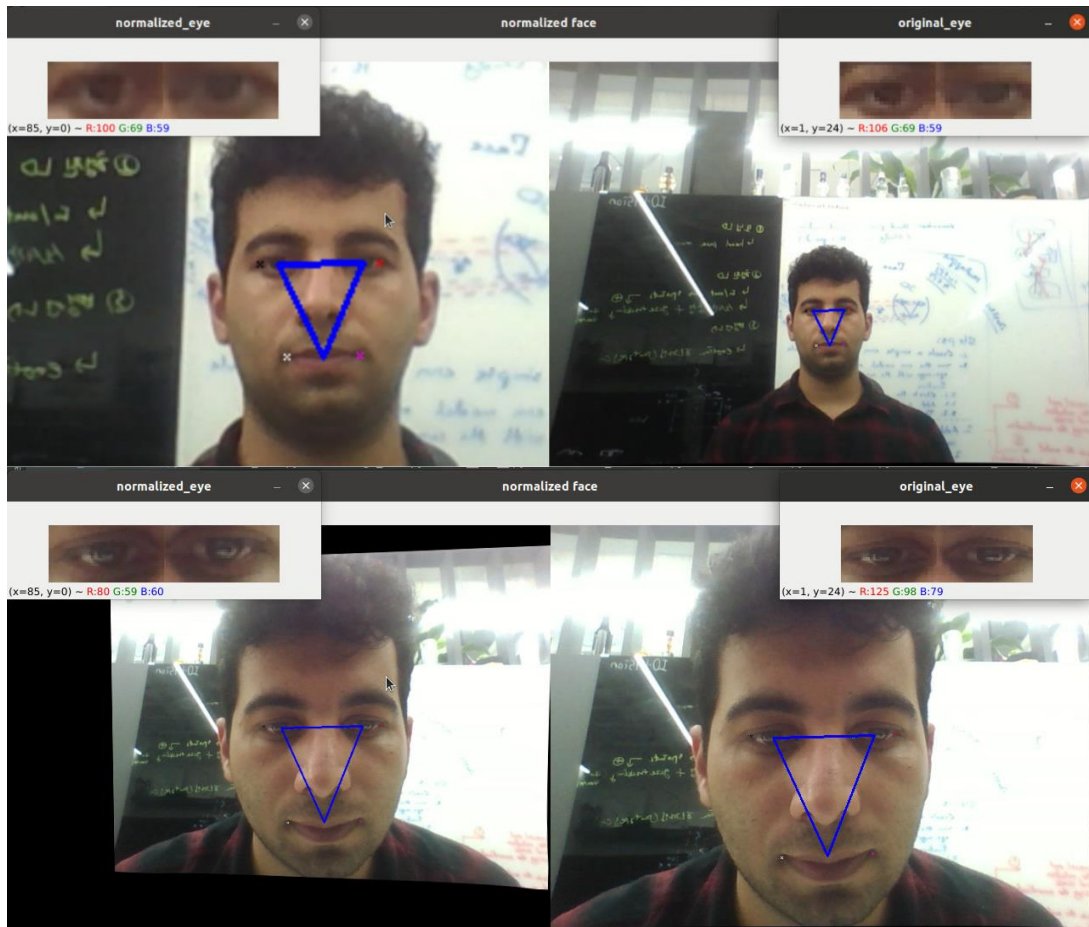


Figure 11: Canceling out the effect of scaling using data normalization.

3.4. Gaze smoothing algorithm

We use the intersection of two gaze vectors from the left and right eye with the screen plane as the final gaze vector. The final gaze vector is accompanied with noise and outliers (the extreme points). To remove the gaze outliers, we deploy a post-processing technique named gaze smoothing that smooths out the final gaze vector by measuring the current gaze output within a min and max threshold of history.

Part 3. Experiment and Results

3.5. Results

3.5.1. Qualitative analysis

The gaze estimation system is tested qualitatively by choosing five random people (three people from the Mayfarm soft company and three people from outside) to look at nine pre-defined fixation points on the screen (a.k.a., original fixation points). The users were allowed to change their head-pose during the test. We achieve an average L2 distance error of 2.6 cm from all 6 users.

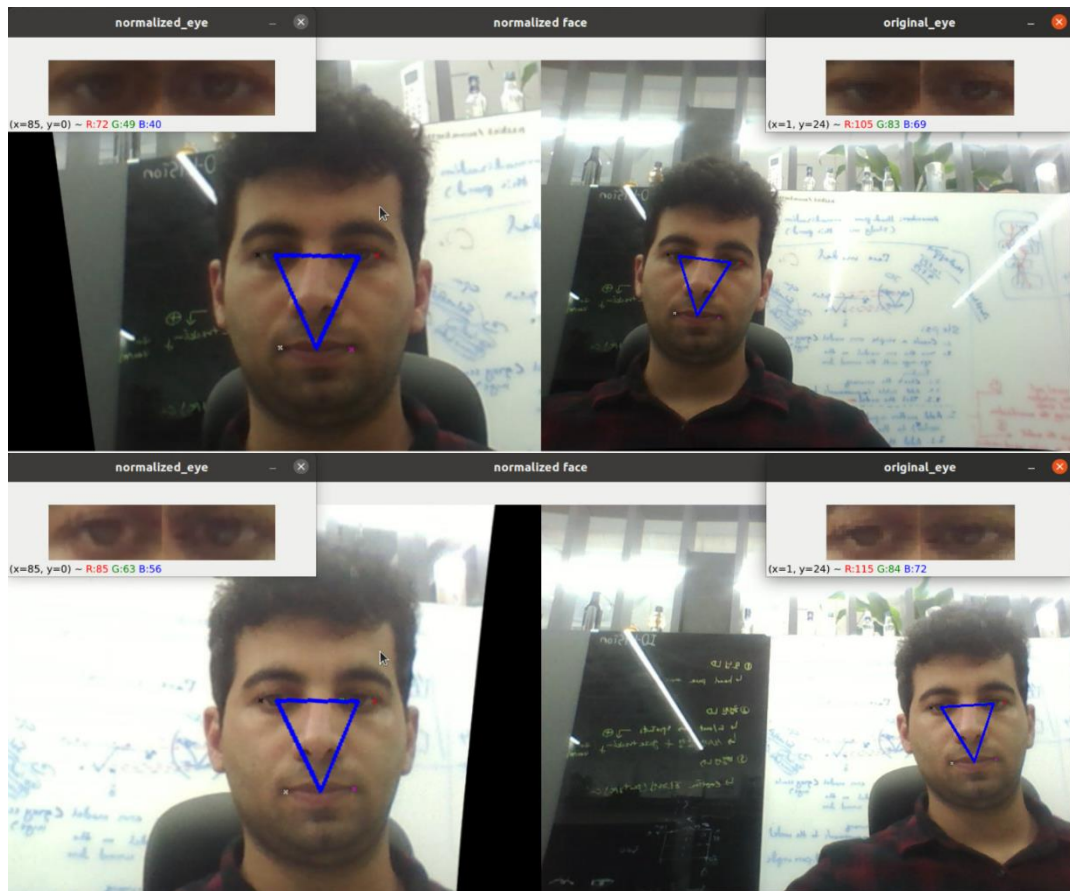
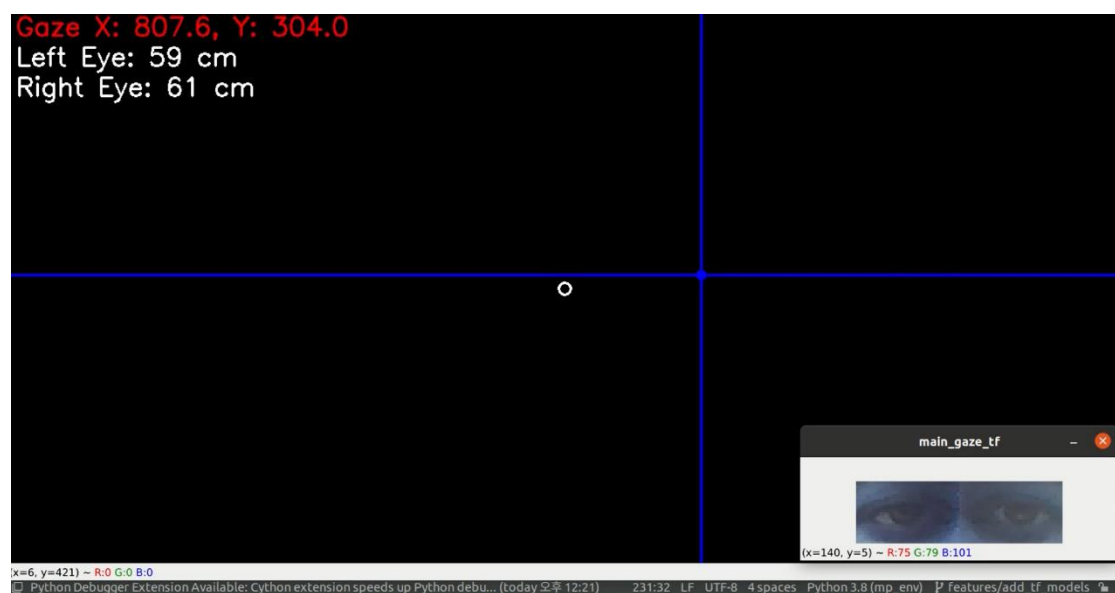
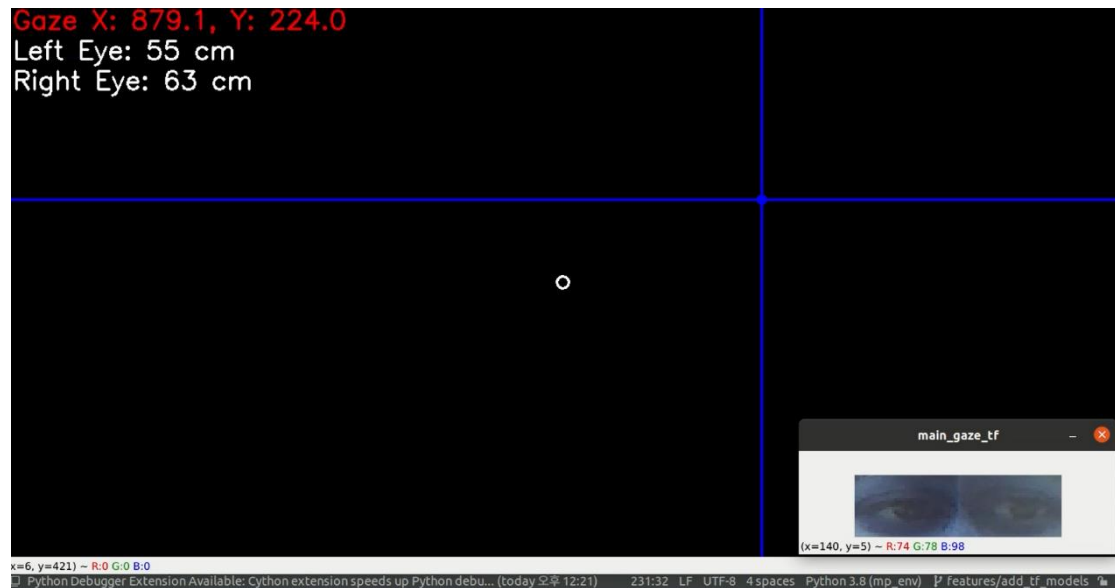


Figure 12: Canceling out the effect of translation using data normalization

Part 3. Experiment and Results

Gaze Screen



Part 4. REFERENCES

4.1. References

- [1] D. Lian *et al.*, “Multiview multitask gaze estimation with deep convolutional neural networks,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 10, pp. 3010–3023, Oct. 2019, doi: 10.1109/TNNLS.2018.2865525.
- [2] C. D. McMurrough, V. Metsis, J. Rich, and F. Makedon, “An eye tracking dataset for point of gaze detection,” *Eye Track. Res. Appl. Symp.*, pp. 305–308, 2012, doi: 10.1145/2168556.2168622.
- [3] A. Villanueva and R. Cabeza, “A novel gaze estimation system with one calibration point,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 38, no. 4, pp. 1123–1138, Aug. 2008, doi: 10.1109/TSMCB.2008.926606.
- [4] U. Weidenbacher, G. Layher, P. M. Strauss, and H. Neumann, “A comprehensive head pose and gaze database,” *IET Conf. Publ.*, no. 531 CP, pp. 455–458, 2007, doi: 10.1049/CP:20070407.
- [5] J.-W. Kim, “Webcam-Based 2D Eye Gaze Estimation System By Means of Binary Deformable Eyeball Templates,” *J. Inf. Commun. Converg. Eng.*, vol. 8, no. 5, pp. 575–580, Oct. 2010, doi: 10.6109/JICCE.2010.8.5.575.
- [6] K. A. F. Mora, F. Monay, and J. M. Odobez, “EYEDIAP: A database for the development and evaluation of gaze estimation algorithms from RGB and RGB-D cameras,” *Eye Track. Res. Appl. Symp.*, pp. 255–258, 2014
- [7] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, “MPIIGaze: Real-world dataset and deep appearance-based gaze estimation,” *arXiv*. 2017, Accessed: Jan. 21, 2021. [Online]. Available: <https://www.mpi-inf.mpg.de/MPIIGaze>.
- [8] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 10691–10700, May 2019, Accessed: Oct. 05, 2021. [Online]. Available:

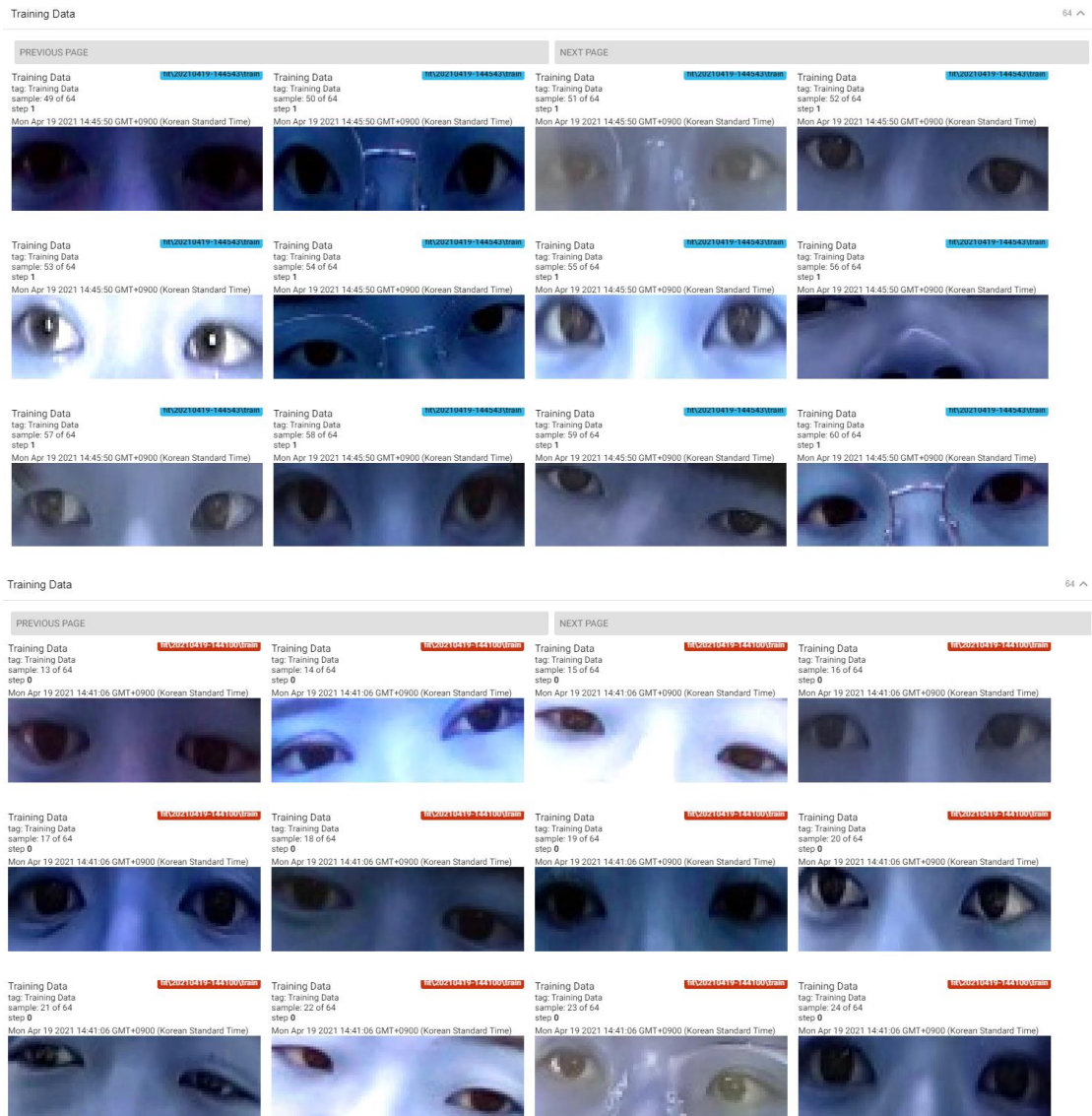
Part 5. Appendices

Part 5. APPENDICES

5.1. Appendix A. Visualization on tensorboard

5.1.1. Input Data Visualization

a. Batch of Double Eye Images



b. Batch of single (left and right) eye

Part 5. Appendices

Training Data

Figure 1 displays a grid of 12 eye images, arranged in 2 rows and 6 columns. Each image is accompanied by a caption indicating the training data, the GPT model used, and the number of training steps. The images show the effect of increasing the number of training steps on the quality of the generated eye images.

Top Row:

- Image 1 (Top Left):** Training Data: `img: Training Data`, `sample: 10 of 64`, `step: 10`. Caption: `Tue May 24 2021 13:08:24 GPT+4050 (Korean Standard Time)`. Model: `GPT+4050-10stepv0.000000`.
- Image 2 (Top Second):** Training Data: `img: Training Data`, `sample: 50 of 64`, `step: 50`. Caption: `Tue May 24 2021 13:08:24 GPT+4050 (Korean Standard Time)`. Model: `GPT+4050-50stepv0.000000`.
- Image 3 (Top Third):** Training Data: `img: Training Data`, `sample: 100 of 64`, `step: 100`. Caption: `Tue May 24 2021 13:08:24 GPT+4050 (Korean Standard Time)`. Model: `GPT+4050-100stepv0.000000`.
- Image 4 (Top Fourth):** Training Data: `img: Training Data`, `sample: 500 of 64`, `step: 500`. Caption: `Tue May 24 2021 13:08:24 GPT+4050 (Korean Standard Time)`. Model: `GPT+4050-500stepv0.000000`.
- Image 5 (Top Fifth):** Training Data: `img: Training Data`, `sample: 1000 of 64`, `step: 1000`. Caption: `Thu May 13 2021 11:10:19 GPT+4050 (Korean Standard Time)`. Model: `GPT+4050-1000stepv0.000000`.
- Image 6 (Top Sixth):** Training Data: `img: Training Data`, `sample: 5000 of 64`, `step: 5000`. Caption: `Thu May 13 2021 11:10:19 GPT+4050 (Korean Standard Time)`. Model: `GPT+4050-5000stepv0.000000`.

Bottom Row:

- Image 7 (Bottom Left):** Training Data: `img: Training Data`, `sample: 10 of 64`, `step: 10`. Caption: `Thu May 13 2021 11:10:19 GPT+4060 (Korean Standard Time)`. Model: `GPT+4060-10stepv0.000000`.
- Image 8 (Bottom Second):** Training Data: `img: Training Data`, `sample: 50 of 64`, `step: 50`. Caption: `Thu May 13 2021 11:10:19 GPT+4060 (Korean Standard Time)`. Model: `GPT+4060-50stepv0.000000`.
- Image 9 (Bottom Third):** Training Data: `img: Training Data`, `sample: 100 of 64`, `step: 100`. Caption: `Thu May 13 2021 11:10:19 GPT+4060 (Korean Standard Time)`. Model: `GPT+4060-100stepv0.000000`.
- Image 10 (Bottom Fourth):** Training Data: `img: Training Data`, `sample: 500 of 64`, `step: 500`. Caption: `Thu May 13 2021 11:10:19 GPT+4060 (Korean Standard Time)`. Model: `GPT+4060-500stepv0.000000`.
- Image 11 (Bottom Fifth):** Training Data: `img: Training Data`, `sample: 1000 of 64`, `step: 1000`. Caption: `Thu May 13 2021 11:10:19 GPT+4060 (Korean Standard Time)`. Model: `GPT+4060-1000stepv0.000000`.
- Image 12 (Bottom Sixth):** Training Data: `img: Training Data`, `sample: 5000 of 64`, `step: 5000`. Caption: `Thu May 13 2021 11:10:19 GPT+4060 (Korean Standard Time)`. Model: `GPT+4060-5000stepv0.000000`.

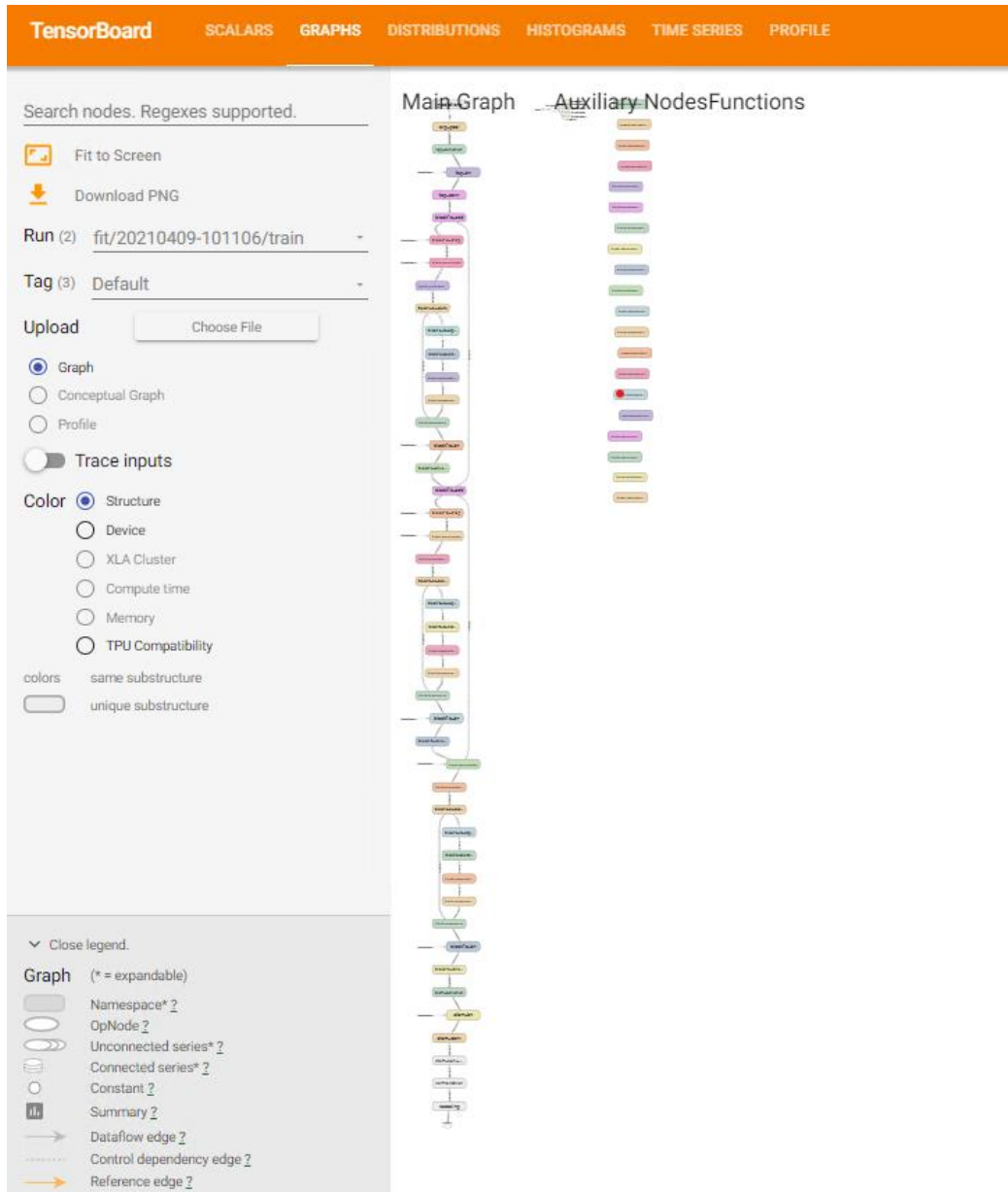
Training Data

Figure 1 displays a 4x4 grid of eye images, illustrating the effect of increasing blur levels on the training data. The images are arranged in four rows, each representing a different level of blur, and four columns, each representing a different sample. The top row shows the original training data (sharp). The subsequent rows show images with increasing blur, labeled as sample 25, 31, 33, and 34 of 64. Each image is accompanied by a caption indicating the training data source, the specific sample number, and the amount of blur (step 100).

Part 5. Appendices

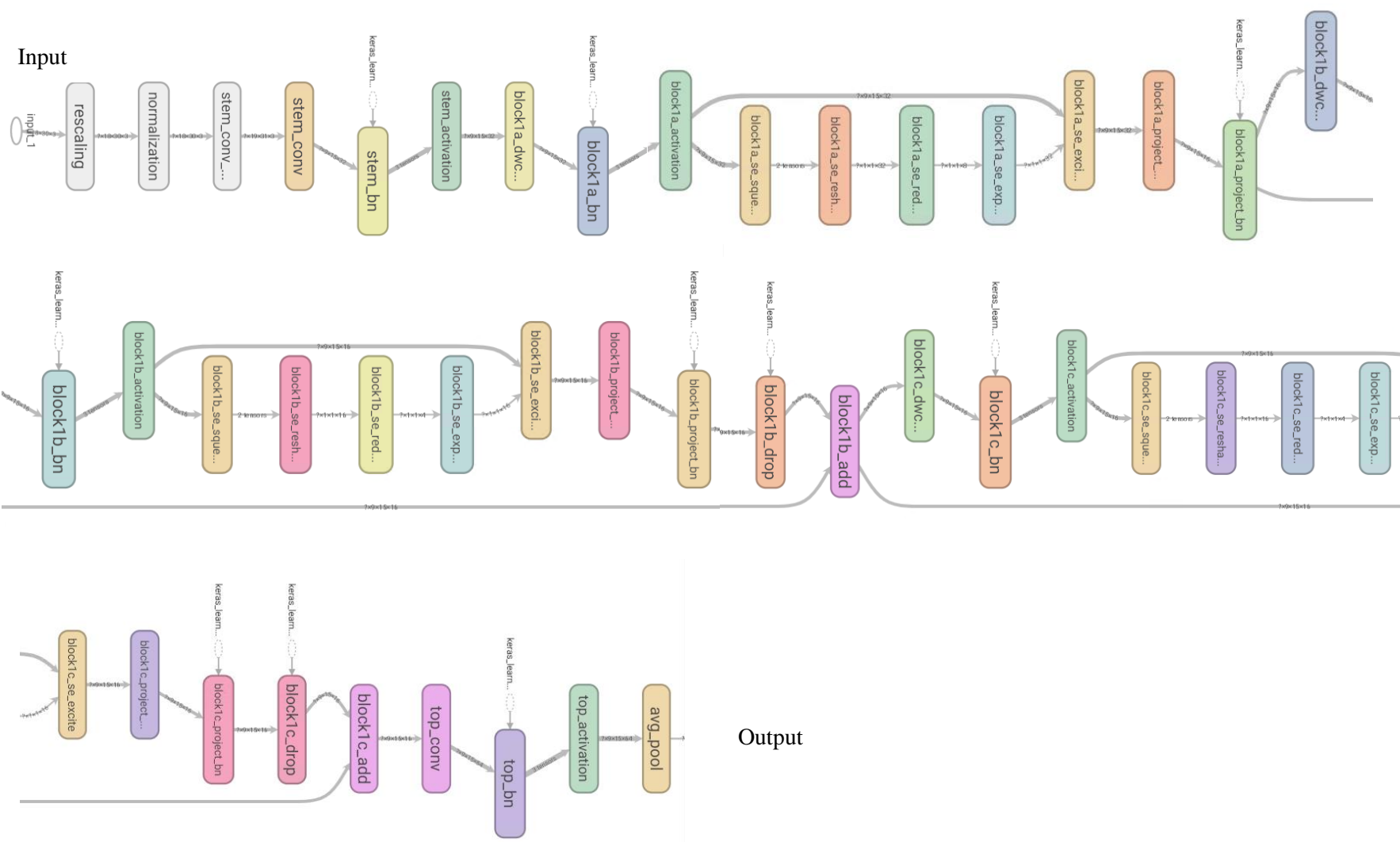
5.1.2. Model Visualization and Debugging

a. Customized efficientNetB0



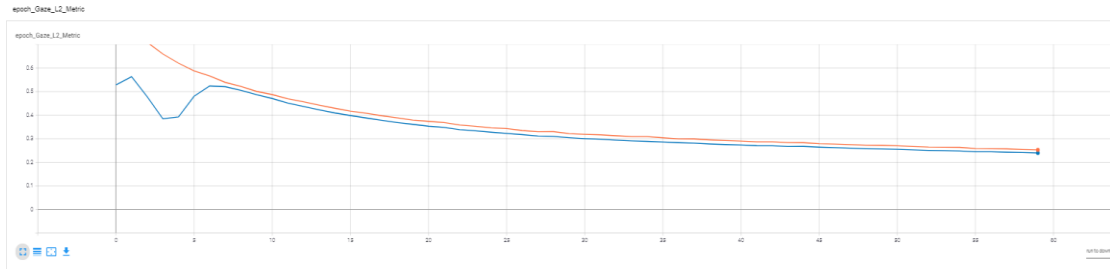
b. A closer look to the nodes of the model

Part 5. Appendices



Part 5. Appendices

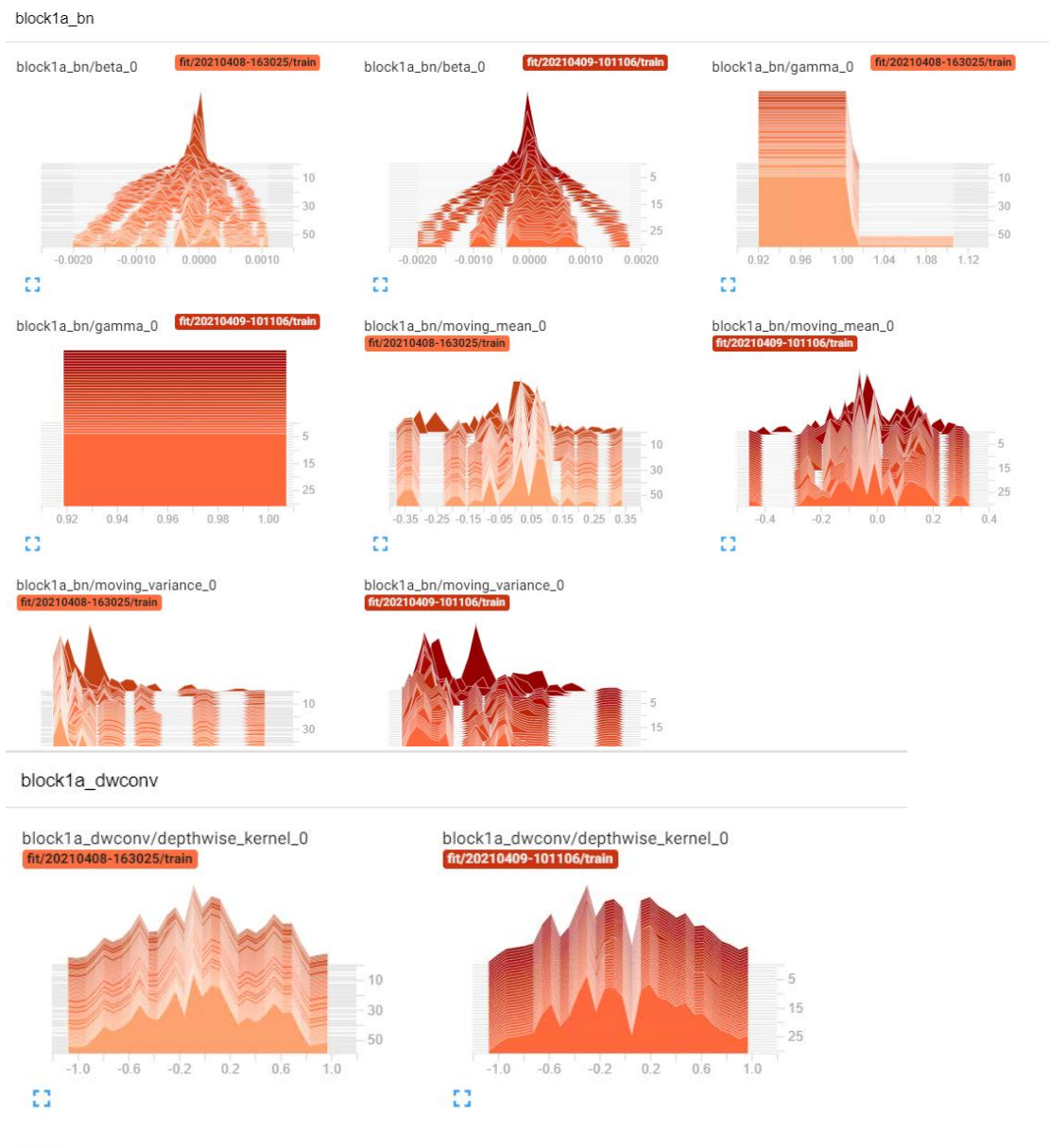
5.1.3. Model Training Graph



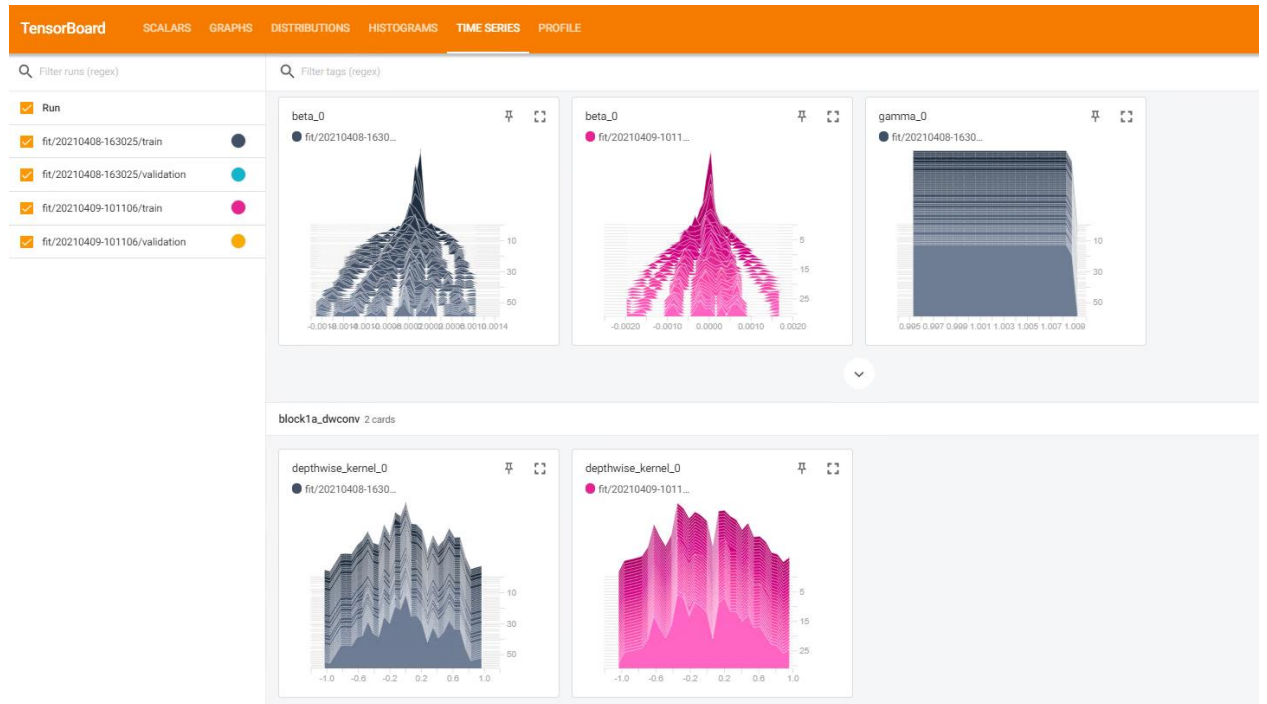
Blue line: Training loss

Orange Line: Validation loss

5.1.4. Model Weights

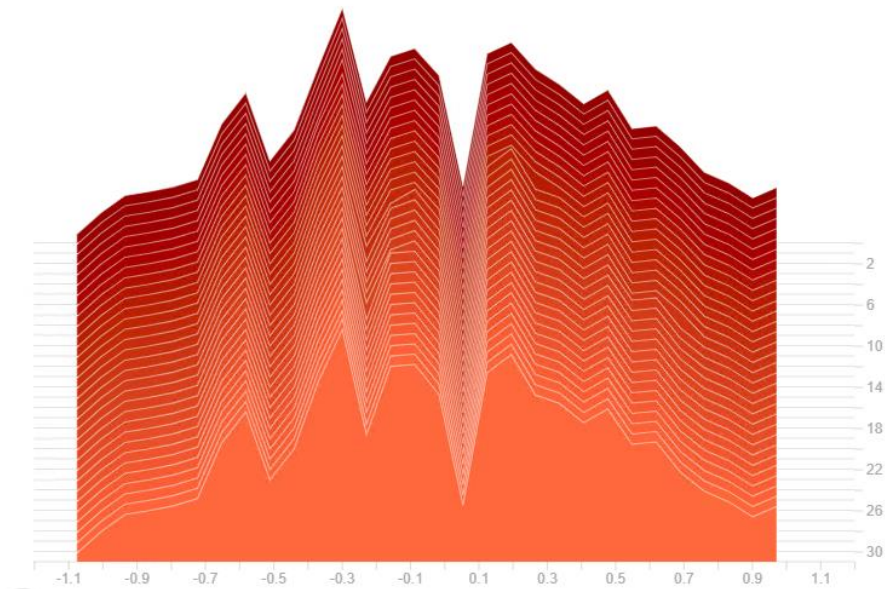


Part 5. Appendices



block1a_dwconv/depthwise_kernel_0

fit/20210409-101106/train



Part 5. Appendices

5.1.5. Model profiling

Memory Profile Summary

Memory ID
show memory profile for selected device

GPU...

#Allocation

499

#Deallocation

501

Memory Capacity

29.39 GiBs

Peak Heap Usage

high water mark in lifetime

0.72 GiBs

Peak Memory Usage

stack + heap, within profiling window

• Timestamp: 13.6 ms

• Stack Reservation: 0.00 GiBs

• Heap Allocation: 0.40 GiBs

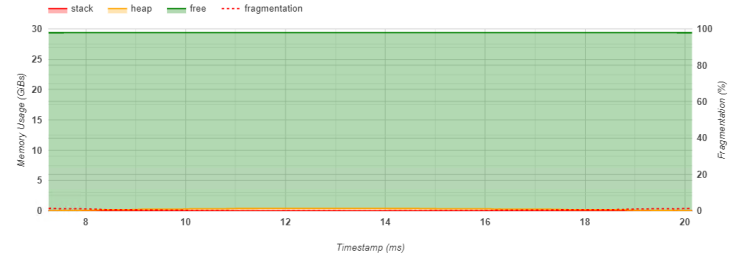
• Free Memory: 28.99 GiBs

• Fragmentation: 0.00%

0.40 GiBs

Memory Timeline Graph

Tips: Zoom In: left click and drag; Zoom out: right click; Metadata: click on heap data point.



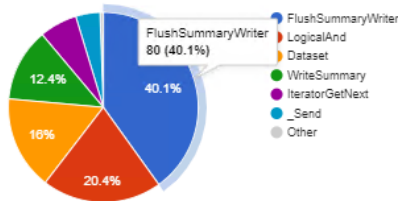
TensorFlow Stats

- (1) In the charts and table below, "IDLE" represents the portion of the total execution time on device (or host) that is idle.
(2) In the pie charts, the "Other" sector represents the sum of sectors that are too small to be shown individually.

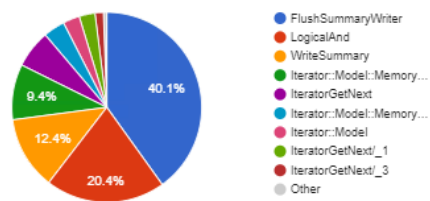
Include IDLE time in statistics

No

ON HOST: TOTAL SELF-TIME (GROUPED BY TYPE)
(in microseconds) of a TensorFlow operation type



ON HOST: TOTAL SELF-TIME
(in microseconds) of a TensorFlow operation



Memory Breakdown Table

Operation

Note: Showing active memory allocations at peak usage within the profiling window. To avoid sluggishness, only the allocations with size over 1MiB are shown in the table below.

Op Name	Allocation Size (GiBs)	Requested Size (GiBs)	Occurrences	Region type	Data type	Shape
GazeNet/block2a_project_bnFusedBatchNormV3	0.002	0.002	1	output	float	[512,24,5,8]
GazeNet/block2b_project_bnFusedBatchNormV3	0.002	0.002	1	output	float	[512,24,5,8]
GazeNet/block2b_se_excite/mul	0.011	0.011	1	output	float	[512,144,5,8]
GazeNet/block1a_project_bnFusedBatchNormV3	0.004	0.004	1	output	float	[512,16,9,15]
GazeNet/block2a_expand_activation/Sigmoid	0.025	0.025	1	output	float	[512,96,9,15]
GazeNet/block2a_dwconv/depthwise	0.011	0.007	1	output	float	[512,96,5,8]
GazeNet/block3b_bnFusedBatchNormV3	0.005	0.005	1	output	float	[512,240,3,4]
GazeNet/block2b_activation/mul-0-4-TransposeNCHWToNHWC-LayoutOptimizer	0.011	0.011	1	output	float	[512,5,8,144]
GazeNet/block3a_expand_activation/Sigmoid	0.011	0.011	1	output	float	[512,144,5,8]
GazeNet/top_activation/Sigmoid	0.001	0.001	1	output	float	[512,64,3,4]
GazeNet/top_conv/Conv2D	0.001	0.001	1	output	float	[512,64,3,4]
GazeNet/block3b_expand_conv/Conv2D	0.005	0.005	1	output	float	[512,240,3,4]
GazeNet/block2b_expand_bnFusedBatchNormV3	0.011	0.011	1	output	float	[512,144,5,8]
GazeNet/block3b_se_excite/mul	0.005	0.005	1	output	float	[512,240,3,4]
GazeNet/block2a_expand_bnFusedBatchNormV3	0.025	0.025	1	output	float	[512,96,9,15]
GazeNet/top_bnFusedBatchNormV3	0.001	0.001	1	output	float	[512,64,3,4]
GazeNet/block2a_expand_activation/Sigmoid	0.025	0.025	1	output	float	[512,96,9,15]
GazeNet/block3a_expand_bnFusedBatchNormV3	0.011	0.011	1	output	float	[512,144,5,8]
GazeNet/block2b_bnFusedBatchNormV3	0.011	0.011	1	output	float	[512,144,5,8]
GazeNet/block3b_se_excite/mul	0.005	0.005	1	output	float	[512,240,3,4]
GazeNet/block2a_project_conv/Conv2D	0.002	0.002	4	output	float	[512,24,5,8]
GazeNet/top_conv/Conv2D	0.001	0.001	1	output	float	[512,64,3,4]
GazeNet/top_activation/mul-0-1-TransposeNCHWToNHWC-LayoutOptimizer	0.001	0.001	1	output	float	[512,3,4,64]

Part 5. Appendices

5.2. Appendix

