Near-Eye Display Gaze Tracking via Convolutional Neural Networks

Robert Konrad
rkkonrad@stanford.edu

Shikhar Shrestha
shikhars@stanford.edu

Paroma Varma
paroma@stanford.edu

Abstract

Virtual reality and augmented reality systems are currently entering the market and attempt to mimic as many naturally occurring stimuli in order to give a sense of immersion. Many aspects such as orientation and positional tracking have already been reliably implemented, but an important missing piece is eye tracking. VR and AR systems equipped with eye tracking could provide a more natural interface with the virtual environment, as well as opening the possibility for foveated rendering and gaze-contingent focus. In this work, we approach an eye tracking solution specifically designed for near-eye displays via convolutional neural networks, that is robust against lighting changes and occlusions that might be introduced when placing a camera inside of a near-eye display. We create a new dense eye tracking dataset specifically intended for neural networks to train on. We present the dataset as well as report initial results using this method.

1. Introduction

Immersive visual and experiential computing systems are entering the consumer market and have the potential to profoundly impact our society. Applications of these systems range from entertainment, education, collaborative work, simulation and training to telesurgery, phobia treatment, and basic vision research. In every immersive experience, the primary interface between the user and the digital world is the near-eye display. Thus, developing near-eye display systems that provide a high-quality user experience is of the utmost importance. Many characteristics of near-eye displays that define the quality of an experience, such as resolution, refresh rate, contrast, field of view, orientation and positional tracking, have been significantly improved over the last years. However, an important missing interface is gaze tracking.

Although the area of gaze tracking has been studied for decades, they have been studied in the context of tracking a user's gaze on a screen placed a distance away from the eyes. Many such techniques are not suitable for near-eye displays where the imaging of the eye is restricted to a small area, and might be subject to occlusions and only partial views of the pupil and iris. Currently, one company, SMI, offers to hack a an Oculus DK2 for a large cost (roughly ten thousand dollars). We seek a more elegant and cost effective solution for providing the benefits of gaze tracking for near eye displays.

Gaze tracking not only provides a new intuitive interface to near-eye displays, but also allows for important techniques increasing immersion and comfort in foveated rendering and gaze contingent focus.

Applications Depth of field, the range of distances around the plane of focus that still appear sharp, is a cue that humans use to detect depth of objects in a scene. Given a user's gaze position on the screen, a depth of field can be rendered into a scene, creating an adaptive depth of field system. As the gaze moves around the scene, different objects come into focus and other objects fall out of focus, creating a more realistic experience than an all-in-focus image that current VR and AR displays present. Such as system has been shown to reduce image fusion times in stereo system in [8], as well as increasing subjective viewing experiences [4, 9]. In a first person environment, rendering a depth of field effect showed an increased sense of immersion [5]. Because the area of real-time depth of field rendering is well studied [3, 18, 17], an adaptive depth of focus system could be immediately created with the implementation of a robust gaze tracking system suitable for near eye displays.

A second potential use of gaze tracking in near-eye displays is gaze contingent focus, which improves comfort of VR and AR displays by reducing the vergence-accommodation conflict. When we look at objects our eyes perform two tasks simultaneously when looking at an object: vergence and accommodation. Vergence refers to the oculumotor cue created by the muscles in our eyes rotating our eyeballs so that they verge to the point we look at. Accommodation is the oculomotor cue created by the muscles in our eyes bending the lenses in our eyes such that the object of interest comes into focus. In normal, real-world, conditions our eyes verge and accommodate to the same distance. However, in near-eye stereoscopic display, the user the user is able to verge at any distance in the

scene but is forced to focus to a fixed distance. This distance is a function of the distance between the lenses and the display in the head-mounted display, as well as the focal length of the lenses themselves. This forces a mismatch between vergence (able to verge anywhere) and accommodation (only able to accommodate to one distance), known as the vergence accommodation conflict. When exposed to such a conflict for extended periods of time, users develop symptoms of headache, eye strain, and, in extreme cases, nausea[13].A system capable of determining the distance to which the user is verged at, through gaze tracking, can either use focus-tunable optics or an actuated display to reduce the vergence accommodation conflict by changing the distance to which users focus to, as explained in [7].

2. Related Work

A variety of remote eye gaze tracking (REGT) algorithms have been reported in literature over the past couple of decades. For our purposes, the general body of knowledge can be divided into two categories: ones which assume a model of the eye and ones which learns an eye tracking model.

2.1. Model Assumed REGT

Methods assuming a model of the eye generally extract features from an image of the eye and map them to a point on the display. This type of work generally either uses intensity images captured from a traditional camera as seen in [19], or uses illumination from an infrared light source and captures the eye with an IR camera.

Infrared-Based REGT IR illumination creates two effects on the eye. Firstly, it creates the so-called bright-eye effect, similar to red-eye in photography, which results from the light "lecting off of the retina. The second effect, a glint on the surface of the eye, is caused by light reflecting off the corneal surface, creating a small bright point in the image. This glint is often used as a reference point, because if we assume that the eye is spherical, it does not move as the eye rotates in its socket.

After grabbing an image of the eye, the glint and pupil an be extracted via image processing algorithms described in [6]. A glint pupil vector can be calculated, and mapped to a 2-D position on the screen via some mapping function. Although many have been proposed, [20, 19], the most commonly used function is the 2nd order polynomial defined in [10], defined as :

$$s_x = a_0 + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2$$
$$s_y = b_0 + b_1x + b_2y + b_3xy + b_4x^2 + b_5y^2$$

	Year	# subjects	# targets	# head poses	Calibration	Resolution	Dataset size
UUlm	2007	20	2–9	19	Yes	1600 × 1200	2,200 imgs.
HPEG	2009	10	Continuous	2	Yes	640 × 480	20 videos (~6.6 k imgs.)
Gi4E	2012	103	12	1	No	800 × 600	1,236 imgs.
CAVE	2013	56	21	5	Yes	5184 × 3456	5,880 imgs.
CVC	2013	12	12–15	4	Yes	1280 × 720	48 videos (~20 k imgs.)
EYEDIAP	2014	16	Continuous	Continuous	Yes	1920 × 1080	94 videos
Multiview	2014	50	160	8 (+synthesized)	Yes	1280 × 1024	64,000 imgs. (+synth.)
MPIIGaze	2015	15	Continuous	Continuous	No	1280 × 720	213,659 imgs.
OMEG	2015	50	10	Continuous	No	1280 × 1024	44,827 imgs.
TabletGaze	2015	51	35	Continuous	No	1280 × 720	816 videos (~120 k imgs.)

Figure 1. Table of existing gaze tracking datasets, which are mostly tailored towards model assumed methods.

where (s_x, s_y) are the screen coordinates, and (x, y) are the pupil-glint vector components. A calibration procedure is performed to estimate the unknown variables $a_0, a_1, ...b_5$ via least squares analysis by asking a user to looking at (at least) 9 calibration targets. The accuracy of the best IR-based methods fall somewhere between 1°to 1.5°accuracy.

View-Based REGT In view-based REGT, only intensity images from traditional cameras are used without any additional hardware. These techniques rely more on image processing techniques to extract features from the eyes directly, which can then be mapped to 2D pixel coordinates. Tan et. al [16] uses an image as a point in high dimensional space and through an appearance-manifold technique is able to achieve a reported accuracy of 0.38°. Zhu and Yang [19] proposed a method for feature extraction from intensity images and using a linear mapping function are able to achieve a reported accuracy of 1.4°.

2.2. Model Learned REGT

Model learned REGT techniques use some sort of machine learning algorithm to learn an eye tracking model from training data consisting of input/output pairs, i.e. 2D coordinates of points on the screen and images of the eyes.

In the work by Baluja and Pomerleau [2], an artificial neural network (ANN) was trained to model the relationship between 15x40 pixel images of a single eye and their corresponding 2D coordinates of the observed point on the screen. In their calibration procedure the user was told to look at a cursor moving on the screen along a path made of two thousand positions. They reported a best accuracy of 1.5°.

Similar work by Xu et. al. [12] was presented, but instead of using raw image values as inputs, they segmented out the eye and performed histogram equalization in order to boost the contrast between eye features. They used three thousand points for calibration and reported an accuracy of around 1.5°.

2.3. Our Approach

A key motivation behind this work is that gaze tracking systems lack the required robustness for commercial ap-

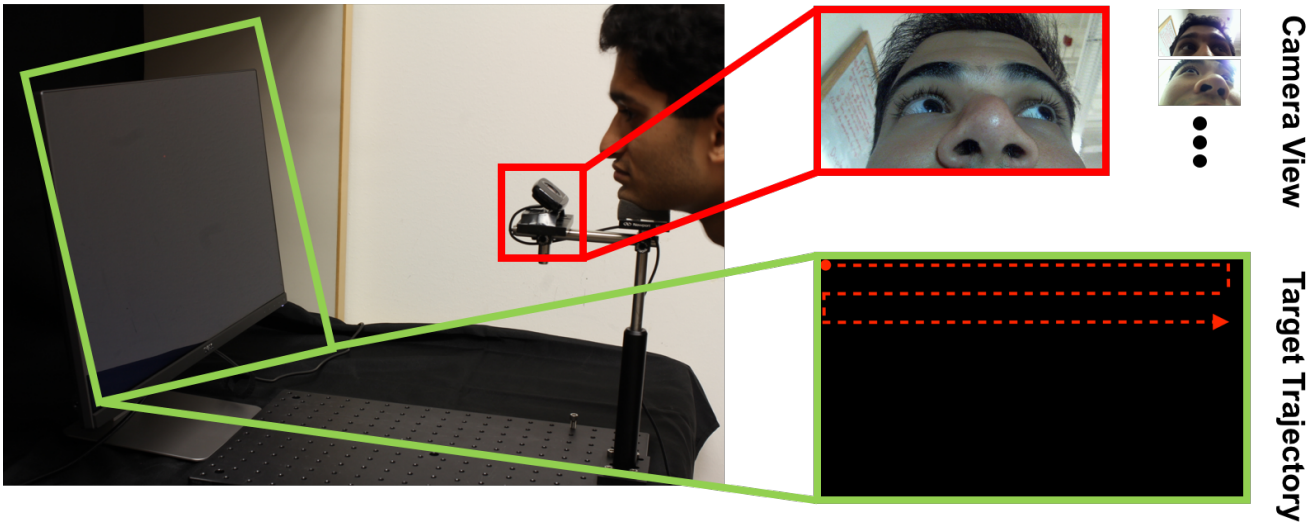

Figure 2. Image of setup, sample captures, and path of point on screen. The left image depicts our setup comprising of an LCD monitor, webcam, and chin rest to keep the head roughly stable. The upper right images show samples images from what the webcam captures. The bottom right image displays a part of the path that the moving target follows during its trajectory.

plications. High variability in the appearance of the pupil and lack of a clear view cause accuracy issues with detector based tracking systems. Deep learning performs well on problems with high intra-class variability. The intuition behind this work was to use a CNN model instead of the parametric calibration since it would probably be more robust to variations in skin/eye color, HMD fixation etc.

In particular the key contributions of this work are:

- Introduction and implementation of an end-to-end CNN based approach to gaze tracking
- Creation of a new gaze tracking dataset with a near-eye camera covering five different subjects and dense position sampling of the screen based on smooth pursuit
- Performance evaluation of the chosen method, conclusions and suggestions for improvement that can be incorporated in future work

3. Dataset

Many eye-gaze tracking datasets exist [14], as seen in Figure 1, however they are mostly tailored for model assumed REGT systems. The majority of the datasets use target based calibration with large spacing between targets. Neural networks are not able to train on such few, and sparsely, sampled points and learn a good relationship between image data and pixel coordinates. The few datasets that used continuous targets also allowed for continuous head movement, which is not a good representation for a near-eye display where the display is strapped to the user's head (with the camera rigidly fixed inside of it).

Instead of using one of the above datasets, we decided to create our own dataset suitable for neural network training. Our strongest criteria was to have a large number of calibration points densely sampling the entire screen, with corresponding images of the eye. With such training data, we would expect the CNN to learn the fine differences between points on the screen.

In our setup, as seen in Figure 2, we placed a user with his/her head resting on a chin rest 51 cm away from a 1080p 24 inch monitor. A webcam was placed very close to the chin rest imitating what a camera placed inside of a near-eye display would see.

Asking a user to fixate on a series of targets is infeasible for the large number of calibration points we wanted to collect. Instead we used the fact that humans are able to track moving objects well, up to some angular velocity. This is the notion of smooth pursuit. We moved a point about a screen at 7.5°/s in a winding pattern from left to right, top to bottom, as seen in Figure 2 and were able to collect 7316 calibration points during a single 4 minute sitting. At this angular velocity the eye is able to smoothly track the point as it moves about without sacades (which occur when the eye attempts to 'catch up' when a point is moving very quickly). We chose this particular angular velocity based on [11], which introduced the concept of a pursuit based calibration and found that points moving about between 3.9°/s and 7.6°/s resulted in best accuracy.

In order to achieve a smoothly moving point, we displayed four points per angular degree, giving us a total of 7316 points. We captured webcam video frames at approximately 30 fps, which roughly corresponded to one frame per

1/4 angular degree point shift. Because our goal was around 1°of accuracy, we binned the calibration points into 1°bins. For example, points displayed between 0.5°and 1.5°would be considered the same as 1°. We found that with a 4x reduction in classes, and a corresponding 4x increase in points per class, the CNN was able to better learn.

The cropped and downsampled captured dataset can be found at [1]

4. CNN Learning Approach

In this work we explore the use of Convolutional Neural Networks (CNN) for gaze tracking in an end-to-end fashion. Traditional approaches like the one mentioned in the previous section rely on hand-engineered feature detectors and then use a parametric model to track the gaze direction of a user.

Recently CNNs have outperformed traditional feature engineering-based computer vision methods in a variety of tasks. This work explores their use for gaze tracking. The key benefit is that this approach is fully data driven. We train the CNN model to take images of the users eye (taken from a camera very close to their face) as input and estimate the gaze direction in terms of x and y pixel coordinates on the screen.

4.1. Description of CNN

CNNs are a class of artificial neural network algorithms with multiple hidden layers that are built using convolutional layers. In this work we treat the gaze tracking problem as a multi-class classification problem, where each class is a specific point on the screen, and use a simple CNN (i.e. LeNet) to learn the mapping from images to gaze position.

LeNet consists of two convolutional layers and two pooling layers followed by a fully connected layer at the end. For our implementation, we modify the fully connected (FC) layer to the desired number of output classes for the gaze tracking problem.

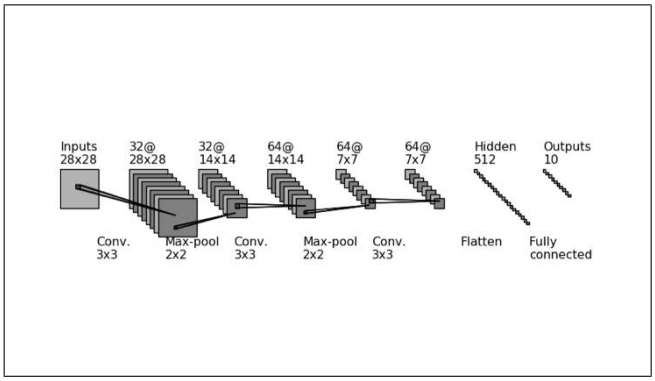

Figure 3. LeNet Architecture

5. CNN Implementation

5.1. Caffe

The CNN implementation was performed in both Caffe and Tensorflow for comparison. Caffe has LeNet in the model repository implemented for MNIST digit classification. The .prototxt file was reconfigured to point to the lmdb file for our dataset. The image files and binary with class labels are converted to the Caffe compatible lmdb file and then training is performed. The figure below shows the setup workflow for the Caffe implementation.

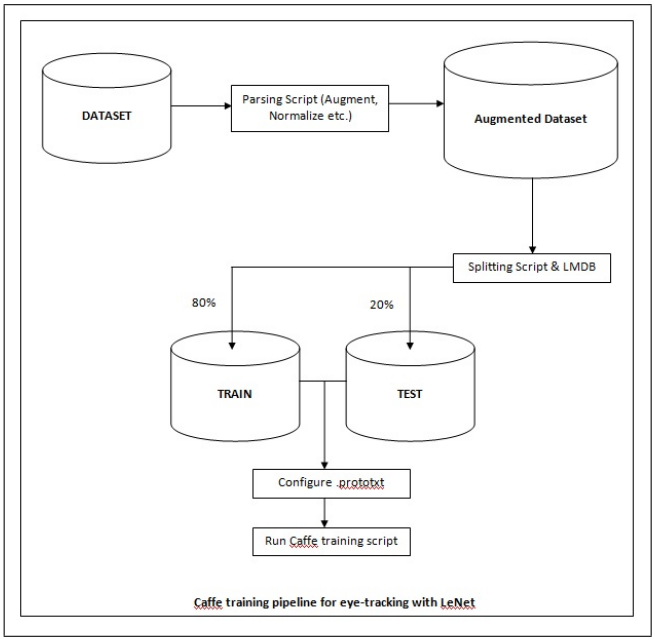

Figure 4. Caffe Model Architecture

As the results were not very promising and training was suspending abruptly, we later focused on the TensorFlow implementation to complete the project.

5.2. TensorFlow

Tensorflow also consists of the LeNet model as an example trained for MNIST. The example was reconfigured by changing the number of classes to train on, size of convolutional kernels etc. to work with the CAVE and captured gaze tracking dataset. The figure below shows a visualization of the TensorFlow model.

Some of the parameters that were explored were the size of the convolutional kernels and number of convolutional layers. Unfortunately, due to limited computing power, it remained difficult to add too many layers to the network.

6. Data Organization

Due to the large size of the data captured and the limited computational power available, a few steps were taken to

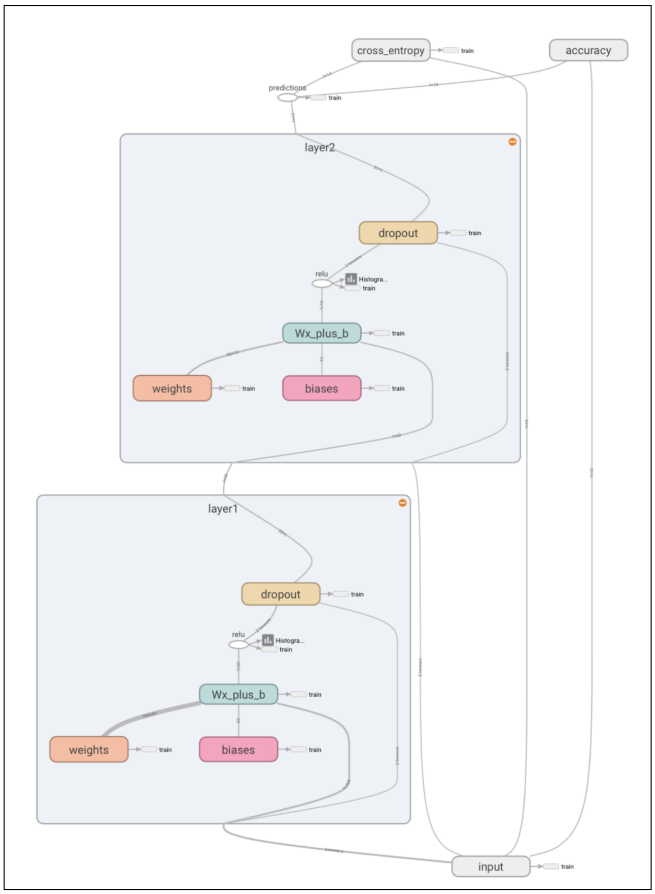

Figure 5. TensorFlow Model Architecture

make the data manageable while not compromising the data available for the CNN.

6.1. Dataset Augmentation

The dataset consisting of cropped images of the eye and their respective pixel coordinates for the gaze was augmented before training the CNN. The augmentation consisted of adding different types of noise (Poisson, speckle, Gaussian etc.) to make the model more robust to variations in the image during inference. The augmentation extends the dataset which is very useful as many samples are needed to fully train the CNN model. The augmented dataset is then spit into training, test and cross-validation sets depending on the requirements.

6.2. Image Preprocessing

The pipeline to feed both the CAVE and the captured dataset were as follows:

1. Convert Images to Grayscale
2. Downsample Images to 28x28 (for efficiency)
3. Crop out eye region from each image

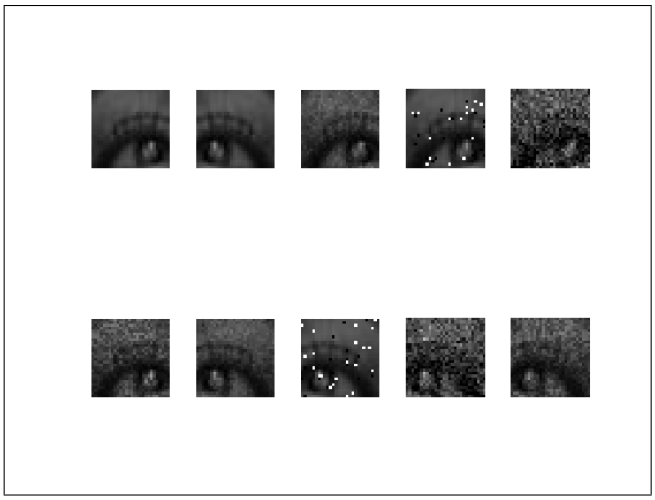

Figure 6. Augmented Images

4. Save images and respective labels

The size 28x28 matched the dimensions of the MNIST dataset. More importantly, we found that downsampling to a slightly large size, like 50x50, did not necessarily lead to a better classification accuracy. Similar results were found in [] where a larger image did not mean better neural network performance.

Cropping out the eye region was done differently for the CAVE and captured dataset. For the CAVE dataset, a very basic estimation of the bounding box around both eyes was formed. The same bounding box was applied to all images for a particular pose. This was an error-prone methodology but did better than using eye recognition software, which often failed on subjects with glasses. For the captured dataset, we manually selected the region around the eyes per subject and then applied the same crop region to the rest of the images for that particular subject. This manual processing was also not perfect, since subjects moved slightly between frames.

The labels were created using one-hot vectors - the length of the vector was equal to the number of classes per experiment and the index of the class the image belonged to was a 1 while the rest of the entries were 0. The images and the label vector pairs were converted to Tensors/lmdb format for ease of processing.

7. Results

The CNN was initially tested with the CAVE dataset to ensure the model learned something. Next, the same model was applied to the captured dataset, where the images were significantly better (they were focused just on the eyes) but the number of classes was significantly greater (21 classes for the CAVE dataset and 1829 classes for the captured dataset).

Figure 7. CAVE Dataset Downsampled Images

Figure 8. Captured Dataset Downsampled Images

7.1. CAVE Dataset

Our initial gaze tracking implementation was based on the CAVE (Columbia Gaze Data Set) [15] dataset. The dataset has 5,880 images collected for 56 subjects for 21 gaze directions for 5 distinct head poses. To simulate the near-eye display conditions we only use one head pose i.e. the subject directly facing the screen. Even though the dataset was not built for fine-grained tracking and the captured images are highly inconsistent, the results were promising.

The problem was framed as a gaze localization problem where the screen is divided into 21 (7X3) grids which form the 21 output classes for the CNN model. We show results obtained on the CAVE dataset with 3 output classes (one of the three horizontal bands on the screen) and with 21 output classes. The graphs show how the classification accuracy (measured as the fraction of gazes correctly classified) changed across iterations of training the CNN. Each training batch was 100 images and the CNN was trained for 500 iterations. The figures 9 and 10 show the training accuracy increasing across iterations. The red dotted line shows the baseline classification accuracy - $\frac{1}{\text{number of classes}}$.

While the accuracy for the 21 class case isn't significant, the model does learn a mapping and performs better than a random guess over the output classes. From the CAVE dataset images, it is easy to see why the model does not

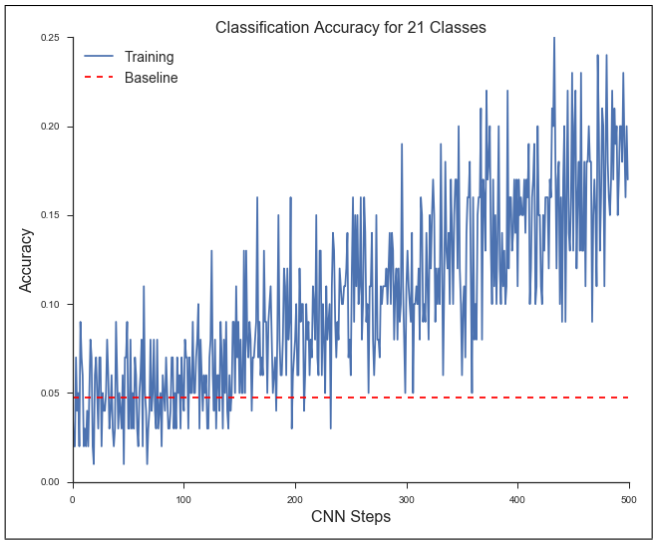

Figure 9. CAVE Classification Accuracy with 21 Classes

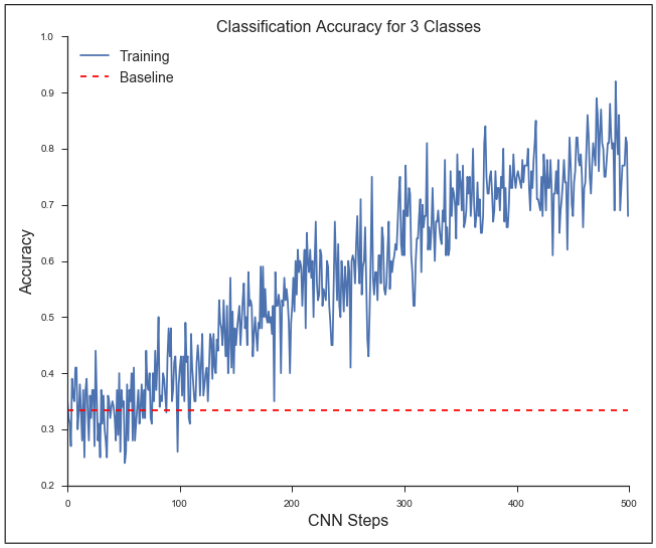

Figure 10. CAVE Classification Accuracy with 3 Classes

learn as well as it would be expected to. Out of the 28x28 pixels, a very small fraction actually comprises of the eyes and the pupils. In the results, we look at training accuracy which can be interpreted as the inverse of the loss functions and shows the progress of the model as it learns a better model. The trained model is then tested on a batch from the test set to get an accuracy which represents the percentage of correctly classified images. For the 3 and 21 class case, the final test accuracy was 0.893 and 0.236.

7.2. Captured Dataset

Since the number of classes with the real dataset was significantly large, the classification accuracy is not a good measure of how well the model does. If the true and estimated classes are next to each other, that is a misclassifi-

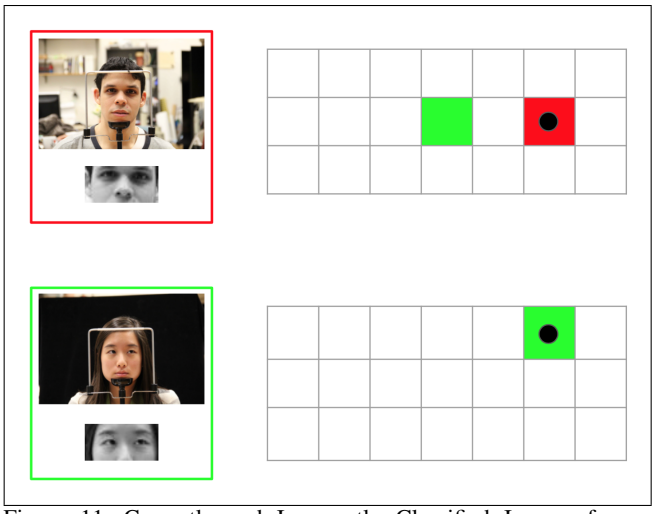

Figure 11. Correctly and Incorrectly Classified Images from CAVE Dataset

cation and measured the same as if the true and estimated classes were on opposite corners of the viewing screen.

Since our goal is to track the user's gaze as accurately as possible, we also look at angular error i.e. actual difference in angle between the inferred gaze and the actual gaze of the user. Looking at this particular metric, we can see that the gaze can be localized to a few degrees in accuracy but there is scope for improvement. Looking at confusion matrix or classification accuracy is not appropriate for this problem as it does not capture the degree of misclassification or how far the inferred gaze is from the actual gaze direction.

The testing classification accuracy was 0.003. Even though this is ridiculously low, the testing angular error is only 6.7 degrees. And in the gaze tracking sense, the angular degree matters much more than the classification accuracy.

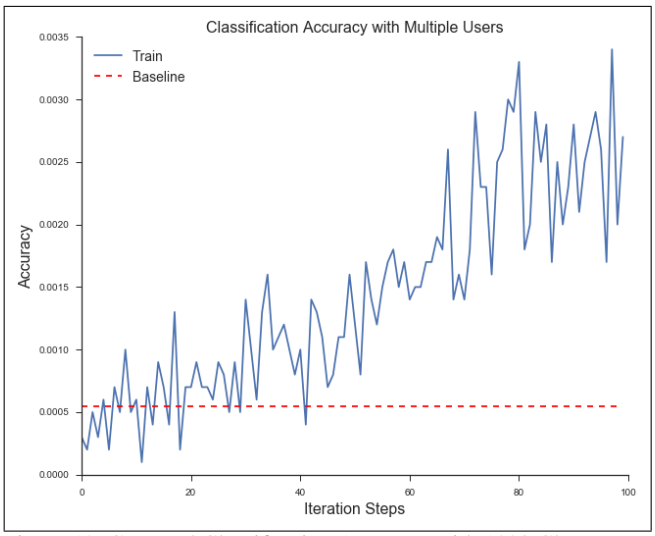

Figure 12. Captured Classification Accuracy with 1829 Classes

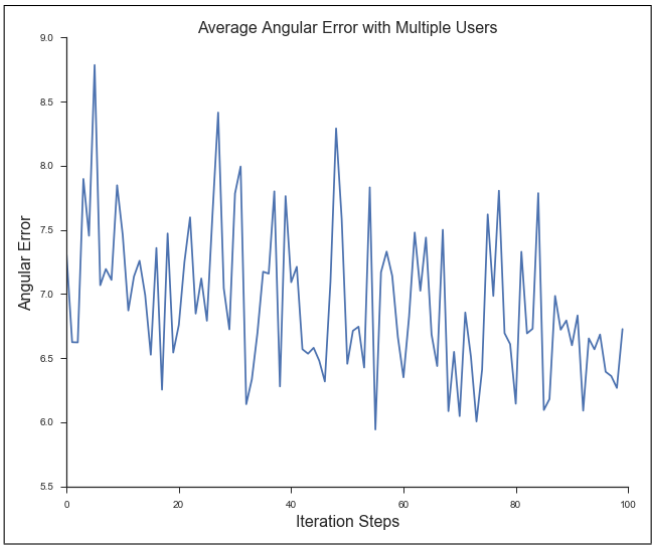

Figure 13. Captured Average Angular Error

8. Conclusion and Future Work

We explored an end-to-end deep learning approach for gaze tracking in a highly constrained environment (camera very close to the eye) and obtained promising results with a vanilla implementation. This demonstrates the viability of this approach and also highlights some of its shortcomings.

CNN needs a lot of data to be properly trained and lack of publicly available datasets for fine gaze tracking was a major impediment. We implemented our own data capture setup and created an appropriate dataset for this purpose but given the severe time constraints were only able to gather data for 5 subjects. This dataset will be extended in the future for more subjects and made publicly available for future experiments.

The model used for this work was the simplest instantiation of a CNN and the eye images had to be significantly downsampled even with the simple model to facilitate quick training. This is one of the limitations of using CNNs and prevents quick iterations or cross-validation.

We believe a higher quality input image of the users eye that also has support features (fixed in the image irrespective of the subject) will help improve the gaze tracking accuracy. This is also the situation that will be useful when doing tracking in a near-eye display. Given that the HMD is fixed to the user's head, the camera should see a consistent images across different users. The model can also be more complex but due to lack of training data, it did not make sense to increase model complexity in this work.

To conclude, the work shows promise in using a deep learning approach for gaze tracking and has potential to outperform the feature based methods based on parametric models. However, further exploration is needed to achieve state of the art results.

References

[1] Captured Dataset. https://www.dropbox.com/s/mxbis1osiedclkd/data.zip?dl=0.

[2] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. Technical report, Pittsburgh, PA, USA, 1994.

[3] B. A. Barsky and T. J. Kosloff. Algorithms for rendering depth of field effects in computer graphics. pages 999–1010, 2008.

[4] S. Hillaire, A. Lecuyer, R. Cozot, and G. Casiez. Using an eye-tracking system to improve camera motions and depth-of-field blur effects in virtual environments. In *Proc. IEEE VR*, pages 47–50, 2008.

[5] S. Hillaire, A. Lcuyer, R. Cozot, and G. Casiez. Depth-of-field blur effects for first-person navigation in virtual environments. *IEEE Computer Graphics and Applications*, 28(6):47–55, Nov 2008.

[6] T. E. Hutchinson, K. P. White, W. N. Martin, K. C. Reichert, and L. A. Frey. Human-computer interaction using eye-gaze input. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1527–1534, Nov 1989.

[7] R. Konrad, E. Cooper, and G. Wetzstein. Novel optical configurations for virtual reality: Evaluating user preference and performance with focus-tunable and monovision near-eye displays. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI16)*, 2016.

[8] G. Maiello, M. Chessa, F. Solari, and P. J. Bex. Simulated disparity and peripheral blur interact during binocular fusion. *Journal of Vision*, 14(8), 2014.

[9] M. Mauderer, S. Conte, M. A. Nacenta, and D. Vishwanath. Depth perception with gaze-contingent depth of field. *ACM SIGCHI*, 2014.

[10] C. H. Morimoto and M. R. M. Mimica. Eye gaze tracking techniques for interactive applications. *Comput. Vis. Image Underst.*, 98(1):4–24, Apr. 2005.

[11] K. Pfeuffer, M. Vidal, J. Turner, A. Bulling, and H. Gellersen. Pursuit calibration: Making gaze calibration less tedious and more flexible. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 261–270, New York, NY, USA, 2013. ACM.

[12] L. qun Xu, D. Machin, P. Sheppard, M. Heath, and I. I. Re. A novel approach to real-time non-intrusive gaze finding, 1998.

[13] T. Shibata, T. Kawai, K. Ohta, M. Otsuki, N. Miyake, Y. Yoshihara, and T. Iwasaki. Stereoscopic 3-D display with optical correction for the reduction of the discrepancy between accommodation and convergence. *SID*, 13(8):665–671, 2005.

[14] B. Smith, Q. Yin, S. Feiner, and S. Nayar. Gaze Locking: Passive Eye Contact Detection for HumanObject Interaction. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 271–280, Oct 2013.

[15] B. Smith, Q. Yin, S. Feiner, and S. Nayar. Gaze Locking: Passive Eye Contact Detection for HumanObject Interaction. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 271–280, Oct 2013.

[16] K.-H. Tan, D. J. Kriegman, and N. Ahuja. Appearance-based eye gaze estimation. In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, WACV '02, pages 191–, Washington, DC, USA, 2002. IEEE Computer Society.

[17] S. Xu, X. Mei, W. Dong, X. Sun, X. Shen, and X. Zhang. Depth of field rendering via adaptive recursive filtering. In *SIGGRAPH Asia 2014 Technical Briefs*, SA '14, pages 16:1–16:4, New York, NY, USA, 2014. ACM.

[18] T. Zhou, J. X. Chen, and M. Pullen. Accurate depth of field simulation in real time. *Computer Graphics Forum*, 26(1):15–23, 2007.

[19] J. Zhu and J. Yang. Subpixel eye gaze tracking. pages 124–129, May 2002.

[20] Z. Zhu and Q. Ji. Novel eye gaze tracking techniques under natural head movement. *IEEE Transactions on Biomedical Engineering*, 54(12):2246–2260, Dec 2007.