Gaze Estimation with Deep Convolutional Neural Networks

Alexandros Ferles, Erifili Ichtiaroglou, Pantelis Myriokefalitakis

KTH Royal Institute of Technology
DD2424 Deep Learning in Data Science

Abstract. *The problems of eye-tracking and gaze-estimation consequently, can be met in related research among various fields. A plethora of methods have been proposed in order to tackle this problem effectively, but until recently no robust solution was suggested. The work of Krafka et. al[1] introduces the creation of a large-scale datataset among with the implementation of iTracker; a robust Deep Convolutional Neural Network that can successfully predict the gaze position, with insignificant deviations from the ground truth. To further increase our understanding on Deep Learning and Convolutional Neural networks through a non-trivial task, we implement the iTracker model and present our results on the GazeCapture dataset.*

Keywords: Computer Vision, Human-Computer Interaction, Deep Machine Learning, Convolutional Neural Networks, Eye tracking, Gaze Estimation

1 Introduction

Convolutional Neural Networks (CNNs) is an Artificial Neural Network architecture which is widely used to address Computer Vision applications and extract valuable information that can be used in a variety of domains and applications. Deep Learning implementations of CNNs firstly rose to prominence due to the recent work of Krizhevsky et. al[2], when a significant drop on the ImageNet classification error was reported. The research interest in this domain has been kept high ever since, and a numerous of image recognition and object detection applications have been reported up to this day[3][4].

A typical CNN architecture consists of a convolutional layer, serving as a filter that produces an *activation map* which will be additionally filtered through an activation function and will serve as input in the pooling layer. The activation map contains *feature identifiers* which are able to isolate local features of the data (e.g a curve on an image) that will be combined accordingly to produce a suitable solution. While functions such as the *sigmoid* and *tanh* functions have been widely used in activation layers of neural networks, lately the *ReLu* function was introduced in [2] and has since been adapted in many related applications. The pooling layer minimizes the dimensionality of the input data by selecting

the features that are dominant in local areas of the raw data. These stages can be replicated many times, allowing for a deep network to progress from learning low-level to high-level features on the input data. The results of this process serve as the input layer to fully connected neural network, and its weights are configured to predict the gaze position. Loss is minimized through back-propagation

1.1 Gaze estimation and eye detection

Detection of the position of the eye in an image or a video frame, is a well-known problem in Computer Vision applications. It is of great importance to numerous scientific fields such as Gamification and Psychology, and can be formulated as follows: Given an image such as a video frame of a person, our primary purpose is to locate the eye position and predict the exact spot that our subject will look at. The performance of the prediction (x_{pred}, y_{pred}) can be measured through distance metrics, such as its mean Euclidean distance *eucl* from the target position (t_x, t_y) in the 2D plane:

$$\min eucl(x_{pred}, y_{pred}) = \frac{1}{2}\sqrt{(t_x - x_{pred})^2) + (t_y - y_{pred})^2} \tag{1}$$

Gaze estimation methods can be divided into *shape-based* methods, *appearance-based* and *hybrid-based* methods and combinations of both, known as *hybrid* methods. Shape-based methods operate by firstly locating the relative position of the eye into the grid, and then based on prior eye models focusing locating features like the pupil and the iris. Light sources are requested in order for proper results to be derived[5]. Appearance-based methods, use input images as representations in higher dimension feature spaces, and estimate the location of the gaze through extraction of principal features.

The rest of this paper is structured as follows: In section 2, we discuss related work, while in section 3 we present our approach in building our own version of iTracker. In section 4, the results of our experiments are presented, while Section 5 and 6 delve with our conclusions and future work suggestions respectively.

2 Related Work

In this section, we turn our attention into related literature that provides methods for estimating the gaze position in the 2D plane. We begin with earlier implementations that concern appearance-based methods and may involve hybrids of these methods with other Machine Learning techniques, ending up to end-to-end CNNs, similar to the work of Krafka et. al[1].

2.1 Appearance-based models

Lai et. al[6] use both the head pose variable with an eye appearance vector in the image to construct a high dimensional feature space. This space is then used

to create a random-forest[7] based neighborhood, which is evaluated through reconstruction weight-modification in the neighbor sets. Through regression, the final estimation of the gaze position is delivered.

With the technique of Artificial Neural Networks, combined with feature extraction, TóSér et al.[8], integrate the CNN in [9] by adding the use of dropout[10] and using rectified linear units in every layer except the output layer. The training process was furtherly configured with Adamax[11] and early stopping. The results of their CNN were boosted with the parameters of head pose and pupil position.

Wang et. al.[7] use the DeepID[12] architecture (three convolutional layers combined with max-pooling layers, one max pooling layer, one hidden layer and a softmax layer) to extract deep features of the images of the eye. Based on the results of [13], where random forests were reported to outperform other appearance-based gaze estimation models, they combine the extracted deep features with the head position and use random forests to rate many candidates of the gaze position and decide by ensemble voting.

In the work of [14] the extracted activation pattern of an eye image is fed to a neural network with a two-split hidden layer which learns the x and y co-ordinates of the gaze respectively.

2.2 End-to-end appearance-based CNNs

CNNs that could track the gaze without requesting standard position of the subjects participating firstly appeared in the work in [15].

Zhang.et.al[9] use the work of[16] (two stages of a convolutional and a max pooling layer as input to a fully connected layer) that takes the angles of the original gaze position as an input, and predicts the angles of the next gaze position. The output of this multimodal CNN is concatenated with the original head angles and the final prediction is derived with the usage of linear regression layer. Performance is rated via the L_2 loss between true gaze angle vectors and the predicted ones.

The work of Krafka[1] proposes the *iTracker*, a solid deep CNN architecture, that with the addition of the large-scala *GazeCapture* dataset provides state of the art results, and achieve minimum error performance. We implemented *iTracker*, accompanied with various takes in the training process and modifications in its hyperparameters, tested on a subset of *GazeCapture* with adequate size. Since this architecture constitutes the main work of this paper, we present it in full detail in section 3.

In a similar fashion like [1], Zhanget et. al[17] proposed a deep CNN that extends the network in [2] and combine it with pixel-by-pixel classification[18].

Before passsing the output of the convolution layers to the fully-connected layers, spatial weights are applied to the extracted features. This results in assigning varying importances to these features, thus enabling the regression stage to delve with critical information first. This re-definition of the AlexNet network improves the state of the art results provided by [1].

2.3 Other resources

For a full and comprehensive list of Computer Vision models (not necessarily into the Deep Learning domain) proposed for eye detection and gaze estimation, the reader is strongly suggested to read the work of Hansen[19].

3 Approach

In this section we describe the approach we followed for building the gaze estimation model. Based on the paper of Krafka, Kyle et al [1], we tried to re-implement the described network and train it on a small subset of the GazeCapture Dataset[1]. In section 3.1 we describe the characteristics of the dataset used and in section 3.2 we present the architecture of the final model and describe the details of the network, such as the parameters used and the complexity.

3.1 GazeCapture Dataset

GazeCapture is a large-scale dataset obtained through the use of crowdsourcing. In particular, an application was built, where users had to follow dots appearing on the screen, while the front-facing camera was recording their gaze.

In order for the task to provide reliable data, since it was not supervised, users had to activate Airplane Mode, to avoid distractions of any notification messages and tap on the left or right side of the screen, when the corresponding letter appeared (L, R). Thus, the researchers were sure that the users were only focusing on the task.

Another really important aspect of a dataset is variability. A network trained by such a dataset, would be able to generalize better and provide more accurate results for a wide range of input data. GazeCapture dataset was designed to fulfill this requirement as well. The users varied greatly in face characteristics, pose and illumination [1]. They were also instructed to move their heads and change the phone distance relative to them, as well as to change the rotation of the phone while they were following the dots. The amount of the participants in combination with these techniques ensure the variability of the dataset.

The whole dataset was collected from 1474 participants and consists of a total of 2,445,504 frames with fixation locations [1]. The information provided by the dataset include images of a person's face and eyes, a face grid indicating

where the face appears in the image and the dot location given as the vertical and horizontal distance from the front camera (in centimeters).

3.2 Architecture of the model

In the paper of Krafka, Kyle et al [1] the network chosen was a CNN with the size of each layer similar to those of the AlexNet [2]. The success of the Convolutional Neural Networks, CNNs, in computer vision, motivated the decision to design a network where the inputs would be the images, without including any manually engineered features. Thus, given enough input data, the network would be able to extract appropriate features and use them to estimate accurately the gaze fixation.

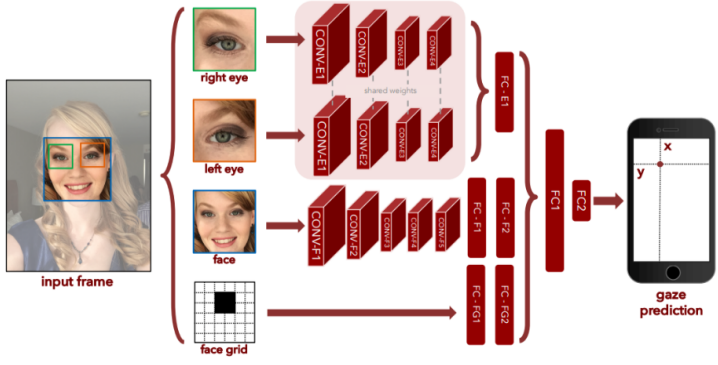

Fig. 1: CNN architecture used for the gaze estimation model [1].

Figure 1 presents the main architecture used for the gaze estimation network. Some variations of the main model were implemented, in order to examine the importance of every feuture (eyes, face, face-grid) for the accuracy of the model. However, all of the models are based on the main one. Table 1 provides a more detailed representation of this model and information about the size of each layer, according to [1] and the AlexNet network [2].

Layer	Kernel Size/filters	Stride
ConvE1-F1	11 x 11/96	1
ConvE2-F2	5 x 5/256	1
ConvE3-F3	3 x 3/384	1
ConvE4-F4	1 x 1/64	1
FC-E1	128	-
FC-F1	128	-
FC-F2	64	-
FC-FG1	256	-
FC-FG2	128	-
FC1	128	-
FC2	2	-

Table 1: Size of every layer for the gaze estimator network.

After each layer, the *rectified linear unit (Relu)* was used as an activation function, as it was proposed by [2]. The reason for using *Relu* instead of *sigmoid* or *tanh* functions, which adds non-linearity in the model, was to overcome the phenomenon of vanishing gradients. The gradient of the Relu function is either 0 (for negative of zero inputs) or 1 (for positive inputs). Thus, the gradients of the network will not tend to zero as we go back to the early layers, which is the case when using sigmoid function, where the gradients vary between 0 and 0.25.

After each convolutional layer, we applied a max pooling layer of size 2x2 and stride 1. This layer performs downsampling and reduces the amount of the network parameters and thus, the computational cost. At the same time it controls over-fitting, due to the fact that it selects a subset of the features, hence we are less likely to train the model on relations that might not exist.

Having decided the size of each layer and the output in Table 2 we present the number of the parameters for the network and the complexity.

Layer	Kernel Size/filters Stride	Parameters	FLOPs	Outup Size
Input				3x64x64
ConvE1-F1	11 x 11/96 1	(11x11x3 +1)*96*2 = 69,888	(11x11x3 +1)*96*54*54*3 ≈ 306M	96x54x54
max-pool1	2x2 2	0		96x27x27
ConvE2-F2	5 x 5/256 1	(5x5x96+1)*256*2 =1,229,312	(5x5x96+1)*256*23*23*3 ≈ 975M	256x23x23
max-pool2	2x2 2	0		256x11x11
ConvE3-F3	3 x 3/384 1	(3x3x256 +1)*384*2 = 1,770,240	(3x3x256 +1)*384*9*9*3 ≈ 216M	384x9x9
max-pool3	2x2 2	0		384x4x4
ConvE4-F4	1 x 1/64 1	(1x1x384 +1)*64*2 = 49,280	(1x1x384 +1)*64*4*4*3 ≈ 1.2M	64x4x4
max-pool4	2x2 2	0		64x2x2
FC-E1	128	(2x2x2x64 +1)*128 = 65,664	65,664	128
FC-F1	128	(2x2x64 +1)*128 = 32,896	32,896	128
FC-F2	64	(128+1)*64 = 8,256	8,256	64
FC-FG1	256	(25*25+1)*256 = 160,256	160,256	256
FC-FG2	128	(256+1)*128 = 32,896	32,896	128
FC1	128	(128+64+128+1)*128 = 41,088	41,088	128
FC2	2	(128+1)*2=258	258	2
$\sum$		3,460,034	≈1,5billion	

Table 2: Number of parameters and forward computations(FLOPs) for the main model

4 Experiments

In this section, we are going to discuss about the experimental setups as well as the evaluation of the model using a subset of the large-scale GazeCapture Dataset [1]. Beside the main architecture of the model, which was described in section 3.2, we performed some alternations in order to better understand and assess the use of different input features such as eyes, face, face grid or some combinations of them and how they affect the efficacy of the model. In addition, we performed some more experiments by changing the activation function from ReLU to ELU and/or adding some dropout layers to deal with overfitting.

4.1 Data preparation

A part of the GazeCapture dataset was used for training and evaluating the model. The dataset consists of 68000 frames, where both the 2 eyes and the face were detected. A sub-set of 48000 frames were used for training, 5000 were used for validation and 15000 frames were used for testing the model. From each frame we extracted images for the *2 eyes*, the *face* and the *face grid*, which is a mask that indicates the position of the face in the initial image. These four types of data were used as input to our model. The image size of the 2 eyes and the face frame is 64x64x3, while the image size of the face grid is 25x25.

Prior to training we processed the input data, so that they have values in the range of (-1,1). The purpose of this normalization procedure is that each feature has similar range, thus the parameters would act uniformly and the gradients would not get out of control. This practice facilitates the learning procedure and makes it faster.

4.2 Model setup

In order to implement the model, TensorFlow API was used. The final experiments were executed on K80 nodes of Google cloud platform.

Initially, our concern was to re-implement and study the iTracker model, as it was firstly done by Krafka et. al in [1], with the same architecture. Moreover, something of interest to us was to examine which types of the input data (eyes, face, face grid) were the ones that mostly influence the error rates. Thus, we performed some more experiments with different combinations of input data types. For example, we conducted experiments where only the face or the eyes were used as input to the network and we removed all the extra convolution and fully-connected layers, corresponding to the rest of the inputs. The results from these experiments, as well as from the iTracker model are presented in Table 3. It is important to mention that up to this point, any other parameter was kept stable.

During the training procedure, Adam optimizer was used for the weights' update with momentum 0.9. The number of epochs was set to 100 and the batch size for both training and validation procedure was set to 128. In order to avoid overfitting and stop network's training when convergence occured we used early stopping (on average 37 epochs were needed). The learning rate was initialized to 0.001 and it was moreover decreased after some learning steps. We assess the accuracy of the model by computing the mean squared error between the predicted output and the true fixation points (in centemeter).

After having conducted the main experiments which were described above, we proceeded with further experiments in an attempt to reduce the error in lower levels. These experiments (whose results can be found in Table 5) mainly concentrated in:

1. Joining the two separate fully connected layers for face (FC-F1) and face grid (FC-FG1) to a combined fully connected layer. This idea was borrowed from the combination of the left and right eye in the first sub-part of the iTracker. Analogously, the available data of face and face grid concern the same area of the input, and thus can be combined.
2. Tuning the network by trying different activation functions. Rectified linear Unit (ReLU) was substituted by Exponential Linear Unit (ELU). The reason behind this change was the fact that ReLU removes the negative values. Our purpose was to find out the effect of small gradients close to zero instead of zero ones.
3. Tuning the network by trying different optimizers. Instead of the Adam, we tried the mini batch Gradient Descent optimizer. We would like to examine its effects on the model and on the other hand we wanted to observe if this algorithm does make the model to generalize better, as it is suggested in paper [20] even though it converges slowly.
4. Using dropout layers to reduce overfitting by shutting down some nodes of the layers which had the biggest number of parameters like the fully-connected layer FC1.

4.3 Results

Model(opt)	Train/Val error	Epochs	lr rate	Test error
main model (Adam)	0.83201/1.70202	49	0.001	**5.195964**
face only (Adam)	1.19942/1.76550	32	0.001	5.739900
eyes only (Adam)	1.57277/2.14380	22	0.001	6.176817
face + eyes (Adam)	1.12779/1.77996	29	0.001	4.770012
face + face grid (Adam)	1.23526/1.84256	55	0.001	6.151198
eyes + face grid (Adam)	1.18659/1.95578	38	0.001	6.639260

Table 3: Results from the main model(iTracker) and some alternations of the input types.

Model (act fun)	Optimizer	Train/Val error	Epochs	lr rate	Test error
face only (ReLU)	mini batch GD	0.70581/1.59713	50	0.002	5.277456
modified model (ReLU)	adam	0.88519/1.69867	35	0.001	5.334911
main model + dropout(0.2) (ReLU)	adam	1.67403/2.16666	53	0.001	5.299715
main model (ReLU)	mini batch GD	1.52909/2.18849	38	0.002	5.138686
modified model (ELU)	adam	0.75722/1.65689	43	0.001	**4.453536**

Table 4: Results from the alternations of the main model(iTracker) architecture.

As we can observe in table 3 the main model had the second best performance with 5.19cm error on unseen data. The best performance, though, was given by the same model without the face grid input. This may seem strange at first, baring in mind that the main model takes advantage of more information by having the face grid as input. However, the input images were cropped to include only the face area of the frame and, thus, the face grid may introduce some extra, non-needed noise to the model, leading to these results. One could also argue that the eyes are of no inportance as input to the model, since they are already included in the face image. Although, this is not the case, as the combination of both face and eyes reached to lower error rates. The rest of the results in table 3 seem quite reasonable, as the error increases when we remove useful information such as the eyes and/or the face from the model.

By observing the evolution of the loss in figures 2 and 3 we see that the mini batch Gradient descent optimizer was able to generalize better, even though we got higher error. This may mean that with fine tunning we could achieve optimal results. The same conclusion was reached by using dropout layers, although the model converged slower in both cases. Overall, the best results were given by the modified model using ELU as activation function. The fact that ELU keeps a

bit of information for negative values instead of setting them to zeroo, as ReLU does, has contributed to the quality of these results.

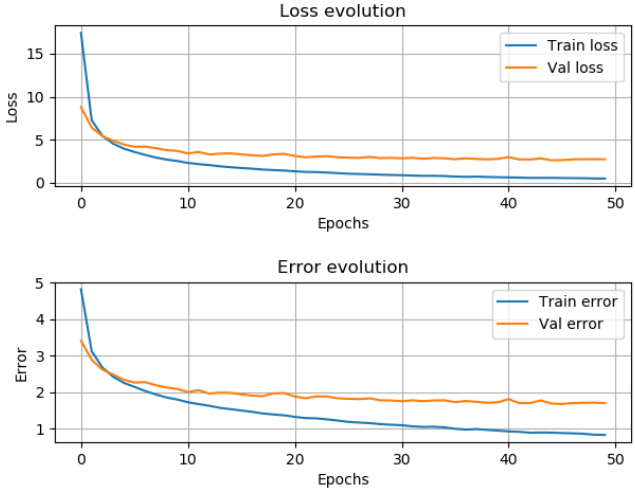

Fig. 2: Loss and error evolution after training the main model (*main model(Adam)* in table 3). We achieved test error 5.19 cm.

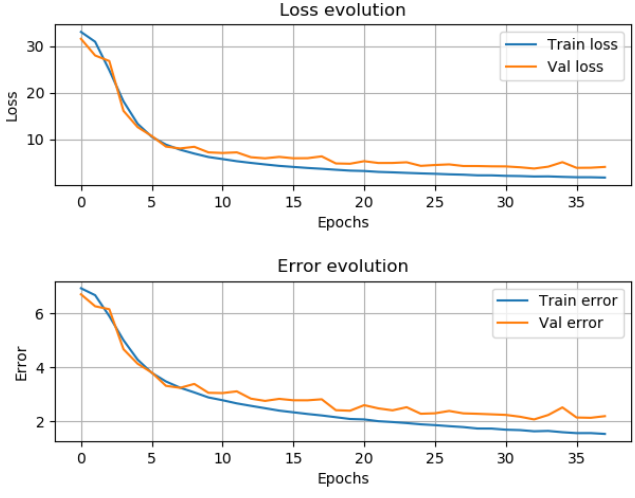

Fig. 3: Loss and error evolution after training the main model applying mini batch Gradient Descent optimizer with batch size 256 (*main model(ReLU)* in table 4) We achieved test error 5.14 cm.

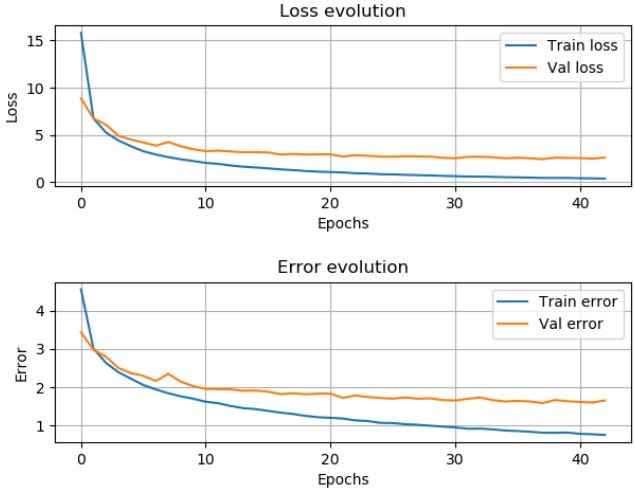

Fig. 4: Loss and error evolution after training the modified model (*modified model(ELU)* in table 4) We achieved test error 4.45 cm.

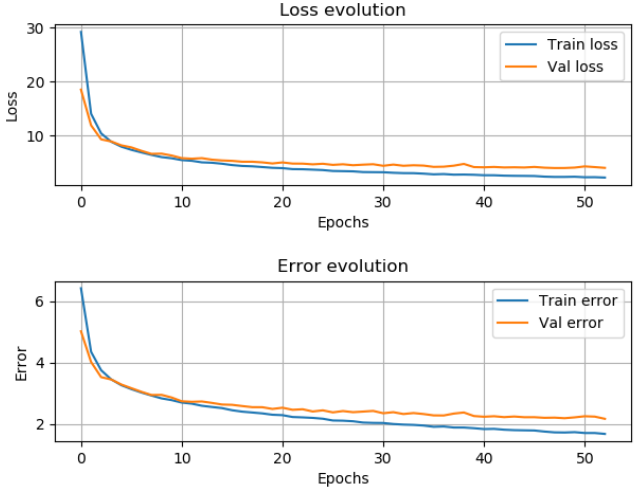

Fig. 5: Loss and error evolution after training the main model (main model + dropout(0.2)(ReLU) in table 4) We achieved test error 5.3 cm.

5 Conclusions and Future Work

Through this work, we firstly implemented the *iTracker* network as defined in [1], while we also tried some modifications to further optimize the accuracy performance of our network. Our network achieves predictions of the gaze position with little deviations from the ground truth. This performance can be further improved by extending our work in other key points and testing our ideas through simulations. We intend to try the following:

- As suggested in [20], an alternate implementation can contain both Adam optimization in the first stages of the training process, which will be later switched to mini-batch Gradient Descent in order to improve our network in terms of generalisation.
- The AlexNet architecture can be substituted by the ResNet[21] architecture in order to test if altering the main architecture of the network while keeping the same training techniques will optimize the quality of the results derived.
- Fine tuning in the hyperparameters (e.g the number of convolutional layers) of both AlexNet and ResNet architectures in order to define the optimal values that construct a strong model with good generalisation in the real world.

References

1. Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., Torralba, A.: Eye tracking for everyone. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS'12, USA, Curran Associates Inc. (2012) 1097–1105
3. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '14, Washington, DC, USA, IEEE Computer Society (2014) 580–587
4. Zeiler, M., Fergus, R.: Visualizing and understanding convolutional networks. European Conference on Computer Vision(ECCV) **8689** (01 2013) 818–833
5. Guestrin, E., Eizenman, M.: General theory of remote gaze estimation using the pupil center and corneal reflections. **53** (07 2006) 1124 – 1133
6. Lai, C.C., Chen, Y.T., Chen, K.W., Chen, S.C., Shih, S.W., Hung, Y.P.: Appearance-based gaze tracking with free head movement. (12 2014) 1869–1873
7. Wang, Y., Shen, T., Yuan, G., Bian, J., Fu, X.: Appearance-based gaze estimation using deep features and random forest regression. Knowledge-Based Systems **110** (2016) 293 – 301
8. TźSér, Z., Rill, R.A., Faragó, K., Jeni, L.A., LźRincz, A.: Personalization of gaze direction estimation with deep learning. In: 39th Annual German Conference on AI on KI 2016: Advances in Artificial Intelligence - Volume 9904, New York, NY, USA, Springer-Verlag New York, Inc. (2016) 200–207
9. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: Appearance-based gaze estimation in the wild. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2015) 4511–4520
10. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15** (2014) 1929–1958
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
12. Sun, Y., Chen, Y., Wang, X., Tang, X.: Deep learning face representation by joint identification-verification. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. NIPS'14, Cambridge, MA, USA, MIT Press (2014) 1988–1996
13. Sugano, Y., Matsushita, Y., Sato, Y.: Learning-by-synthesis for appearance-based 3d gaze estimation. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '14, Washington, DC, USA, IEEE Computer Society (2014) 1821–1828
14. qun Xu, L., Machin, D., Sheppard, P., Heath, M., Re, I.I.: A novel approach to real-time non-intrusive gaze finding (1998)
15. Baluja, S., Pomerleau, D.: Non-intrusive gaze tracking using artificial neural networks. In Cowan, J.D., Tesauro, G., Alspector, J., eds.: Advances in Neural Information Processing Systems 6. Morgan-Kaufmann (1994) 753–760
16. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. **86** (12 1998) 2278 – 2324

17. Zhang, X., Sugano, Y., Fritz, M., Bulling, A.: Its written all over your face: Full-face appearance-based gaze estimation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Volume 00. (July 2017) 2299–2308
18. Lecun, Y.: Efficient object localization using convolutional networks. In: Computer vision and pattern recognition (CVPR 2015). (2015)
19. Hansen, D.W., Ji, Q.: In the eye of the beholder: A survey of models for eyes and gaze. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(3) (March 2010) 478–500
20. Shirish Keskar, N., Socher, R.: Improving generalization performance by switching from adam to sgd. (12 2017)
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 770–778