Lightweight Convolutional Neural Network for Image Processing Method for Gaze Estimation and Eye Movement Event Detection

JOSHUA EMOTO¹ YUTAKA HIRATA^{1,a)}

Received: December 19, 2019, Accepted: January 6, 2020

Abstract: Advancements in technology have recently made it possible to obtain various types of biometric information from humans, enabling studies on estimation of human conditions in medicine, automobile safety, marketing, and other areas. These studies have particularly pointed to eye movement as an effective indicator of human conditions, and research on its applications is actively being pursued. The devices now widely used for measuring eye movements are based on the video-oculography (VOG) method, wherein the direction of gaze is estimated by processing eye images obtained through a camera. Applying convolutional neural networks (ConvNet) to the processing of eye images has been shown to enable accurate and robust gaze estimation. Conventional image processing, however, is premised on execution using a personal computer, making it difficult to carry out real-time gaze estimation using ConvNet, which involves the use of a large number of parameters, in a small arithmetic unit. Also, detecting eye movement events, such as blinking and saccadic movements, from the inferred gaze direction sequence for particular purposes requires the use of a separate algorithm. We therefore propose a new eye image processing method that batch-processes gaze estimation and event detection from end to end using an independently designed lightweight ConvNet. This paper discusses the structure of the proposed lightweight ConvNet, the methods for learning and evaluation used, and the proposed method's ability to simultaneously detect gaze direction and event occurrence using a smaller memory and at lower computational complexity than conventional methods.

Keywords: Deep learning, Saccade, Blink, Pupil, Multi-task

1. Introduction

Eye movements reflect various human emotions, perceptions, and states of attentiveness, drowsiness, etc. [1], [2], [3]. Eye movement, therefore, has gained attention as an objective indicator of human conditions, and has potential applications in various fields such as automobile safety, marketing, sports, personal healthcare, etc. Eye movement measurement devices are also now being used as input interfaces for smartphones and games [4]. Eye measurement technologies, therefore, are foreseen to have an increasingly large impact on society.

In this study, we propose a new eye image processing method based on a head-mounted video-oculography (VOG) measurement system using a compact arithmetic unit. This measurement device is portable and can be used for measuring eye movements outdoors or in experiments that involve moving subjects. Since VOG can non-invasively measure eye movement sequence, it is used in a wide range of applications other than in inferring human conditions [5].

VOG measurement systems estimate gaze direction (center of pupil) through image processing of eye images taken with a head-mounted camera. Accurate gaze estimation has recently been demonstrated to be possible with the application of deep learning. For example, DeepVOG [6] uses convolutional neural net-

work (ConvNet) for robust gaze estimation at an accuracy of approximately 0.5 degrees.

The application of deep learning in conventional methods like in DeepVOG is premised on the use of a PC equipped with a GPU and therefore requires a large memory and involves high computational complexity because of the large number of parameters used in ConvNet. Real-time gaze estimation and event detection, therefore, have been difficult to carry out using small devices with limited computing resources. Also, in DeepVOG for example, post-processing of the ConvNet output after computing gaze direction requires the use of a separate algorithm for detecting eye movement events such as blinking, fixed gaze, or saccades.

In this paper, therefore, we propose a new image processing method for gaze inference and event detection that is suitable for use with compact devices. The proposed method uses an end-to-end, batch-processing neural network (hereinafter referred to as the “model”) to process the outputs of the two tasks of gaze estimation and eye movement event detection based on the inputted eye movement video. As a result of the simultaneous processing of the two tasks, the proposed method does not require a separate algorithm for post-processing. Memory consumption and computational complexity of ConvNet were reduced by using MobileNetV2 [7] for one part of the model.

First, in this paper, we will describe the details of the structure of the model used in the proposed method and learning method of the model. We will then explain the method for creating the

¹ Chubu University, Kasugai, Aichi 487–8501, Japan

^{a)} yutaka@isc.chubu.ac.jp

Fig. 1 Overview of the proposed method. The model of the proposed method is composed of three ConvNets; namely, the feature extraction component, the keypoint detection component, and the event detection component. After learning for the keypoint detection component, the model adds the event detection component and carries out transfer learning for event detection. Upper left: eye movement video as input. Lower left: keypoints and eye movement events as outputs. Outline arrows indicate the time flow, black arrows indicate the data flow from input to output, and the rounded boxes indicate the processing carried out along the black arrows.

datasets for learning and evaluation of the model and present the results of the evaluation using the created datasets.

2. Proposed Method

2.1 Outline

Figure 1 shows the overview of the proposed method. The eye video (input video) indicated on the upper left of the figure is the input to the model, and the keypoints (output keypoints) and eye movement events (output events) indicated in the lower right of the figure are the outputs of the model. Outline arrows indicate the time flow and black arrows indicate the data flow. The rounded boxes indicate the processing carried out along the black arrows.

The model of the proposed method is composed of three convolutional neural networks (ConvNet); namely, the feature extraction component, the keypoint detection component, and the event detection component. First, the feature extraction component of the model computes the feature map from each frame of the eye video and outputs the sequence of feature maps for all the frames. The keypoint detection component then detects the keypoints based on the feature map. In particular, it outputs the coordinates for nine keypoints: center of pupil (red point on the keypoint image on the lower right), two endpoints each for the minor

axis (green points) and the major axis (blue points) of the pupil ellipse, and the four apex points of the curved line along the outer corner, inner corner, upper eyelid, and lower eyelid of the eyes (orange points). The event detection component detects the eye movement event of each frame from the feature map sequence. In particular, it detects and outputs the type of eye movement event (“output event” in lower right) in each frame as fixation, blink, or saccade. In accordance with the number of detected eye movement events, there are three feature dimensions in the output layer of the proposed method. Any number of eye movement events can be set for detection by adding corresponding datasets and increasing the number of feature dimensions of the output layer.

Although parameters other than the nine keypoints and the three eye movement events above can be outputted as detection results, in this study, we chose these keypoints and eye movement events for detection because they are the widely used indicators in the analysis of eye movements. Also, although the model of the proposed method uses feature maps outputted by the feature extraction component to process two tasks, the number of tasks to be processed can be increased to three or more. For example, to detect drowsiness from eye movements, an independent neural network that uses feature maps as input can be added to expand

the model to simultaneously detect drowsiness along with the two tasks performed by the model.

2.2 Learning of the Model

Learning in ConvNet usually requires a large amount of data. However, since there are few publicly available eye video datasets that can be used in the learning and evaluation of the proposed method's model, there was a need to independently create datasets for learning and evaluation. Acquiring large amounts of eye videos and annotating them, however, is not easy. DeepLabCut [8], however, was able to successfully learn accurate keypoint detection for different subjects such as humans, mice, and horses, even with only a small amount of annotated images. Since DeepLabCut enables accurate keypoint detection through transfer learning of a model previously trained based on large-scale data, we also decided to use transfer learning in this study. Since transfer learning enables reusing pre-trained neural networks for different tasks and data, it is used as a method for relearning new data.

Although DeepLabCut uses ResNet [9] for transfer learning, to achieve accurate detection of keypoints and events in real time in the proposed method, we used MobileNetV2 [7] because it enables faster processing at a discrimination accuracy comparable with ResNet. MobileNetV2 has a publicly available pre-trained dataset on the Internet called ImageNet [10].

Training of the model is carried out through mini-batch gradient descent method using error back-propagation. To enable detection regardless of the resolution of the image inputted to the model, for training, we used images randomly expanded and reduced through bi-linear interpolation. The coordinates of the keypoints were also expanded and reduced accordingly.

2.3 Structure of the Model

As explained above, the model of the proposed method is composed of three ConvNets; namely, the feature extraction component, the keypoint detection component, and the event detection component. The structure of each ConvNet is explained below.

2.3.1 Feature Extraction Component

The inputs for the feature extraction component are the frame-by-frame eye videos, and its outputs are their feature maps. In this model, the output of MobileNetV2 is used as the output (feature maps) of the feature extraction component. The resolution of the feature map is computed as follows based on the resolution of the input image. First, the convolutional layers of MobileNetV2 either have stride (s) or not; resolution of the output image is the same as the resolution of the input image for the layers without stride, but is lowered for those that have. Generally, if the resolution of the input image for convolutional layer is i , the output resolution (o) for i can be computed using the following equation [11]:

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1 \quad (1)$$

The first term in parenthesis represents the floor function, where p and k are the padding number and kernel size, respectively, for each layer.

In this model, a 14-layer MobileNetV2 is used. Of the 14 layers of MobileNetV2, four have stride and parameters of those layers are $s = 2$, $p = 1$, $k = 3$. Therefore, by repeatedly applying Equation 1 for the four layers, the resolution of the feature map becomes around 1/16 of the input image resolution.

Also, the number of dimensions is increased by 1.4 times in each convolutional layer to increase the expressive power of the model. Therefore, the output will have 136 feature dimensions for each dimension of the input feature.

The output of the feature extraction component will be the input for the keypoint detection component and the event detection component described below.

2.3.2 Keypoint Detection Component

As with the output layer of DeepLabCut, the keypoint detection component is composed of two transposed convolution layers (tConv layers). In the ConvNet of the keypoint detection component, the coordinates of the keypoints are computed based on score map and location refinement (locref) [12].

“Keypoint detection” in the lower left of Fig. 1 shows the score map and locref matrices outputted by the two tConv layers based on the feature map. The score map maps the coordinates of the space dimension to the coordinates of the input image and expresses the probability that a keypoint exists in those coordinates. Since one feature dimension corresponds to one keypoint, there are nine feature dimensions in the proposed method because there are nine keypoints for detection. The number of space dimensions (resolution) outputted by tConv is twice the number of space dimensions of the inputted feature map. Since the space dimension number of the input image is around 1/16, increasing the number of space dimensions of the feature map to twice would not restore the original space dimension number, wherein it is reduced to around 1/8. Therefore, the score map can be said to roughly express the coordinates of the keypoints in the input image. The coordinates of the green box in the score map in Fig. 1 points to the location where the corresponding keypoint (center of pupil in this case) exists with high probability. Whereas, locref expresses the distance that is needed to accurately infer the coordinates of the keypoint, i.e., the distance between the input image coordinates, which are mapped in the same way as in the score map, and the coordinates of the keypoint. The longitudinal and transverse components of the distance respectively express the two feature dimensions. Since two feature dimensions are needed for one keypoint, there are 18 locref feature dimensions in the proposed method.

“locref” in the lower left of Fig. 1 shows the inferred distance of the transverse component from the center of the pupil for each coordinate. The green box represents the coordinates that correspond to the maximum value in the score map. The brightness of each coordinate corresponds to the distance for correcting the keypoint coordinates. Figure 1 shows that the distance needed for correction becomes smaller towards the green box. The coordinates of the keypoint are computed by using only the maximum value of the score map from among the locref matrix values and adding that locref value to the coordinates of the green box.

For the loss functions of keypoint detection, as with DeepLabCut, cross-entropy error based on the sigmoid function is used for

the score map, and the Huber loss function for locref.

2.3.3 Event Detection Component

The event detection component is composed of a two convolutional layers and a Global Average Pooling (GAP) layer. The structure of the event detection component is shown in the upper right (“Event Detection”) of Fig. 1. The length and width of the box in the figure represent the space dimensions, the depth represents the feature dimension, and the outline arrow (time) represents the time dimension. The input of the event detection component is the feature map sequence, with the frame rate of the input video set to 60 FPS. The event detection component estimates the eye movement event for each frame from the information for five frames around each frame (83 ms). For this purpose, in the first convolutional layer (conv1), feature dimension number is reduced to 64 through convolutional operations for 1 space dimension and 5 time dimensions. Next, in the second layer (conv2), the feature dimension number is reduced to the number of events through convolutional operations for 3 space dimensions and 3 time dimensions. And, in the GAP layer, the average for each feature dimension in the space dimension is obtained, and the probability of each event is computed. For the loss function of event detection, cross-entropy error based on the softmax function is used.

The “output event” shown in the lower right of Fig. 1 is an example of event detection results showing the sequence of events wherein 1 represents the event with maximum output value and 0 represents the other events, after executing the above operations for all frames of the input video.

Since the model of the proposed method is composed of operations that are not dependent on space dimension, i.e., through convolutional and GAP layers, its structure is not dependent on the resolution of the input image, and arbitrary resolutions for the input image can therefore be used. Also, since the feature extraction component and keypoint detection component are composed of operations that are not dependent on time, processing for keypoint detection in particular can be carried out from one eye image input.

3. Evaluation Method

The validity of the proposed method was evaluated based on the following five criteria: (1) whether the feature map extracted by the feature extraction component through transfer learning reflects features that are effective for keypoint detection and eye movement event detection, (2) whether keypoint detection is possible through the use of the feature map, and, at the same, (3) whether event detection is possible through the use of pre-trained feature map in keypoint detection, and also, (4) whether it is possible to apply the proposed method for different devices and subject data, and (5) whether it is scalable for multi-task learning related to eye movement. The eye movement measurement device used for obtaining the datasets for evaluation and the evaluation procedures are described below.

3.1 Eye Movement Measurement Device

For the evaluation, coordinates of eye keypoints estimated using a currently available VOG-based eye movement measurement device were used as training data for the proposed method.

Fig. 2 Learning and evaluation procedure.

In particular, we used the EyeSeeCam (EyeSeeTec) [13] as the measurement device. EyeSeeCam is widely used for medical purposes and can take eye videos for both eyes at a maximum of 500 FPS and accurately record eye rotation angle (both yaw and pitch). Also, to evaluate the performance of the proposed method when applied to eye videos obtained with a different eye movement measurement device, we fabricated a head-mounted device (hereinafter referred to as “RPiCam”) by attaching an infrared LED camera (Kuman, model SC15) to a Raspberry Pi Zero W. RPiCam only has a function for sending images wirelessly through Wi-Fi, wherein image processing was carried out in the laptop PC to which the images were sent.

3.2 Procedures

Transfer learning in the ConvNet of the proposed method was carried out in this study through the following procedures to evaluate the above five criteria ((1) to (5)).

First, since the keypoint detection and event detection in the proposed method have different time scales despite being related tasks, the model of the proposed method is evaluated by learning keypoint detection and event detection separately. In other words, although keypoint detection involved only learning the eye images with different gaze directions by frame, event detection required learning each different image with different eye movement events separately. Therefore, to simplify the training algorithm for the ConvNet of the proposed method, we perform transfer learning for event detection independently of transfer learning for keypoint detection.

Figure 2 shows the procedures for transfer learning. The rectangles in the figure show that transfer learning is carried out for the ConvNet up to that point before the arrow. The contents of the rectangles indicate the type of learning and the names of subject groups (A, B, and C) and datasets (dev1, dev2, dev3, and dev4). The white rectangles represent transfer learning for keypoint detection, and the gray rectangles represent the transfer learning for event detection. The rounded boxes at the bottom of the figure represent the testing of the two tasks using the datasets for testing (test1, test2). In the evaluation, keypoint detection was trained using three datasets, and each task was tested using the dataset for testing, after training of event detection using one dataset.

Each dataset was created from three subject groups (A, B, and C). Subject Group A data were measured using both EyeSeeCam and RPiCam, once for each device, and were used for

Table 1 Summary of dataset.

Device	Subject Group	Number of videos/subjects	Keypoint detection		Number of videos for event detection ¹		
			Dataset name	Number of images	Dataset name	Blink	Saccade
EyeSeeCam	A	5	dev1	500			
	A	5	dev2	250	dev4	50	100
RPiCam	B	8	dev3	400			
	C	8	test1	400	test2	80	160

¹ Blink and saccade videos include fixation events.

evaluating keypoint detection through the proposed method using both devices (Details on data number, etc. are explained in Section 4.1). The data were also used for evaluating event detection using RPiCam. Subject Group B data were measured using RPiCam, and were used for evaluating the proposed method for data obtained from different subjects through transfer learning of ConvNet trained with Subject Group A data. Subject Group C data were not used in training of ConvNet, and were only used in evaluating ConvNet trained for keypoint detection and event detection.

4. Datasets

Six types of datasets were created for the four learning operations and two tests described in Section 3.2. First, we conducted an experiment for recording eye videos. Next, we extracted data each for keypoint detection and event detection from the recorded eye videos. Each set of extracted data was then annotated.

4.1 Experiment

Eye movements were recorded from 21 subjects to create the datasets through the procedure below.

Subjects were asked to sit on a chair in a dark room and fix their heads by placing their chins on a chin support at a distance of 62 cm from a display (35-inch, MARS3500-B, BenQ) with a screen size of 67 deg width and 31 deg length. Points of focus for the subjects to fix their gaze on were created by generating a single white point in the center of the screen, as well as 8, 12, 16, 20, 24, 28, 32, 36, 40, and 44 white points arranged at equal intervals respectively inside 10 concentric circles with a radius of 3, 5, 8.5, 12, 16, 20, 24, 28, 32, and 36 deg. From the generated points, a total of 135 points were displayed within the screen.

Subjects were asked to fix their gaze once on a particular point according to a specified order. The particular point of focus was specified by encircling the point in a circle. The circle was moved randomly to encircle each of the 135 points for around 1 second each. Eye movements during this procedure were recorded for a total of around 2 minutes and 30 seconds. An eye video file included the subject's eye movement events, such as saccade, fixation, and natural blinking movements.

Of the 21 subjects, 5 were assigned to Subject Group A, and 8 each to Subject Group B and C. A total of 26 eye videos were recorded at 256 FPS using EyeSeeCam for the five subjects in Group A, and at 60 FPS using RPiCam for all the 21 subjects (total of 26 eye videos).

The resolution of the videos recorded using EyeSeeCam was 188 (width) by 100 (height) pixels. For the videos recorded using RPiCam, images were clipped as shown in the lower right of Fig. 1 by manually setting the region for each subject. Resolution

of images, therefore, varied for each video and ranged from 180 to 270 in width and from 85 to 150 in height for this experiment.

4.2 Extraction of Data

Images or videos were clipped from the eye videos recorded using EyeSeeCam and RPiCam to create datasets for training and evaluation of keypoint detection and event detection tasks. **Table 1** shows the number of extracted images and videos and the names of the datasets.

4.2.1 Keypoint Detection

This task required images where the subjects gaze in different directions. In the model of the proposed method, since keypoint detection is not dependent on time, we created the dataset for keypoint detection from images, rather than from videos that could include many images of the eye gazing in the same direction.

Images for the dataset were obtained by clipping 100 fixation images at one-second intervals from the start of the experiment from each of the 5 videos for Subject Group A recorded using EyeSeeCam (total of 500 images), and 50 images each from the 5, 8, and 8 videos for Subject Groups A, B, and C recorded using RPiCam (total of 250, 400, and 400 images each group).

4.2.2 Event Detection

Since the event detection component required processing that is dependent on time, we created the dataset for event detection using videos.

Videos for the dataset were obtained by visually checking portions of the video that included blinking and saccade movements from Subject Group A and C videos recorded using RPiCam, and randomly extracting from them 10 videos that included blinking and 20 videos that included saccade at 16 frames each video (0.27 seconds).

4.3 Annotation

Extracted data were annotated as follows.

4.3.1 Keypoint Detection

The extracted images for keypoint detection were annotated by assigning the coordinates for the nine keypoints by visual observation. Annotation of pupil center coordinates for images recorded using EyeSeeCam was based on the EyeSeeCam output. We used an independently developed GUI application for annotation. The coordinates for the five keypoints including the pupil center and surrounding keypoints were annotated so that the ellipse matches the edges of the pupil by adjusting three parameters; namely, the center of the ellipse, the length of the major and the minor axes, and the rotation angle. The coordinates for the other four keypoints were annotated by visually checking the location of the keypoint and assigning the coordinates by pointing the mouse over the eye image.

4.3.2 Event Detection

Ongoing eye movement events (fixation, blink, or saccade) were annotated on each frame of the videos extracted for event detection by visually checking them while playing back the video frame by frame. All frames that did not include blinking or saccade movements were annotated as fixation.

In videos with blinking events, blinking occurred throughout all the 16 frames; while in videos with saccade events, saccade occurred in 4 to 12 of the 16 frames.

5. Evaluation Results

ConvNets of the proposed method were trained according to the procedure shown in Fig. 2. As described in the learning method in Section 2.2, input images were randomly reduced and expanded to between 0.64 to 1.5 times, with a learning rate of 0.005. The following are the results of the evaluation of keypoint detection and event detection tasks.

5.1 Keypoint Detection

5.1.1 Learning

Transfer learning was carried out using dataset dev1 with the pre-trained MobileNetV2 as the feature extraction component. In training of keypoint detection, ConvNet parameters for the feature extraction component and keypoint detection component were updated. The 500 images of dataset dev1 were divided in an 8 : 2 ratio for each subject as training data (400 images) and validation data (100 images). Its learning curves are shown in the topmost graph of Fig. 3, with epoch number on the horizontal axis and loss on the vertical axis. The black curve represents the learning curve for the training data, and the gray curves represent the learning curves for the five subjects using dev1 validation data. From the learning curves, it was determined that convergence of learning took place at 1,000 epochs, and the ConvNet immediately after 1,000 training epochs was considered as the pre-trained ConvNet based on dev1.

The pre-trained ConvNet based on dataset dev1 was used for transfer learning with dev2. Using the same ratio as for dataset dev1, 200 of the 250 images of dev2 were used as training data and the remaining 50 as validation data. Learning curves for dev2 are shown in the middle graph of Fig. 3. As with training with dev1, it was determined from the learning curves that convergence took place at 1,000 epochs, and the ConvNet immediately after 1,000 training epochs was considered as the pre-trained ConvNet based on dev2.

The pre-trained ConvNet based on dataset dev2 was used for transfer learning with dev3. Using the same ratio as for the above training processes, 320 of the 400 images of dev3 were used as training data and the remaining 80 as validation data. Learning curves for dev3 are shown in the bottom graph of Fig. 3. From the learning curves, it was determined that convergence took place at 200 epochs, and the ConvNet immediately after 200 training epochs was considered as the pre-trained ConvNet based on dev3.

5.1.2 Accuracy

The accuracy of keypoint coordinates inferred through the proposed method was evaluated using the root-mean-square error (RMSE) between the annotated keypoint coordinates and the cor-

Fig. 3 Learning curves for each transfer learning process. Upper graph shows learning curves when dev1 was used, middle graph when dev2 was used, and bottom graph when dev3 was used.

responding keypoint coordinates inferred through the proposed method as the estimation error. **Figure 4** shows histograms of the estimation errors for the inferred keypoint coordinates for each image using test1 testing dataset, with RMSE on the horizontal axis and number of images on the vertical axis. Histograms are arranged from top to bottom according to the dataset used, with the results for estimation using ConvNet pre-trained based on dev1 at the topmost row, followed by those using dev2 and those using dev3. From left to right, histograms are arranged according to the three groups of the nine keypoints; namely, pupil center (1 point), pupil ellipse (4 points: 2 points each for the major and the minor axes), and eyelid edges (4 points: outer corner, inner corner, upper eyelid, and lower eyelid). The values on the horizontal axis of the histogram range from 0 to 20 pixels, and results beyond that range were considered as failure. The number on the upper right of each histogram is the number of images for which estimation failed among the total of 400 images. Most of the images for which estimation failed were for blinking.

Table 2 shows mean and standard deviation of estimation error after learning of each dataset for pupil center, pupil ellipse, and eyelid edges in pixel. The average size of input images is 200

Fig. 4 Histograms of errors of keypoint detection for pupil center (left), pupil ellipse (center), and eyelid edges (right) for each transfer learning process. Upper row shows histograms when dev1 was used, middle row when dev2 was used, and bottom row when dev3 was used.

Table 2 Accuracy of keypoint detection.

Dataset	RMSE (Mean $\pm$ SD pixel)		
	Pupil center	Pupil ellipse	Eyelid edges
dev1	4.09 $\pm$ 5.04	9.52 $\pm$ 10.5	16.9 $\pm$ 12.8
dev2	2.35 $\pm$ 3.87	5.48 $\pm$ 5.88	6.68 $\pm$ 4.29
dev3	2.14 $\pm$ 4.19	5.03 $\pm$ 6.19	6.08 $\pm$ 4.19

pixels in widths and 100 pixels in height.

5.2 Event Detection

5.2.1 Learning

Transfer learning for the ConvNet in the event detection component was carried out using the dataset for event detection (dev4), by learning the ConvNet pre-trained based on dev2 dataset for keypoint detection from Subject Group A. In the training for event detection, parameters for the feature extraction component were fixed, and only the event detection parameters were updated. From among the 150 dev4 videos, 120 were used as training data and 30 as validation data.

Figure 5 shows the learning curves, with epoch number on the horizontal axis and loss on the vertical axis. The black curve represents the learning curve for the training data, and the gray curve represents the learning curve for the validation data. From the learning curves, it was determined that convergence of learning took place at 1,000 epochs, and the accuracy of event detection was evaluated using the dataset for testing (test2).

5.2.2 Accuracy

Table 3 shows the detection accuracy and F1 score for each eye movement event. Detection accuracy was computed as the ratio of correctly inferred eye movement events within the 3,840 frames that represent ongoing blinking, saccade, and fixation movements included in a total of 240 16-frame videos (80 blink videos and 160 saccade videos). Although the accuracy for detecting saccade is lower than the other events, this is attributed to

Fig. 5 Learning curves for event detection.

Table 3 Accuracy of event detection.

	Number of detected / total events	Accuracy	F1 score
Blink	1008 / 1165	86.5%	0.839
Fixation	1455 / 1747	83.3%	0.807
Saccade	568 / 928	61.2%	0.681

the relatively lower number of frames with saccade events, which could have led to insufficient learning. Judging only whether a saccade event occurred in each video of test2, detection rate was 87.0% (F1 score of 0.906).

6. Discussion

The following is a discussion of the five evaluation criteria mentioned in Chapter 3 based on the results of evaluation.

6.1 Keypoint Detection

The topmost row of Fig. 4 shows estimation errors computed using test1 dataset obtained using RPiCam for ConvNet trained

based on dev1 dataset obtained using EyeSeeCam. The estimation errors for pupil center (left) were smaller than the estimation errors for pupil ellipse (center) and eyelid edges (right). Also, cameras with different performances were used (EyeSeeCam and RPiCam) to obtain eye images at varying angles of view and resolutions. In other words, the model of the proposed method was able to detect keypoints of eye images that were recorded using RPiCam and were not used in learning, pointing to its ability to detect keypoints even when using a different device for capturing eye images.

Also, as shown in Fig. 4 and Table 2, estimation accuracy remained almost the same even after learning with additional subjects using dev2 in the middle row and dev3 in the bottom row. This points to the high versatility of the proposed method even when used for different subjects.

Since the proposed method was able to estimate pupil center coordinates at an error of around 2 pixels, it possesses an accuracy that is sufficient for application in gaze estimation for marketing and gaming purposes.

On the other hand, the learning curves for training data shown in Fig. 3 indicate that the loss values for convergence differ depending on the subjects, meaning that it is not enough to simply increase the number of subjects, but there is a need to add data in consideration of individual differences and differences in quality of images in each video.

Moreover, the histograms at the bottom of Fig. 4 show that there is smaller improvement in estimation error for transfer learning from dev2 to dev3 than for transfer learning from dev1 to dev2. This suggests that improving accuracy entails not only adding data, but also enhancing the ConvNet structure, such as improving the model based on dependency between keypoints.

In regard to the evaluation criteria mentioned in Chapter 3, since the proposed method enables keypoint detection for gaze estimation (criteria 2) and possesses versatility even when used with different devices or subjects (criteria 4), it can be applied for tasks using different devices and subjects.

6.2 Event Detection

Table 3 shows that the proposed method results in F1 scores above 0.8, indicating that it can detect eye movement events at high accuracy.

Also, since the feature extraction component parameters were not updated for event detection learning, the event detection results are also based on the same feature values as those for keypoint detection. This shows that the feature map outputted by the feature extraction component trained for keypoint detection reflects features that are effective not only for keypoint detection but also for eye movement event estimation.

In regard to the evaluation criteria, it was shown that the feature map of the trained ConvNet reflects features that are effective for keypoint detection and event detection (criteria 1), and that event detection using that feature map is possible (criteria 3). Also, in the event detection component, the gaze estimation task is added after training, wherein the eye movement event detection task is expanded, indicating that it is possible to expand the method for multi-task learning related to eye movement (criteria 5).

6.3 Trade-off between Accuracy and Lightness

There is a trade-off between accuracy and lightness of neural network. The number of parameters in the neural network employed in DeepVOG is around 37 million. In contrast, our lightweight model has around 1.3 million parameters. This reduction of parameters indicates the advantages of the proposed method in memory consumption and computational complexity. On the other hand, DeepVOG achieved around 1 pixel estimation error in pupil center detection when the resolution of images is 320×240 pixels. By contrast, the estimation error of the proposed method is around 2 pixels when the average resolution of images is 200×100 pixels. Users should choose either method by considering this trade-off.

6.4 Training Procedure

In this article, we designed multi-stage transfer learning as shown in Fig. 2 to evaluate five evaluation criteria. However, when users apply our proposed method to their own device or datasets, they should design an appropriate training procedure for the particular device or datasets.

In multi-stage transfer learning, inappropriate source of transfer learning reduces the performance of the inference model [14]. As shown in Fig. 4, the estimation error of eye lid edges on dev1 is bigger than other results. This result indicates that the trained model failed to learn to estimate eye lid edges on dev1 because of inappropriate source of transfer learning. Therefore, if users can collect a large number of annotated images from their own device, they can train the model from full scratch and multi-stage transfer is not necessary.

On the other hand, when collecting a large number of annotated images is difficult, multi-stage transfer learning is useful to efficiently train the model of the proposed method. At least in the estimation of pupil center, the model of our proposed method can be trained by using the multi-stage transfer learning described in this article on a small number of annotated images collected from users own device.

7. Conclusion

In this paper, we proposed a new image processing method based on deep learning to be used for VOG eye movement measurement devices. By using MobileNetV2 for the ConvNet to be used in image processing and by implementing end-to-end batch processing of gaze estimation and event detection, we were able to develop a model with minimal memory consumption and computational complexity. To demonstrate the validity of the proposed method, we designed a unique evaluation method and fabricated an inexpensive eye-movement-measuring device in addition to the use of an existing high-performance measurement device. We created original datasets from 21 subjects and used the datasets to train and evaluate the performance of the model. Results showed that the proposed method possesses versatility that allows it to be used independently from device or subjects as well as scalability for it to be used in multi-task processing.

The proposed method enabled ConvNets for extracting feature values that are effective in analyzing eye movement from eye images. The model of the proposed method is easily scalable to han-

dle three or more tasks depending on the purpose, and has potential for use in a variety of applications. For example, we believe that it would be useful in building eye-movement-analysis systems for end-to-end processing of eye videos for use in detection of movements not dealt with in this paper, such as smooth pursuit, optokinetic response, vestibulo-ocular reflex, and microsaccade.

The development of technologies related to deep learning has progressed rapidly in recent years. Other than GPUs, processors such as TPUs, which can rapidly execute neural network operations, are being developed for use in smartphones and embedded devices and are now available in the market. Methods similar to the proposed method, which carries out end-to-end batch-processing of eye movement analysis, unlike conventional methods, can reap the benefits of these new technologies.

References

- [1] Eckstein, M.K., Guerra-Carrillo, B., Singley, A.T.M. and Bunge, S.A.: Beyond eye gaze: What else can eyetracking reveal about cognition and cognitive development?, *Developmental Cognitive Neuroscience*, Vol.25, pp.69–91 (online), DOI: <https://doi.org/10.1016/j.dcn.2016.11.001> (2017).
- [2] Zhao, M., Gersch, T.M., Schnitzer, B.S., Doshier, B.A. and Kowler, E.: Eye movements and attention: The role of pre-saccadic shifts of attention in perception, memory and the control of saccades, *Vision Research*, Vol.74, pp.40–60 (online), DOI: <https://doi.org/10.1016/j.visres.2012.06.017> (2012).
- [3] Hirata, Y., Nishiyama, J. and Kinoshita, S.: Detection and prediction of drowsiness by reflexive eye movements, *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp.4015–4018 (online), DOI: 10.1109/IEMBS.2009.5333504 (2009).
- [4] Smith, J.D. and Graham, T.C.N.: Use of Eye Movements for Video Game Control, *Proc. International Conference on ACE*, pp.1–8, ACM (online), DOI: 10.1145/1178823.1178847 (2006).
- [5] Sakashita, Y., Fujiyoshi, H. and Hirata, Y.: Measurement of 3D Eye Movements Using Image Processing, *Journal of JSEM*, Vol.6, No.3, pp.236–243 (2006).
- [6] Yiu, Y.-H., Aboulatta, M., Raiser, T., Opey, L., Flanagan, V.L., zu Eulenburg, P. and Ahmadi, S.-A.: DeepVOG: Open-source pupil segmentation and gaze estimation in neuroscience using deep learning, *Journal of Neuroscience Methods*, Vol.324, p.108307 (2019).
- [7] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.: MobileNetV2: Inverted Residuals and Linear Bottlenecks, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.4510–4520 (online), DOI: 10.1109/CVPR.2018.00474 (2018).
- [8] Mathis, A. and Warren, R.A.: On the inference speed and video-compression robustness of DeepLabCut, *bioRxiv*, pp.1–10 (online), DOI: 10.1101/457242 (2018).
- [9] He, K., Zhang, X., Ren, S. and Sun, J.: Deep Residual Learning for Image Recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.770–778, IEEE Computer Society (2016).
- [10] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C. and Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision (IJCV)*, Vol.115, No.3, pp.211–252 (online), DOI: 10.1007/s11263-015-0816-y (2015).
- [11] Dumoulin, V. and Visin, F.: A guide to convolution arithmetic for deep learning, *ArXiv*, p.15 (2016).
- [12] Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M. and Schiele, B.: DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model, *European Conference on Computer Vision*, pp.5–9 (online), available from (<http://arxiv.org/abs/1605.03170>) (2016).
- [13] Schneider, E., Villgratner, T., Vockeroth, J., Bartl, K., Kohlbecher, S., Bardins, S., Ulbrich, H. and Brandt, T.: EyeSeeCam: An Eye Movement-Driven Head Camera for the Examination of Natural Visual Exploration, *Annals of the New York Academy of Sciences*, Vol.1164, No.1, pp.461–467 (online), DOI: 10.1111/j.1749-6632.2009.03858.x (2009).
- [14] Suzuki, A., Sakanashi, H., Kido, S. and Shouno, H.: Feature Representation Analysis of Deep Convolutional Neural Network using Two-stage Feature Transfer-An Application for Diffuse Lung Disease Classification, *IPJS Online Transactions*, pp.74–83 (2018).

Joshua Emoto received his B.S. (2015) and M.S. (2017) degrees in Computer Science and Engineering from Chubu University. He entered his Ph.D. program of the Graduate School of Engineering, Chubu University in 2017 and engaged in research on eye movements and attentiveness. He received the 2016 JSAI Annual

Conference Student Incentive Award and the 2016 Institute of Electronics, Information and Communication Engineers-Tokai Section Student Research Incentive Award.

Yutaka Hirata received his B.S. (1990) and M.S. (1992) degrees in Electrical and Electronic Engineering from Toyohashi University of Technology. He received his Dr. Eng. degree in System Information Engineering from Toyohashi University of Technology in 1995. He joined the Space Utilization Research Center of

NASDA (JAXA) as an Invited Scientist in 1995. He worked as a Research Associate at Washington University School of Medicine from 1997 (and as Overseas Research Fellow of the Japan Science and Technology Agency from 1998). He joined the Department of Electronic Engineering, Chubu University College of Engineering as Associate Professor in 2000 and became Professor of the Department of Computer Science in 2004 and Professor of the Department of Robotic Science and Technology in 2014. He is engaged in research on biomedical engineering and neuroscience of motor learning of eye movements. He received the IEICE Outstanding Paper Award in 1995 and 2012, the IEICE Inose Award in 1995, the International Symposium Society for Research on the Cerebellum Masao Ito Prize in 2011, the Japanese Neural Network Society Outstanding Research Award in 2011, the Japanese Neural Network Society Most Outstanding Research Award in 2017, the IEICE Human Communication Award in 2014. He is a member of the Vision Society of Japan, the Japanese Neural Network Society, the Japan Society of Aerospace and Environmental Medicine, IEEE, and the Society for Neuroscience.

(Communicated by Junichiro Yoshimoto)