Internship Final Report

Yucheng XIAO
UGA MOSIG M1
Grenoble, France
yucheng.xiao@etu.univ-grenoble-alpes.fr

Supervised by: Didier SCHWAB
MOSIG Supervisor: Renaud LACHAIZE.

I understand what plagiarism entails and I declare that this report is my own, original work.
Name, date and signature: Yucheng XIAO, 10/06/2022

Abstract

Software depending on gaze as the only input, such as *Gazeplay*[1], need a special device called eye-tracker to operate. Eye-tracker uses infrared sensors and cameras to identify the gaze of the user and using algorithms to report the actual point where the user is watching. However eye trackers are expensive, so using the camera of a laptop has become an alternative option. Thanks to the growth of computational power and new state-of-the-art machine learning method, such as *convolutional neural network*[2], analyzing gaze data from video captured by webcam has become more accurate. This report will demonstrate the performance difference of eye-trackers and webcams integrated in laptop, and introduce a method to improve the performance of webcam by adding attention layer to convolutional neural network.

1 Introduction

Many poly-disabled people are not able to use their limbs (such as hands or feet) or their voices in a precise way. Tools and games on digital terminals are thus inaccessible to them. In many cases, gaze is the most direct way that enable them to interact. *Gazeplay*, an free and open-source software using an external eye-tracker are created to satisfy this need. Eye-trackers on the market records the movement of the eyes and transcribe, with a point on the monitor, where the user is looking. However, off-shelf eye-trackers are fairly expensive and some of them has certain compatibility issues. The average price of an moderate eye-tracker is about 200€. Therefore using a front facing camera to capture the gaze of the users has become a feasible alternative options. Most laptops has front facing camera to capture the face of the laptop users, but the quality differs from laptop to laptop. An external webcam for desktop cost around 40€, which makes it significantly cheaper comparing to an eye-tracker, but it is less efficient.

Nowadays the use of artificial intelligence (AI) and convolutional neural networks has become a viable way to complete image-centric tasks, like eye-tracking tasks and analysis, which makes them options for improving the performance of webcams. In neural networks, attention is a technique that mimics cognitive attention. The effect enhances some parts of the input data while diminishing other parts.

This report will mainly focus on 2 subjects : compare of performance of eye-trackers and webcams integrated in laptop, and analyse the effect on performance of an additional attention layer in the convolutional neural network.

1.1 Issue correction on Gazeplay games

The first part of my work consist of working on issue#1572 of Gazeplay software : adding the option to modify the size and the color of progress bar. Initially the size are fixed and the color are restricted yo yellow. I have done the modification for 1 game and then I start working on the performance evaluations.

1.2 Performance evaluation and comparison on webcam and eye-tracker

[Burton *et al.*, 2014] shows that in certain area, webcams has some advantages over eye-trackers. In this part I have used scientific approaches to measure the performance of webcam and eye-trackers. I measured the performance from 3 criteria : precision, accuracy and gaze time, using a modified version of a game in *Gazeplay*. The work is based on the previous work of [Bandyopadhyay *et al.*, 2021b].

1.3 Implementation and evaluation of attention layer

The practice of deep learning also allows for a given neural network to improve at a given task when given enough sample data, which requires a relatively large supply of training data [Zhao *et al.*, 2018]. The dataset chosen for training is the *MPIIGaze* dataset introduced by [Zhang *et al.*, 2015]. To train the neural networks, a modified version of *pytorch_mpiigaze*[3] program proposed by [Zhang *et al.*, 2017] was used.

The attention layer are added on the LeNet [Lecun *et al.*, 1998] and ResNet [He *et al.*, 2015] network that *pytorch_mpiigaze* provided. A total of 4 different attention layers have been implemented :

[1] https://gazeplay.github.io/GazePlay/

[2] https://en.wikipedia.org/wiki/Convolutional_neural_network

[3] https://github.com/hysts/pytorch_mpiigaze

- ECA-Net, Efficient Channel Attention [Wang *et al.*, 2019];
- Coordinate Attention [Hou *et al.*, 2021];
- CBAM: Convolutional Block Attention Module [Woo *et al.*, 2018];
- SK-Net: Selective Kernel Networks [Li *et al.*, 2019].

The gaze result returned by the *pytorch_mpiigaze* program is a 2 dimensional vector, which correspond to the yaw and pitch of user's gaze direction. After the training, the *pytorch_mpiigaze* program reports and records the error in angle, which is the main metric of the evaluation. Another metric is the time consumed for training the network.

2 Performance evaluation and comparison on webcam and eye-tracker

A research performed by [Skovsgaard *et al.*, 2011] has stated that precision and accuracy are two very important measurements for an input device like webcam or eye-tracker. The Tobii test specification document by [Tobii, 2011] also pointed out the importance of these as well. Figure 1 gives a clear view of precision and accuracy.

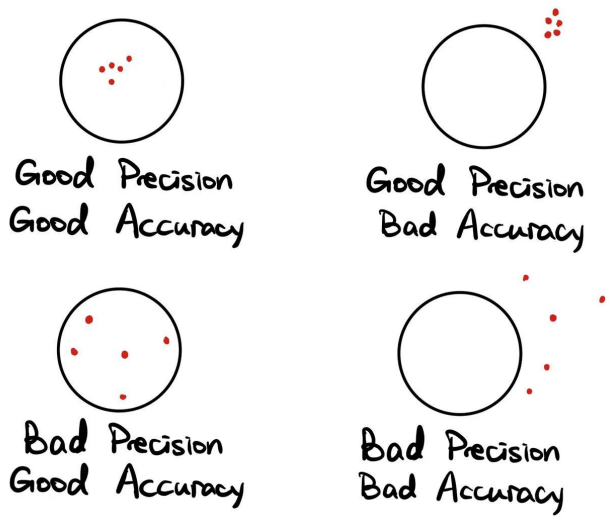

Figure 1: Behavior of precision and accuracy, inspired by [Tobii, 2011]

Another important criteria is the gaze time, which measures the latency between the user move his eyes and the change of position reported by the device.

Precision

Precision is defined as the ability of the eye tracker to reliably reproduce a measurement, i.e. a measure of variance in the recorded data. It is calculated via the Root Mean Square (RMS) of successive samples which is commonly used when calculating eye tracking system's precision. [Tobii, 2011].

Precision (P) is calculated via the Root Mean Square from the successive data points in degrees of visual angle θ_i between successive (x_1 ,y_1) to ($x_i + 1$, $y_i + 1$) samples, both for each eye individually and as a mean from the two (see formula 1). The unit of the precision figure is $pixels$.

$$P = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\theta_i)^2} = \sqrt{\frac{\theta_1^2 + \theta_2^2 + ... + \theta_n^2}{n}} \quad (1)$$

Ideally, the precision figure should be 0.

Accuracy

Accuracy is defined as the mean offset in pixels and gaze angles based on the distance in pixels (d_i) between the gaze position and the eye tracker reported position (see Equation 2).

$$A = \frac{1}{n}\sum_{i=1}^{n} d_i = \frac{d_1 + d_2 + ... + d_n}{n} \quad (2)$$

Gaze time

Gaze time measures the latency between the user move his eyes and the change of position reported by the device. In ideal condition, when the user moves his eyes, the device should report the new gaze position without delay. Given the time t_s between a new target is shown on the screen and the device report a new position, the gaze time t can be presented as (see Equation 3)

$$t = t_s \quad (3)$$

2.1 Evaluation setup

To measure the performance, *Gazeplay* has been chosen due to the fact that it has good support on eye-tracking devices and also allows a secondary mouse input.

Running $Gazeplay$ using eye-tracker

Gazeplay supports multiple Tobii eye-tracker devices, and it reads the position from the eye-tracker directly. In the evaluation, the Tobii 5 Eye-Tracker will be used for comparison.

Running $Gazeplay$ using webcam

Gazeplay does not support webcam as an input device. Currently it only support several Tobii eye-tracking devices and mouse as input. However, there are several existing applications on the market that allows moving the cursor on the screen depending the gaze data from the image captured by the webcam. For the experiment, a Chicony USB 2.0 camera integrated on the Clevo P750TM-G laptop will be used and an application ,GazePointer, is used due to its built-in calibration system and its open source characteristic. This software can process the video captured by webcams, proceeding the gaze data and move the cursor correspondingly.

Games used for evaluation

Out of more than 60 games to choose from, a modified version of the game $creampie$ has been chosen due to the fact that it can generate targets repeatedly across the screen. The target size is a circle with a radius of 300 pixels.

2.2 Evaluation procedure

The program will generate, has random position, a target for the user to gaze at. To clear target, the user has to gaze at the center of the target for 1 second. During the 1 second time when the user gaze at the target, the game will collect the position data of the gaze reported by the device at a rate of 10Hz. After that, the target is cleared, and the game will generate one. When a target is cleared, the program will compute the average precision, accuracy and gaze time and write it into a file. The test ends when 10 targets are cleared.

The difference between the evaluation method in [Bandyopadhyay *et al.*, 2021b] is mainly on :

- The image of target is replaced by a real target rather than an image of animal before, so the tester can locate and gaze the center of image more precisely;
- The size of the target has increased due to the accuracy of webcam is not ideal;
- Position of gaze are fetched every 100ms, and the result of performance (precision, accuracy and gaze time) are calculated in real-time.

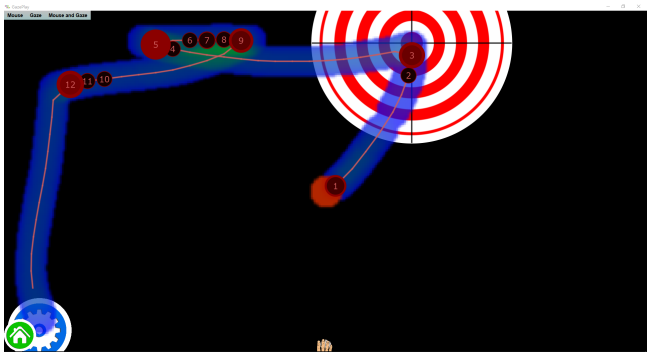

Figure 2: Modified creampie game, the line shows the trace of gaze reported by the Tobii 5 Eye Tracker

2.3 Results

	Precision pixels	Accuracy pixels	Gaze Time ms
Eye-Tracker	29	36	430
Webcam	58	112	989

Table 1 : Evaluation results

2.4 Conclusion

From table 1 we can observe that the eye-tracker overcomes the webcam in every metrics. Eye-tracker is 200% better than webcam in precision, 311% better in accuracy and 230% better in gaze time. This drop in performance of the webcam can be due to certain factors, such as the algorithm used in GazePointer application is not up-to-date, the calibration method of GazePointer application is not advanced, the built-in infra-red optical sensor does improve the performance of eye tracker, as stated in [Burton *et al.*, 2014].

3 Implementation and evaluation of additional attention layer

3.1 The choice of dataset – MPIIGaze

The MPIIGaze dataset contains 213,659 full face images and corresponding ground-truth gaze positions collected from 15 users during everyday laptop use over several months. An experience sampling approach ensured continuous gaze and head poses and realistic variation in eye appearance and illumination. To facilitate cross-dataset evaluations, 37,667 images were manually annotated with eye corners, mouth corners, and pupil centres.

3.2 The choice of app – Pytorch-mpiigaze

To train a model for MPIIGaze, the Pytorch-mpiigaze proposed by [Zhang *et al.*, 2017] is used. This is a pytorch based program aimed to train a LeNet model or a Resnet model. Figure 3 shows an overview of gaze estimation using multimodal convolutional neural networks (CNN).

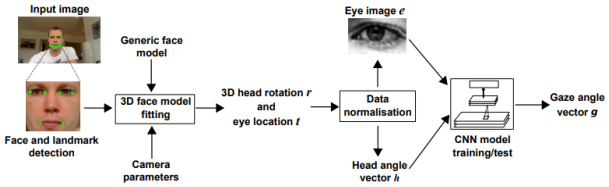

Figure 3: Overview of GazeNet– appearance-based gaze estimation using a deep convolutional neural network (CNN).

Here face detection and facial landmark detection methods proposed by [Sugano *et al.*, 2014] is used to locate landmarks in the input image obtained from the calibrated monocular RGB camera. Then a generic 3D facial shape model was fitted to estimate 3D poses of the detected faces, followed by the space normalisation technique proposed in [Sugano *et al.*, 2014] to crop and warp the head pose and eye images to the normalised training space. Afterwards a CNN was trained to learn the mapping from the head poses and eye images to gaze directions in the camera coordinate system.

Included Network - LeNet

LeNet refers to LeNet-5, which was one of the earliest convolutional neural networks and promoted the development of deep learning proposed by [Lecun *et al.*, 1998]. It possesses the basic units of convolutional neural network, such as convolutional layer, pooling layer and full connection layer, laying a foundation for the future development of convolutional neural network. The figure 4 shows the general structure of LeNet networks. In pytorch-mpiigaze program, the input is slightly modified to suit the size of the image, but the main structure (2 convolutional layer, 2 pooling layer and 2 full connection layer and its order) remains the same.

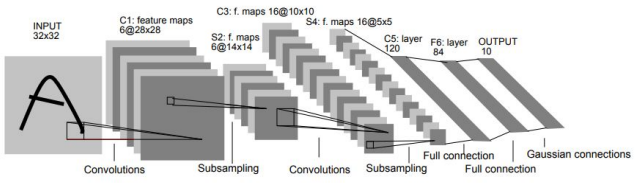

Figure 4: Architecture of LeNet-5

Included Network - ResNet

The residual neural network (ResNet) is proposed by [He *et al.*, 2015]. In the *ILSVRC competition*[4] in 2015, the ResNet network with 152 layers wins the competition, breaking the record of ImageNet, while significantly improves the depth of network. By contrast, the ILSVRC 2014 winner, GoogleNet proposed by [Szegedy *et al.*, 2014], only has 22 layers.

For a long time, deep networks are hard to train because of the notorious vanishing gradient problem — as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly, as shown in figure 5 from the paper of [He *et al.*, 2015].

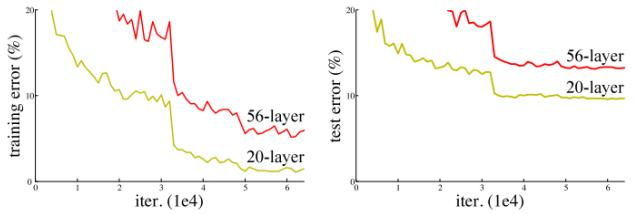

Figure 5: Error on CIFAR-10 with 20 and 56 layer networks

To solve the problem, Res-Net contains revolutionary residual blocks, as shown in the figure 6, which allow skipping one or several layers. This solution eliminates the degradation of network performance. The structure of Res-Net with the comparison of VGG model proposed by [Simonyan and Zisserman, 2014] is shown in figure 7.

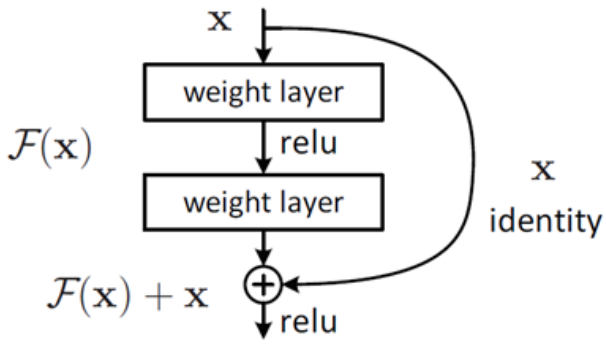

Figure 6: A residual block

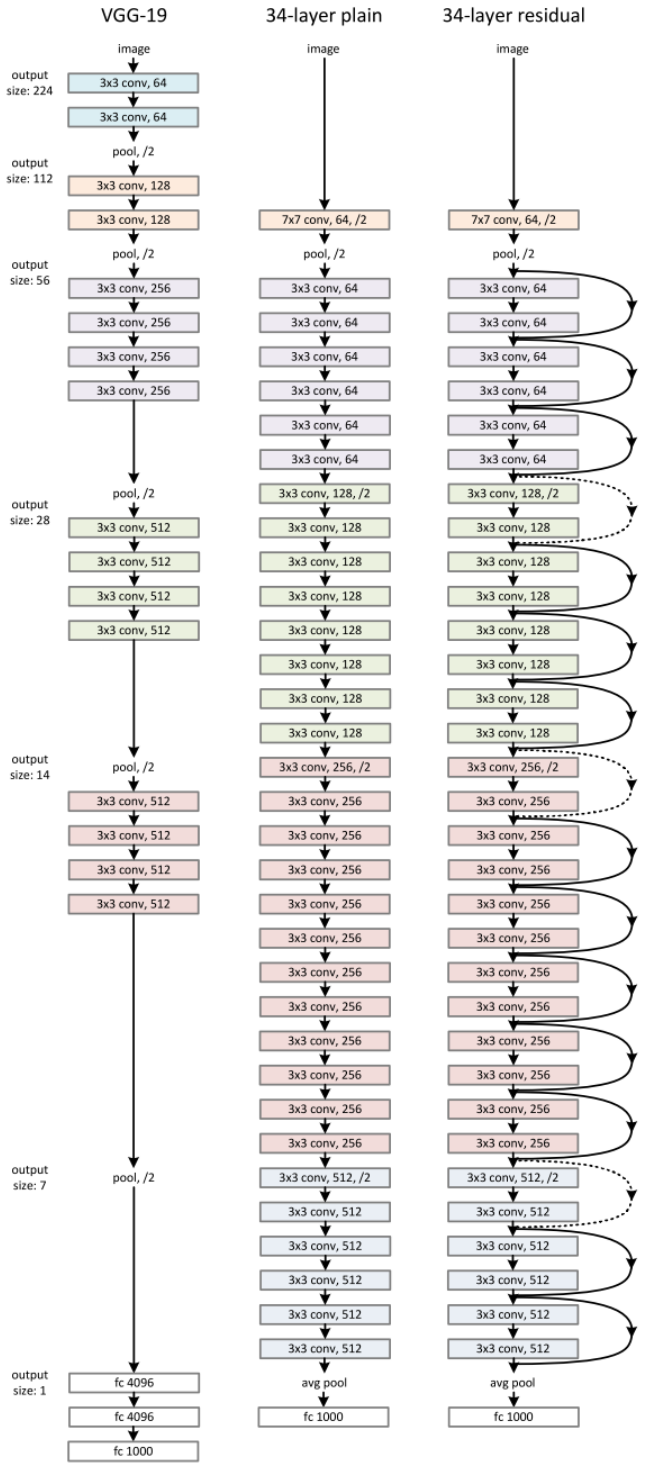

Figure 7: Left: the VGG-19 model (19.6 billion FLOPs) as a reference. Middle: a plain network with 34 parameter layers (3.6 billion FLOPs). Right: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions.

[4]https://www.image-net.org/challenges/LSVRC/

3.3 Attention in neural networks

The Attention Mechanism in neural networks is initially a resource allocation scheme that allocates computational resources to more important tasks with limited computational power, while solving the information overload problem. In neural network learning, in general, the more parameters a model has, the more expressive it is and the more information it stores, but this can lead to information overload. By introducing an attention mechanism that focuses on the information that is more critical to the task at hand and reduces attention to other information, or even filters out irrelevant information, the information overload problem can be solved and the efficiency and accuracy of task processing can be improved.

This is similar to the human visual attention mechanism, which scans the global image to obtain the target area that needs to be focused on, and then devotes more attention resources to this area to obtain more detailed information related to the target, while ignoring other irrelevant information. With this mechanism, the attention resources can be used to quickly filter out the high-value information from the large amount of information and give this part of information more weight, as shown in the figure 8 from the paper of [Xu *et al.*, 2015].

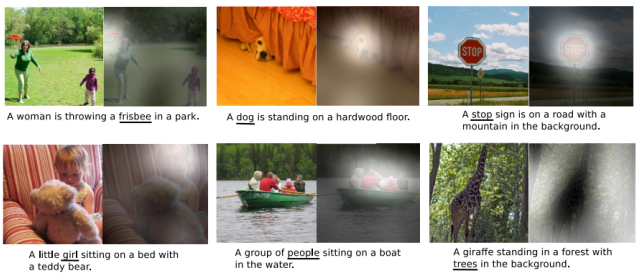

Figure 8: Examples of attending to the correct object (white indicates the attended regions, underlines indicated the corresponding word)

Nowadays attention is widely used in many aspects, such as image captioning [Xu *et al.*, 2015], image generation [Zhang *et al.*, 2018] and so on. It plays an important part in natural language processing tasks, in 2017 [Vaswani *et al.*, 2017] proposed a new transformer architecture based on attention mechanism, instead of the traditional architectures based on recurrent or convolutional layers. The new architecture have achieves better *BLEU* [5] scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost, as shown in the figure 9 from [Vaswani *et al.*, 2017].

[5] https://en.wikipedia.org/wiki/BLEU

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Figure 9: The new transformer architecture achieve better results in both BLEU scores and Training cost.

3.4 ECA-Net

ECA-Net is an extension of SENet [Hu *et al.*, 2017]. The SE-Net structure consist of 3 parts : a global average pooling for crating feature maps with size of 1*1*C, and two fully connected layer (with reduction on dimension between layers). The author of ECA-Net proves the reduction on dimension between the two fully connected layers is inefficient and unnecessary to capture dependencies across all channels [Wang *et al.*, 2019]. The paper proposed the ECA-Net, which keeps the dimension between two fully connected layers. The ECA-Net structure first obtain a 1*1*C vector by global average pooling like SENet, and then using a 1D convolution to get the channel weights between to fully connected layers. The kernel size of this 1D convolution layer is dynamically calculated. Given channel dimension C, the kernal size k is adaptively determined by :

$$k = \psi(C) = \left| \frac{\log_2(C)}{\gamma} + \frac{b}{\gamma} \right|_{odd} \tag{4}$$

Where $|t|_{odd}$ indicates the nearest odd number of t. In this paper, we set γ and b to 2 and 1 throughout all the experiments, respectively. Figure 10 shows a high-level diagram of ECA module.

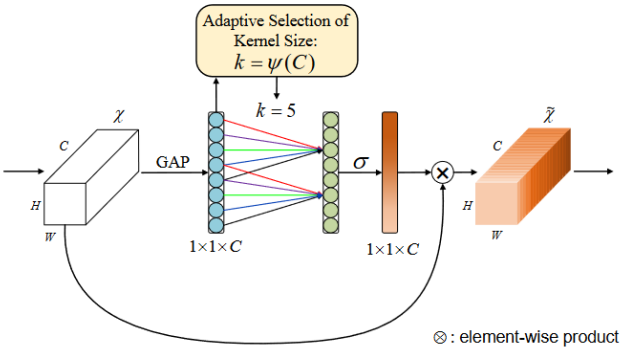

Figure 10: Diagram of efficient channel attention (ECA) module. Given the aggregated features obtained by global average pooling (GAP), ECA generates channel weights by performing a fast 1D convolution of size k, where k is adaptively determined via a mapping of channel dimension C.

Results and comparisons

Figure 11 from [Wang *et al.*, 2019] proves the ECA-Net is more efficient while performing favorably against its counterparts.

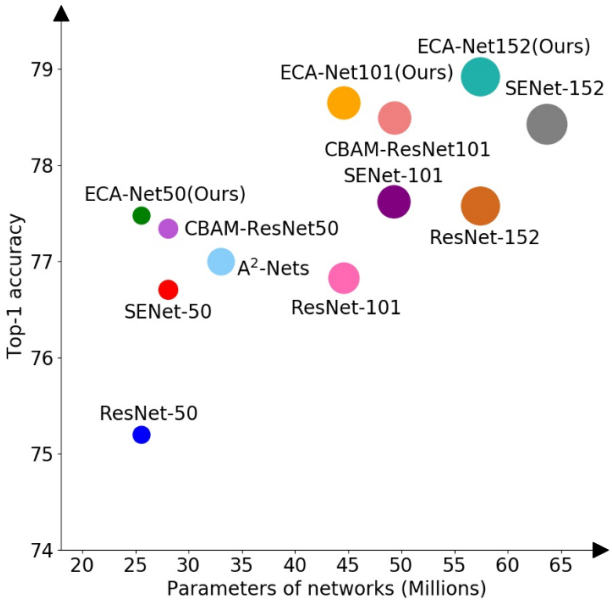

Figure 11: Comparison of various attention modules (i.e., SENet, CBAM, A^2-Nets and ECA-Net) using ResNets as backbone models in terms of classification accuracy, network parameters and FLOPs, indicated by radiuses of circles. Note that the ECA-Net obtains higher accuracy while having less model complexity.

3.5 The coordinate attention

[Hou *et al.*, 2021] propose a novel attention mechanism for mobile networks by embedding positional information into channel attention, which is called "coordinate attention". Unlike channel attention that transforms a feature tensor to a single feature vector via 2D global pooling, the coordinate attention factorizes channel attention into two 1D feature encoding processes that aggregate features along the two spatial directions, respectively. The structure of coordinate attention is shown in figure 12.

Results and comparisons

The coordinate attention inherits the advantage of channel attention methods (e.g., the Squeeze-and-Excitation attention) that model inter-channel relationships and meanwhile captures long-range dependencies with precise positional information. Figure 13 shows the performance advantage of coordinate attention over other different attention methods.

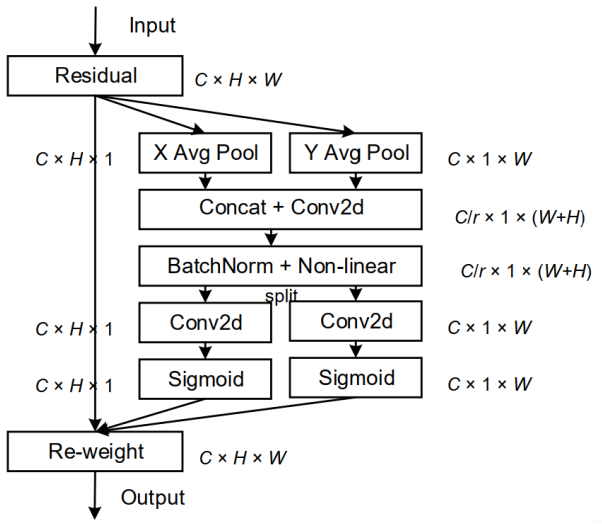

Figure 12: Structure of coordinate attention block

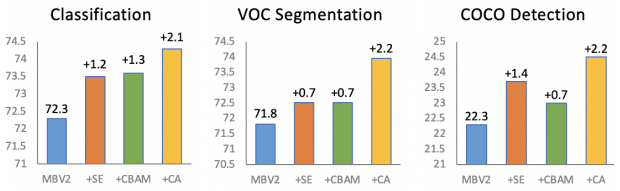

Figure 13: Performance of different attention methods on three classic vision tasks. The y-axis labels from left to right are top-1 accuracy, mean IoU, and AP, respectively. The coordinate attention approach not only achieves the best result in ImageNet classification against the SE block and CBAM but performs even better in down-stream tasks, like semantic segmentation and COCO object detection.

3.6 CBAM

[Woo *et al.*, 2018] proposed a lightweight attention module, which they name as convolutional block attention module. It's a module combining channel attention and spatial attention, and results better than the SENet model which only focus on channel attention. The figure 14 shows a high-level view of the module.

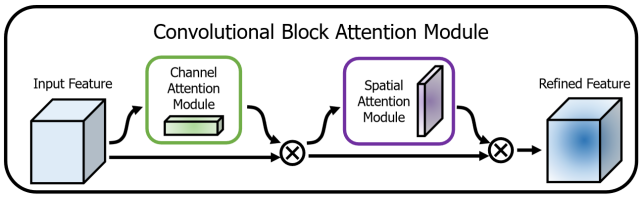

Figure 14: The overview of CBAM

The channel attention module aimed at producing a channel attention map by exploiting the inter-channel relation-

ship of features. As each channel of a feature map is considered as a feature detector, channel attention focuses on 'what' is meaningful given an input image. Figure 15 shows a overview of this sub-module.

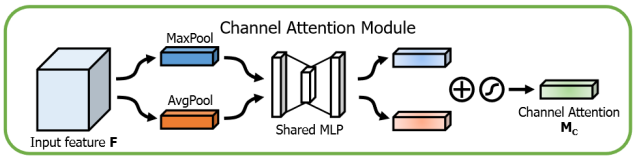

Figure 15: The channel attention sub-module

The spatial attention module generates an attention map by utilizing the inter-spatial relationship of features. Different from the channel attention, the spatial attention focuses on 'where' is an informative part, which is complementary to the channel attention, as shown in the figure 16.

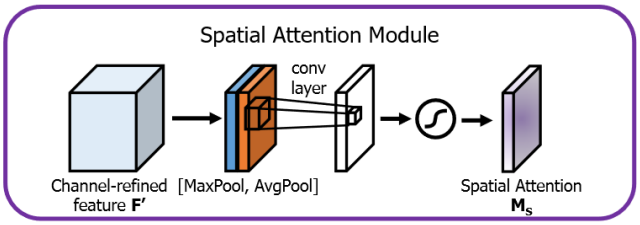

Figure 16: The spatial attention sub-module

Results and comparisons

Architecture	Param.	GFLOPs	Top-1 Error (%)	Top-5 Error (%)
ResNet18 [5]	11.69M	1.814	29.60	10.55
ResNet18 [5] + SE [28]	11.78M	1.814	29.41	10.22
ResNet18 [5] + CBAM	11.78M	1.815	**29.27**	**10.09**
ResNet34 [5]	21.80M	3.664	26.69	8.60
ResNet34 [5] + SE [28]	21.96M	3.664	26.13	8.35
ResNet34 [5] + CBAM	21.96M	3.665	**25.99**	**8.24**
ResNet50 [5]	25.56M	3.858	24.56	7.50
ResNet50 [5] + SE [28]	28.09M	3.860	23.14	6.70
ResNet50 [5] + CBAM	28.09M	3.864	**22.66**	**6.31**
ResNet101 [5]	44.55M	7.570	23.38	6.88
ResNet101 [5] + SE [28]	49.33M	7.575	22.35	6.19
ResNet101 [5] + CBAM	49.33M	7.581	**21.51**	**5.69**
WideResNet18 [6] (widen=1.5)	25.88M	3.866	26.85	8.88
WideResNet18 [6] (widen=1.5) + SE [28]	26.07M	3.867	26.21	8.47
WideResNet18 [6] (widen=1.5) + CBAM	26.08M	3.868	**26.10**	**8.43**
WideResNet18 [6] (widen=2.0)	45.62M	6.696	25.63	8.20
WideResNet18 [6] (widen=2.0) + SE [28]	45.97M	6.696	24.93	7.65
WideResNet18 [6] (widen=2.0) + CBAM	45.97M	6.697	**24.84**	**7.63**
ResNeXt50 [7] (32x4d)	25.03M	3.768	22.85	6.48
ResNeXt50 [7] (32x4d) + SE [28]	27.56M	3.771	**21.91**	6.04
ResNeXt50 [7] (32x4d) + CBAM	27.56M	3.774	21.92	**5.91**
ResNeXt101 [7] (32x4d)	44.18M	7.508	21.54	5.75
ResNeXt101 [7] (32x4d) + SE [28]	48.96M	7.512	21.17	5.66
ResNeXt101 [7] (32x4d) + CBAM	48.96M	7.519	**21.07**	**5.59**

* all results are reproduced in the PyTorch framework.

Figure 17: Classification results on ImageNet-1K. Single-crop validation errors are reported.

As shown in figure 17, the networks with CBAM outperform all the baselines significantly, demonstrating that the CBAM can generalize well on various models in the large-scale dataset. Moreover, the models with CBAM improve the accuracy upon the one of the strongest method – SENet [Hu *et al.*, 2017] which is the winning approach of the ILSVRC 2017 classification task.

3.7 The SK-Net

Selective Kernel Network is proposed by [Li *et al.*, 2019]. In standard Convolutional Neural Networks, the receptive fields of artificial neurons in each layer are designed to share the same size. It is well-known in the neuroscience community that the receptive field size of visual cortical neurons are modulated by the stimulus, which has been rarely considered in constructing CNNs. The paper proposed a dynamic selection mechanism in CNNs that allows each neuron to adaptively adjust its receptive field size based on multiple scales of input information, and in the end fuse all branches together by softmax attention that is guided by the information in these branches. The figure 18 shows the steps of SK-Net.

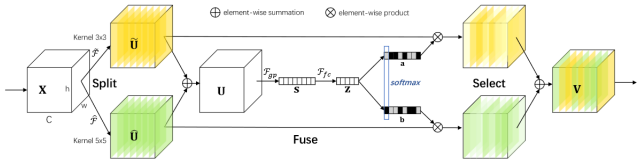

Figure 18: The architecture of selective kernel convolution

The SK-Net is composed by 3 subsections :

- Split, which convolves the original image by different convolution kernels;
- Fuse, which uses gates to control the information flows from multiple branches carrying different scales of information into neurons in the next layer;
- Select, which apply soft attention across channels, in order to adaptively select different spatial scales of information.

Results and comparisons

As shown in figure 19, SKNet improves the performance of the latest 2019-based CNNs. Notably, SKNet-50 outperforms ResNeXt-101 by more than 0.32%, despite ResNeXt-101 being 60% larger in terms of parameters and 80% larger in terms of computation. Compared to InceptionNets, which are of comparable or lower complexity, SKNets have an absolute performance improvement of more than 1.5%, demonstrating the superiority of adaptive aggregation for multiple cores. Using fewer parameters, SKNets can achieve a 0.3-0.4% improvement over network-like SENets in the 224×224 and 320×320 evaluations.

3.8 Implementation of attention layer

On LeNet

The attention layer is implemented between the last pooling layer and the first fully connected layer.

	top-1 err (%)		#P	GFLOPs
	224×	320×		
ResNeXt-50	22.23	21.05	25.0M	4.24
AttentionNeXt-56 [44]	21.76	–	31.9M	6.32
InceptionV3 [43]	–	21.20	27.1M	5.73
ResNeXt-50 + BAM [32]	21.70	20.15	25.4M	4.31
ResNeXt-50 + CBAM [45]	21.40	20.38	27.7M	4.25
SENet-50 [12]	21.12	19.71	27.7M	4.25
SKNet-50 (ours)	**20.79**	**19.32**	27.5M	4.47
ResNeXt-101	21.11	19.86	44.3M	7.99
Attention-92 [44]	–	19.50	51.3M	10.43
DPN-92 [5]	20.70	19.30	37.7M	6.50
DPN-98 [5]	20.20	18.90	61.6M	11.70
InceptionV4 [41]	–	20.00	42.0M	12.31
Inception-ResNetV2 [41]	–	19.90	55.0M	13.22
ResNeXt-101 + BAM [32]	20.67	19.15	44.6M	8.05
ResNeXt-101 + CBAM [45]	20.60	19.42	49.2M	8.00
SENet-101 [12]	20.58	18.61	49.2M	8.00
SKNet-101 (ours)	**20.19**	**18.40**	48.9M	8.46

Figure 19: Comparisons to the state-of-the-arts under roughly identical complexity. 224× denotes the single 224×224 crop for evaluation, and likewise 320×. Note that SENets/SKNets are all based on the corresponding ResNeXt backbones

On ResNet

The attention layer is implemented in the end of each ResNet basic block. Given that there's a total of 3 basic blocks in the ResNet of pytorch-mpiigaze, a total of 3 attention layer is added.

3.9 Experiments

Equipment for evaluation

All the training were done with a notebook version of GeForce GTX 1070 with CUDA availability. During the training the graphic card reached the power limit of 110 watts, with a usage of about 75%.

Evaluation procedure

The pytorch-mpiigaze proposed several scripts for training and verifying the original LeNet and ResNet networks. With some modifications to the yaml configuration files, it's able to do the same training and verification on the networks with attentions. The MPIIGaze dataset contains the image of 15 participants, by using images of 14 participants and leave 1 for evaluation, we can get 15 different error data.Every model is trained for 40 epochs because the training loss and validation loss converges and also the loss function is minimum at that time [Bandyopadhyay *et al.*, 2021b].

3.10 Results

Figure 20 and 21 demonstrates the gaze angle error the network returned for each participants. By dividing the angle error of the network with attention layer into the network without attention layer and the calculate the average of them, we have obtained the relative performance of the new network, as shown in figure 22 and 23.

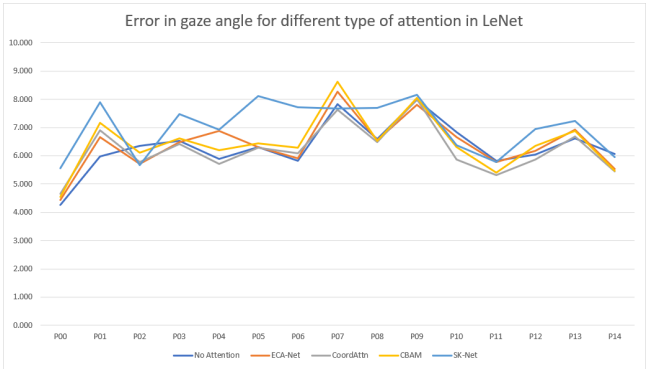

Figure 20: Error in gaze angle for different type of attention in LeNet

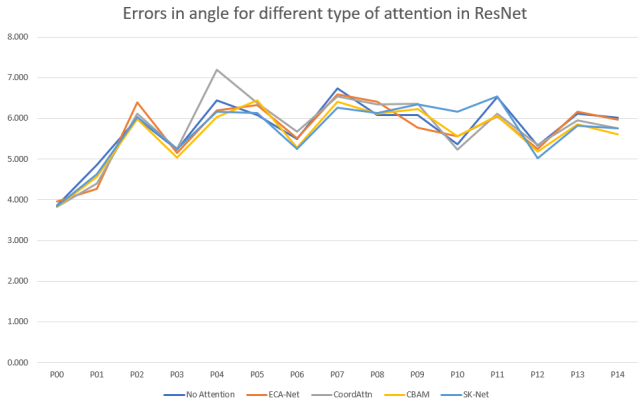

Figure 21: Error in gaze angle for different type of attention in ResNet

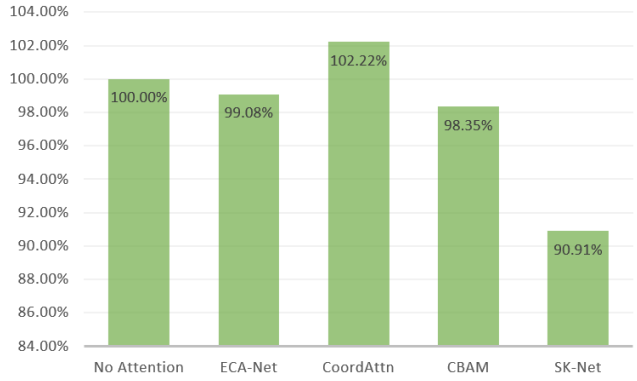

Figure 22: Relative performance in angle error of different LeNet networks. LeNet with coordinate attention layer is 2.2% better overall than the default no attention LeNet network.

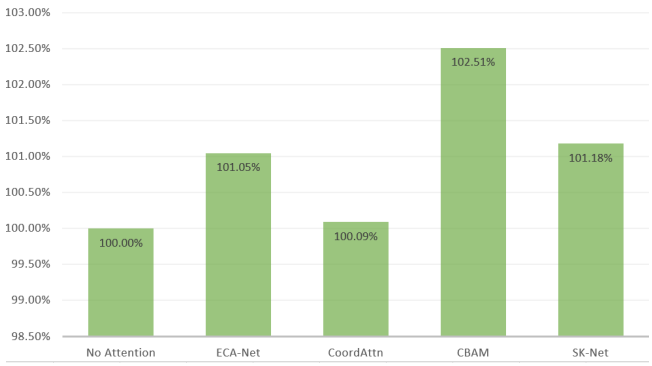

Figure 23: Relative performance in angle error of different ResNet networks. The ResNet with CBAM layer is 2.5% better overall than the default no attention ResNet network.

The time needed for each epoch is also a very important performance metric. The following figure 24 and 25 show the comparisons of the time for each epoch.

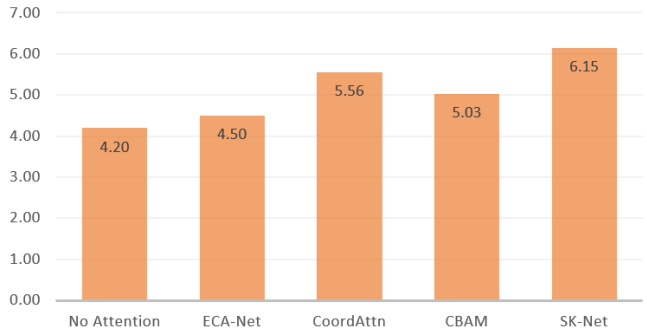

Figure 24: Time for finishing each epoch of different LeNet networks.

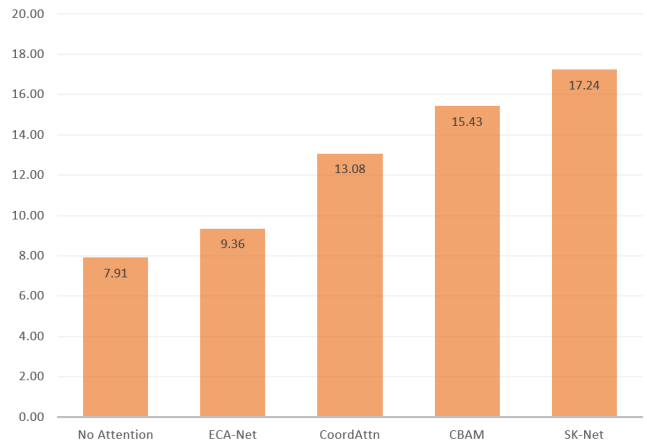

Figure 25: Time for finishing each epoch of different LeNet networks.

3.11 Conclusion

Result analysis

The additional attention layer does effect the performance of the neural network. Adding coordinate attention to LeNet will improve the overall performance in angle error by 2%, and the ECA-Net, CBAM and SK-Net has negative impact on the network. All the attention layer increased the training time comparing to the ResNet without attention, from 107.14% of ECA-Net to 146.43% of SK-Net.

In ResNet, all the additional attention layer does effect the performance positively. The most significant one is the CBAM, which is 2.5% more accurate than ResNet without attention. However all the attention layer significantly increased the training time comparing to the ResNet without attention, from 118.33% of ECA-Net to 217.95% of SK-Net.

The angle error returned by the network with attentions is sometimes worse than the network without attention. Even though some network with attention layer returns a 2% better result, they all significantly increased the time consumed for each epoch. This could happen due to the fact that the attention models are mainly based on ImageNet 2012 dataset [Russakovsky *et al.*, 2014].With an average resolution of 469x387 pixels, it is significantly smaller than the 640x480 images captured by webcam in MPIIGaze dataset.

Future works

One possible future works is improving the performance of attention layer by fine tuning its parameters. The attention models are mainly designed and specially tuned for ImageNet 2012 dataset [Russakovsky *et al.*, 2014]. With an average resolution of 469x387 pixels, it is significantly smaller than the 640x480 images captured by webcam in MPIIGaze dataset. Due to the unique MPIIGaze dataset, the structure and parameter of attention layer (and the network) need to be adjusted to better suits the input of the dataset.

Another possible improvement is to combine the attention layer with calibration. [Bandyopadhyay *et al.*, 2021a] justified that using small amounts of test data in the training set can significantly increase the accuracy of the model. This introduction of small amount of test data should allows the attention layer to focus on the more important part of the subset used for test and hence produce a more significant improvement.

References

[Bandyopadhyay *et al.*, 2021a] Nairit Bandyopadhyay, Sébastien Riou, and Didier Schwab. Effect of personalized calibration on gaze estimation using deep-learning. *CoRR*, abs/2109.12801, 2021.

[Bandyopadhyay *et al.*, 2021b] Nairit Bandyopadhyay, Sébastien Riou, and Didier Schwab. Webcam as alternate option for eye-trackers in gaze gaming software: Gazeplay. 2021.

[Burton *et al.*, 2014] Liz Burton, William Albert, and Mark Flynn. A comparison of the performance of webcam vs. infrared eye tracking technology. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 58(1):1437–1441, 2014.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[Hou *et al.*, 2021] Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. *CoRR*, abs/2103.02907, 2021.

[Hu *et al.*, 2017] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

[Lecun *et al.*, 1998] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Li *et al.*, 2019] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. *CoRR*, abs/1903.06586, 2019.

[Russakovsky *et al.*, 2014] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[Skovsgaard *et al.*, 2011] Henrik Skovsgaard, Javier San Agustin, Sune Alstrup Johansen, John Paulin Hansen, and Martin Tall. Evaluation of a remote webcam-based eye tracker. In *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications*, NGCA '11, New York, NY, USA, 2011. Association for Computing Machinery.

[Sugano *et al.*, 2014] Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. Learning-by-synthesis for appearance-based 3d gaze estimation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1821–1828, 2014.

[Szegedy *et al.*, 2014] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[Tobii, 2011] Tobii. *Accuracy and precision test method for remote eye trackers*. Stockholm, Sweden, 2011.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[Wang *et al.*, 2019] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. *CoRR*, abs/1910.03151, 2019.

[Woo *et al.*, 2018] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. *CoRR*, abs/1807.06521, 2018.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention, 2015.

[Zhang *et al.*, 2015] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. Appearance-based gaze estimation in the wild. *CoRR*, abs/1504.02863, 2015.

[Zhang *et al.*, 2017] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. *CoRR*, abs/1711.09017, 2017.

[Zhang *et al.*, 2018] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2018.

[Zhao *et al.*, 2018] Lei Zhao, Zengcai Wang, Guoxin Zhang, Yazhou Qi, and Xiaojin Wang. Eye state recognition based on deep integrated neural network and transfer learning. *Multimedia Tools and Applications*, 77(15):19415–19438, Aug 2018.