EasyGD: Gaze Detection on Wireless Mobile Systems
CS386W Wireless Networking : Course Project Final Report

Yuchen Cui
University of Texas at Austin
Austin, TX, USA
yuchencui@utexas.edu

Yan Pei
University of Texas at Austin
Austin, TX, USA
ypei@cs.utexas.edu

ABSTRACT

Accurate gaze detection explores the opportunity that machines can identify human’s interest and intent, and therefore leverage such information to improve human’s experience during human-computer interaction. Existing works on gaze detection usually require specialized wearable devices with bulky sensors, which is inconvenient for users. Current device-free gaze detection approaches are often computationally intensive and cannot be easily deployed on average mobile devices, of which both the computational and energy resources are limited.

In this paper, we propose EasyGD, a light weight real-time gaze detection mobile system. EasyGD takes the image stream produced by the embedded (front) camera of a mobile device and provides gaze detection services using a pre-trained deep neural network model. We implemented EasyGD as an android application. The evaluation shows that EasyGD can produce acceptable real-time gaze detection on an average mobile device. By using simple approximate methods, the power consumption can be further reduced by 20% without user experience degradation.

KEYWORDS

Gaze detection, android application, mobile systems, deep neural network, energy saving

1 INTRODUCTION

Human-computer interaction (HCI) is a research area that emerged in early 1980s, initially as a specialty area in computer science embracing cognitive science and human factors engineering. HCI has experienced rapid and steady growth for the last three decades, as computers are more and more accessible for average users. Not only does HCI provide better communication between users and machines, but also open up a great field of research.
in this project, we developed and integrated a gaze detection system that is deployable on a mobile device and uses only images of human faces captured by its embedded front camera as input. Our contributions can be summarized as follows.

- We propose EasyGD, a light weight real-time gaze detection system that produces acceptable gaze detection on an average mobile device.
- We employed a state-of-the-art image classification network that is appropriate for mobile applications and easily transferrable among devices.
- The system is well evaluated in terms of speed, accuracy and energy efficiency. A simple energy saving strategy can further reduce power consumption by 20%.

The rest of this paper is organized as follows. Section 2 provides an overview of related work on gaze detection. Section 3 introduces the choices and approaches during the design of EasyGD. Section 4 describes implementation details. Section 5 shows both quantitative results and our qualitative analysis. Section 6 discusses the limitation and possible extension of our work. And finally, section 7 concludes and outlines our possible future work.

2 RELATED WORK

Eye-gaze tracking is one of the most challenging problems in the research area of computer vision. Gaze prediction can be done by either analyzing the content of the image with saliency models [6] or directly monitoring human eyes. Content-based saliency models are limited to certain class of applications since they assume there exist universally salient objects or areas in an image or stream of images (videos). On the other hand, directly monitoring human eyes makes no assumption on the content of an image and can be used for communication purpose such as using gaze fixation to type. However, such kind of monitoring is often restricted by the displaying medium, i.e. the hardware used for displaying and monitoring. The focus of this project is on the latter problem, where a mobile device is used as the displaying medium and gazes are predicted by directly monitoring human faces using the embedded front camera. Leveraging a common platform will make this technology more accessible to average users.

Many approaches have been proposed to predict gaze by directly monitoring human eyes/faces and they can be divided into two major categories: model-based and appearance-based [7]. Model-based approaches adopt a geometric model of human eyes, which relies on either an external light source to detect desired features [26] [9] or detections of iris edges and pupil centers [11] [3]. These approaches require specialized cameras, suffer from the inherent accuracy of the geometric models and are not robust to various lighting conditions or low resolution images. On the other hand, appearance-based methods [25][20][14] directly use raw images of eyes or faces as input, which are expected to be more robust to low resolution images or changes in lighting conditions, but often require a large amount of paired training data to learn robust mapping from images to gaze fixations. Krafka et al. took a very similar approach as ours by using a deep network for gaze prediction but they collected 1000× more data than we did, which was necessary for training the deep model from scratch and allowed them to achieve higher accuracy. However, by fine-tuning an existing network on a common deep learning platform, the same system or approach is readily applicable to any device with support for the very platform (Tensorflow [1] in our case).

Gaze tracking systems have also been studied under many different use cases. The work of Kelton et al. [13] utilizes gaze fixations to determine where on the webpage needs to be rendered first in order to provide a smooth browsing experience. Zhang et al.[27] have also studied how gaze can be leveraged for attention-guided learning in Atari game playing.

3 APPROACH

The overview of EasyGD is shown in Fig. 1. Our system consists of a data collection system and an image classification system implemented as a deep Convolutional Neural Network (CNN).

The image classification system takes a raw image as input, and produce a classification of the current gaze region. CNNs have the state-of-the-art performance on large-scale image classification tasks. Therefore, we decided to employ a deep convolutional network and cast gaze detection as an image classification task.

Data collection is required to train image classification systems. Since the front camera position varies from device to device, data captured using a different model of cell phone

Date: April 2018.
may not be directly applicable to our target system. Krafka et al. [14] released a dataset containing about 2 million frames of images with gaze labels, captured on iPhone 6 devices. Training a classifier on their dataset first (and then fine-tune on our own) would allow a neural network to extract features specific to the domain of gaze detection. However, due to practical reasons (limited time and resources), we could not leverage their dataset for our project. Therefore, we collected a dataset of our own with a reasonable size for training and testing different models.

Given the span of this project, it was hard to collect a dataset with a desirable size to train a deep network from scratch, while training state-of-the-art deep networks from scratch by itself takes a prohibitive amount of time. Hence, we decided to use existing CNN architectures with pre-trained weights on the ImageNet [5] dataset and fine-tune on the small dataset we collected.

Figure 2: Top-1 Accuracy vs. Number of Operations and Number of Parameters for different CNN architectures, obtained from the work of Canziani et al. [2]

One of our major design decisions is to pick a CNN architecture to use. In order to make our system deployable on an average commercial mobile device for practical use, the network needs to be relatively small in size and with reasonable processing speed for real-time applications. Canziani et al. [2] conducted a detailed study of a variety of well-known CNN architectures and compared their performance in terms of prediction accuracy, number of operations per inference and total size of the network, benchmarked on the ImageNet dataset. Fig. 2 shows one of the major results presented in their work.

For a given CNN architecture, number of operations measures how much computation is required per inference and is directly related to how much energy the model will consume at runtime. The size of the network or the number of parameters indicates how much memory is needed to use this model. Therefore, when targeting on mobile applications, we need to minimize both the size and number of operations of a model while not sacrificing the accuracy by too much. We decided to use Inception-V3 [24] architecture in our system, based on its relatively high accuracy and that it is of a small size in terms of both number of parameters and number of operations comparing to other networks with higher accuracy. Fig. 3 shows a diagram of the Inception-V3 network architecture.

Since the performance of neural networks varies from dataset to dataset, as a comparison (while we believed that Inception-V3 is the best model to be used in our system), we also evaluated the performance of Inception-V4 [23], ResNet-50 [8] and VGG-19 [22] architectures on our dataset to see if we can gain any performance gain using these alternatives.

In terms of the classifier, we first formulated the problem as a 9-way classification problem by dividing the display screen into a 3 × 3 grid. However, we later found out that it is very hard to predict gaze at this level of granularity using raw images based on our training set size. The accuracy achieved by Krafka et al. [14] with a hand-crafted neural network and trained on about 1.5 million images was around 2 cm on an iPhone, which is not accurate enough to do the 9-way classification problem we defined. Therefore, we decided to solve a 4-way classification problem instead, by dividing the display screen into 2 × 2 areas: up-left, up-right, down-left, and down-right. Figure 4 illustrates the change in granularity of our classification problem.

4 IMPLEMENTATION

Following the design choices described in Section 3, we implemented EasyGD as an Android application on Motorola XT1095 and Pixel 2. The implementation details of EasyGD can be separated into four parts: data collection, preprocessing, model training and application design.

4.1 Data Collection

To collect a set of training data, we developed another Android application and deployed it on Motorola XT1095 phone running Android 5.0. A red dot indicating the desired gaze point for training shows up at a random location on the screen and the subject is asked to take a photo of themselves after they move their gaze onto the dot. Fig. 5 shows the application interface. The blue background displays the front camera image at run time.

Data collection was done in casual settings. 62 subjects participated in our data collection process and a total of 1726 images were taken. However, we realized that the dataset failed to cover a variety of subjects from different ethnic background or balancing the gender and age of subjects. The majority of the subjects were Asian male students at their 20s.

Date: April 2018.
4.2 Preprocessing

The quality of the picture in the dataset we collected varies from subject to subject. Some pictures failed to capture the whole face of the subject, some were taken when the subject closed their eyes, and some subjects had exaggerating facial expressions that can potentially have features picked up by the CNN in an undesired manner. Therefore, it was necessary to preprocess the dataset before using it for training deep models.

We applied three steps to preprocess our data. 1) We first removed undesirable pictures by visually examining if the subjects’ eyes were closed or the facial expression of a subject was exaggerated. 2) We then ran a face detection algorithm on all the remaining images to filter out the pictures that failed to capture a face recognizable according to the algorithm. The reason for this step is that image background also varies much due to distance and environment difference when taking photos. Due to the limitation of our training.
set size, interference introduced by background and face integrity should be eliminated for the sake of training accuracy. The effect of face recognition is shown in Fig. 7. Lastly, we observed that samples at the boundary of two classes are visually similar, as displayed in the left part of Fig. 6. The distance between these sample points may not be able to represent the difference between different gaze regions. Thus these data points may not present learnable differences in the feature space of the network. As a result, we created an artificial band between classes and stripped out sample data lying inside these bands. The right part of Fig. 6 shows the final training set we used in the model training process.

By applying these three preprocessing steps, our model accuracy is improved by 20%.

4.3 Model Training

The user interface of our android application is given in Fig. 9. The top blue part displays current gaze regions with their probability, mainly for debugging reason. The screen is divided into a $2 \times 2$ grid with grey outlines. Note that the region with the highest probability will be covered with a semi-transparent white layer. This white layer helps the user learn the current gaze prediction without looking into the top blue display, since user’s gaze is usually at somewhere else other than the top area.

The last part of this interface is the demo toggle button which was originally designed for class live demo. In the demo mode, instead of using white layers, we use commercial products as gaze region covers, in order to imitate a shopping website. (We didn’t show the live demo during the presentation but we shown it to Dr. Qiu).

Date: April 2018.
5 EVALUATION

5.1 Quantitative Results
The best performance of different models on the validation dataset are shown in table 1. The reported accuracies are all for training only the fully connected layer of different architectures since training all the layers had lower accuracy rates. Inception-V3 performed best on the validation/test dataset while ResNet_50 performed best on the training dataset.

5.2 Qualitative Results
The obtained best-performing model was deployed on the Moto X phone so that we could evaluate its performance qualitatively. Fig. 10 and 11 show some selected gaze detection results. Both subjects have images taken in the training dataset.

The system seems to perform well on known subjects with an acceptable success rate. For example, when the author made some exaggerated eye ball movement (the gaze point may be out of the screen), EasyGD can work pretty well, as shown in Fig. 10. However, Fig. 11 indicate that for small eye movement, especially horizontally, there is much uncertainty in EasyGD’s decision.

From interactions with the model we observed that the EasyGD is able to better distinguish gazes looking up or down than those looking left or right. We suspect reasons for this behavior include that 1) the 4-way classification problem allowed more error in the vertical direction than in the horizontal direction because of the shape of the screen, and 2) the motion of looking up or down often involves moving eye lids, which incurs relatively large changes in both the size and shape features of eyes so that they become more easily distinguishable.

5.3 Energy Consumption
In order to evaluate the energy consumption of the system, EasyGD was also deployed on a Pixel 2 phone running Android 8.0, which has better computational power than the Moto X. The inference time for processing a single image is around 950ms on Moto X, while it is only about 350ms on Pixel 2. Fig. 12 shows the test interface with debug information of inference time as displayed on the bottom left side of the screen.

Table 1: Accuracy of the Best-Performing Model of Different Network Architectures

<table>
<thead>
<tr>
<th></th>
<th>Inception-V3</th>
<th>Inception-V4</th>
<th>ResNet</th>
<th>VGG19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>44%</td>
<td>35%</td>
<td>48%</td>
<td>26%</td>
</tr>
<tr>
<td>Test</td>
<td>45%</td>
<td>30%</td>
<td>37%</td>
<td>26%</td>
</tr>
</tbody>
</table>
We recorded the battery level before and after running the App for 20 minutes on each of the two cell-phones and found out that running on Moto X consumed 6% of its battery and running on Pixel 2 consumed 18% of its battery after 20 minutes. Since the system runs faster on the Pixel 2 phone, the battery usage is expected to be higher after running for the same amount of time since more computation has been done. For applications that do not require gaze detection at a high frequency, the energy consumption can be reduced by lowering the frame rate. By simply accepting every other inference request, we were able to reduce the energy consumption of the system by 20% on the Pixel 2 phone.

6 DISCUSSION
6.1 Limitations
We have implemented a proof-of-concept system for cheap gaze detection on wireless mobile systems by fine-tuning a state-of-the-art image classification system. With a limited amount of dataset, we were only able to achieve an accuracy of 45% at its best. We have also observed that it is important to control the data collection process or perform preprocessing to obtain clean training data (to reduce noise ratio). We believe that a dataset that covers a variety of head poses, lighting conditions, and backgrounds will improve the classification accuracy. A better data collection technique is needed for such purpose.

From our limited observation, our system’s performance seems limited to the device that was used to collect the dataset and limited to the known subjects. However, an extensive test on various different platforms is needed to see how much of the model is transferrable.

6.2 Future Work
As discussed previously, it is important to have a better scheme for collecting training data for our models. One option could be training another image classifier to automatically identify successful and unsuccessful samples using currently available datasets. Expert knowledge from human-computer interaction and cognitive science can also be borrowed to design better user interfaces. For example, instead of showing boring dots, artificially designed attention-grabbing images or videos can be used for motivating users to look at areas as we desired during data collection processes.

While we believe deep CNNs can learn translation-invariant and rotation-invariant features, the observation point, i.e. the position of the camera, can influence the prediction of gaze. It is important to test if different devices with cameras positioned differently can share the same model. If the learned model is somehow not invariant to the camera position, applying CapsuleNets [19] may alleviate such dependancy on hardwares and is an interesting area of future work.

7 CONCLUSION
Gaze detection, which reflects human’s interest or intent, opens up a new research area for better human-computer interaction, and many more commercial applications. Existing approaches either require bulky and expensive specialized devices, which is inconvenient and inaccessible for average users, or expensive in computation and large in size, which is in applicable to mobile devices.

In this paper, we propose EasyGD, a light weight real-time gaze detection mobile system. EasyGD can provide acceptable gaze detection quality on an average mobile device. Implemented as an Android application, it takes image stream produced by the embedded front camera of a phone as input and output the predicted region where the user’s gaze is. The core module of EasyGD is a CNN-based image classifier using Inception-V3 architecture. The model accuracy was improved by 20% when preprocessing techniques, such as face recognition and increasing sample distance, are applied. By using simple energy saving methods, the power consumption can be further reduced by 20% without user experience degradation.
ACKNOWLEDGMENTS
This work is based on the Tensorflow open-source library. The authors would like to thank all the Tensorflow developers. The dataset used in this project was collected from friends and colleagues of the authors. The authors would like to send their sincere gratitude to Wenguang Mao, Shanshan Wu, Zhifeng Jing, Ryan Zhang, Jingyu Liao, Jeremy Glass, Barnabas, Haoran Yang, Peng Liu, Joseph Kim, Michael He, Yi-shan Lu, Gurbinder Gill, Roshan Dathathri, Swarnendu Biswas and many other friends for their generous contribution to this project.

REFERENCES


