The Eye and Pupil Tracking and Segmentation with Gaze Estimation using RGB images.

1st Bolatov Aldiyar
Nazarbayev University
Nur-Sultan, Kazakhstan
aldiyar.bolatov@nu.edu.kz

2nd Tassov Timur
Nazarbayev University
Nur-Sultan, Kazakhstan
timur.tassov@nu.edu.kz

3rd Baiturov Agakhan
Nazarbayev University
Nur-Sultan, Kazakhstan
agakhan.baiturov@nu.edu.kz

4th Rayev Asset
Nazarbayev University
Nur-Sultan, Kazakhstan
asset.rayev@nu.edu.kz

***Abstract*—This paper considers the method of the eye and pupil tracking using RGB images. Also, we estimate the gaze direction. The project consists of several parts, including training the model to detect the eye region on the face and then using cropped image detect pupil and iris with estimated gaze vector. While the first part was not so effective, the second part yielded very powerful results. Several datasets and models are used such as UnityEyes, CelebFaces Attributes containing more than 50000 elements, that allow getting the most accurate, but not the fastest results. For detection eye region we used MobileNet backbone and for gaze + iris + pupil tracking we modified the ViT network.**

***Index Terms*—Deep Learning; Eye Tracking; Gaze estimation; Eye Region Detection; Face Detection; UnityEyes; CelebFaces Attributes; EfficientNet;**

I. Introduction

Eye tracking is detection of gaze point(point at where the eye is looking at), or the motion of it relative to the head. This process can be implemented in several ways: by using the attachment to the eye, that has a particular eye lens with a magnetic or mirrored sensor that allow for particularly precise measurements of the eye gaze direction. The tightly attached contact lenses have been used by the researchers for its accurate results [1]. Another method for eye gaze detection is using the non-contact optical tools to detect the eye gaze, such as on-screen attachments, or some other tools that utilize the corneal reflection methods [2]. In this project, we will utilize the method of eye gaze detection by using the information from the video camera, allowing us to achieve sufficient accuracy without the need to buy additional equipment, and allowing us to avoid the uncomfortable feeling that is brought by using the lens for the purpose of eye tracking. To estimate all the points and values, we used deep learning techniques, which are very powerful at image tasks. The source code for this paper will be submitted together with this paper.

II. Training Samples

A. UnityEyes

UnityEyes was first presented by the students of University of Cambridge. Coded on Unity and using the simple k-Nearest-Neighbor approach for gaze estimation, the software is focused on creating 3D models of eyes. On a medium machine, it is very time efficient and consumes little resources to perform its task. Generating and rendering at the same time, the method saves two types of files, including jpg image of a generated and json file. Those are then used to train a gaze estimation algorithm. The dataset generated by UnityEyes consisted of 117968 files, which makes 53984 3D rendered eye region images.

B. CelebFaces Attributes (CelebA)

CelebFaces Attributes (CelebA) Dataset was designed as training data for face identification across different platforms. What is unique about this dataset is that it is great when dealing with the recognition of separate facial attributes [3]. Overall it contains around 202,000 images of different celebrities with 5 different landmarks (eyes, nose and mouth sides) and over 10000 identities. It is worth mentioning that the creators of the CelebA cropped and aligned images in a separate folder. Additionally, there is a file containing coordinates for the face bounding box, where we can extract two opposite corners. To make sorting easy, CelebA offers 40 binary attributes for each image, where 1 represents trait and -1 represents lack of that trait. This dataset is used for eye localization.

III. Methods

The main aim is to have a balance between the performance of the whole system and inference speed. The importance of rate per second is underlooked, but it is necessary to have a high frame per second processing speed due to its contribution to overall usability and pleasantness.

The first subtask is to localize the eye region on the face on the overall image. For the speed, we compress the input image to very small sizes, and still we as a human will see an approximate face region. Thus, the machine can also; means we can use compressed images for a much lower computation. Furthermore, the structure of the neural network and a number of its parameters also significantly contribute to the speed. To achieve that goal, we used MobileNetV3 backbone, which is small, fast and effective, which look like the best fit between speed (number of parameters) and accuracy. We trained our model on the CelebA dataset presented above. The problem that we encountered was the lack of robustness in pre-existed models such as input shape. Even if pre-existing models demonstrates better results, we still made a progress because we achieved one of our goals.

However, we also could use for this, we could used either direct training on data, which has eye coordinates, or

landmarks dataset and extracting only landmarks for eyes. We chose using CelebA dataset with landmarks to save time, because training model took significant amount of time due to technical capabilities of our hardware.

The last part of our work was eye-region. We used the UnityEyes utility, which is presented above. With this tool, we generated data by our own and did not search for datasets. We also pursued to set different labels, such as eye direction. After training on a vast database, we got an accurate sub-model, but with a lack of speed. To overcome this problem. We assumed that the data-hungry transformer and unlimited dataset is the best match, which we used in terms of vision transformer.

A. MobileNetV3

MobileNet is an architecture that uses depth-wise separable convolutions that allow to create lightweight deep neural networks. The MobileNet architecture are mostly optimized for use in mobile and embedded vision applications. The architecture provides two hyper-parameters that allow to find balanced solution between accuracy and speed based on the scope of the problem. In the paper written by Howard et.al. [4], even the first version of the architecture shows stronger performance when compared to other popular models based on ImageNet classification. The next generation of MobileNet architectures, named MobileNetV3, that has been separated into two new models MobileNetV3-Large and MobileNetV3-Small, has been tuned for mobile CPUs through the usage of combination of hardware-aware network architecture search (NAS), with the help of NetAdapt algorithm with minor tuning through novel architecture improvements. In their work, Howard et.al. [5] provide a new efficient segmentation decoder named Lite Reduced Atrous Spatial Pyramid Pooling, with achieving result of 3.2% increased accuracy with reduced latency by 20% when compared to the previous generation of MobileNet in the MobileNetV3-Large model. At the same time, MobileNetV3-Small managed to achieve 6.6% more accuracy with comparable latency when compared to the MobileNetV2 model.

B. Visual Transformer

The transformer architecture previously surpassed the RNN-based models in natural language task (NLP) and became the standard model for text-related problems [6]. However, in the computer vision domain, transformer networks were only used with convolutional layers. The reason was to make the overall convolutional neural network (CNN) more expressive [7]. However, only recently, it was shown, that reliance on convolutional layers is not essential [8]. Dosovitskiy et al., in their work, presented a pure transformer model that uses sequences of image patches as an input. They heavily pre-trained the model, and then by transfer learning, they mapped the pretrained transformer model to image recognition benchmarks like ImageNet, and CIFAR [8]. Their vision transformer achieved comparable, with the state-of-art, results while consuming fewer computational resources during training.

Benefits of transformer architectures are their computational efficiency, lesser number of hyperparameters to tune, and scalability [8]. This lead to a lack of saturating performance with an increase in the model and dataset.

However, eye tracking and gaze estimation have some dissimilarities. For example, we do not exactly classify but more regress on the data. Thus we added some changes to the model, such as convolution bottleneck feature extractor to map it to heatmaps. For example, to differentiate between two different animals, it is enough to "understand" animals' shape and color. Eye and gaze tasks are a little bit trickier and have more underlying complexities such as direction.

Furthermore, pure vision patch transformers were not tried on eye tracking and gaze estimation tasks. This leads us to the question: how satisfactory vision transformers can perform more complex vision tasks such as eye tracking and gaze vector estimation. To fill the research gap, we proposed a patched vision transformer for our task. It is possible due to the fact that we can create a dataset as big as we want and this is extremely important to the data-hungry transformers.

C. Network Architecture of ViT

The overall model architecture can be seen in figure 1. Firstly, since the transformer accepts only a one-dimensional sequence of token embedding, the input image needs to be handled. Dosovitskiy et al. proposed to divide two dimensional images $im \in R^{C\times H\times W}$ into N patches of size $patch_n \in R^{P^2C}$, where C is number of channels; H, W is stand for original image resolution; P, P is the resolution of only one patch and number of patches $N = HW/P^2$ [8]. Then patches are flattened, and N serves as an input sequence length. Afterward, every patch is mapped to some constant dimensionality using the trainable linear projection.

Further, as in BERT [9], a learnable class token is prepended to the embedded patches, which output states of the transformer is used for the classification. Also, because of the fact, those transformers are not aware of the spatiality of the information, learnable positional embedding was augmented into every patch embeddings. In turn, this helped to keep positional information.

At the end we get a sequence x of length $N + 1$, where each sequence element is $x_n \in R^D$ and encode spatial information. This sequence is then fed into a six-layer transformer network. The transformer network itself consists of two main layers: multi-head self-attention (SA) and feed-forward neural network (FF).

The self-attention function is a process of quantifying the representation of the relative importance of each sequence element. Here, the self-attention mechanism relates each embedded patch with all other patches of the sequence. Then, this new information and relative importance added to the value representation [6]. This is done using the compatibility function. In our work, we used scaled dot-product because our input sequence length N was relatively low. Different functions are focused on the optimizing quadratic complexity concerning the sequence length [10]:

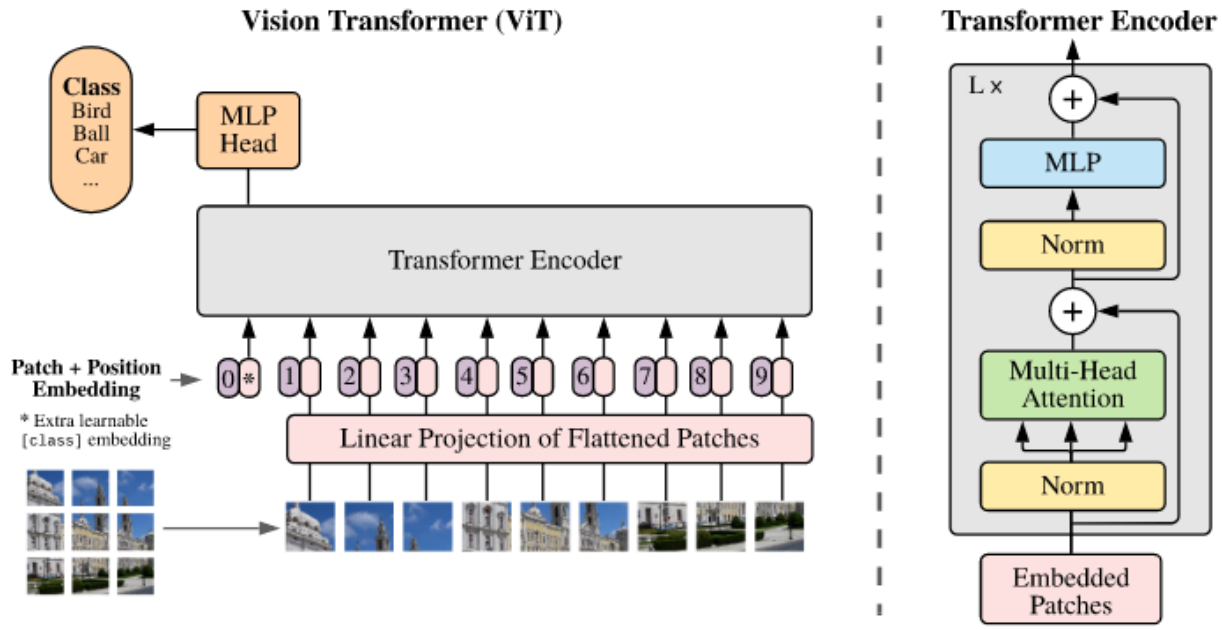


Fig. 1. The Model Structure

$$f_{scaled}(Q,K) = \frac{Q,K^T}{\sqrt{d_k}} \tag{1}$$

Then new value representation is then computed using:

$$f_{at}(Q,K,V) = V' = softmax(f_{scaled}(Q,K))V \tag{2}$$

As Vaswani et al. [6] specified, Q, K, V are three different representation of the input sequence, which were acquired using separate learned linear transformation. Usually, the self-attention mechanism is split between heads, which are then concatenated. The purpose is that each head can capture unique features by having individual parameters.

The feed-forward neural network in our work has a form of gated activation linear unit, which is followed by linear projection to the same space as input. There, swish activation function was used [11]. In the formula 3, $LN \in R^(D \times D')$ stands for learned linear projection.

$$f_{ff}(X) = LN_D(LN_{glu1}(X) \times swish(LN_{glu2}(X))) \tag{3}$$

The transformer utilizes Layernorm (LN) and residual connections, which both help to better gradient flow. The overall equations that took place in every block (t) of the transformer are presented below with multi-layer perceptron (MLP), which takes output states of the class token. We then predict our gaze vector. However, on all other processed data, we first restructure it back to the image size and then use bottleneck convolution to predict heatmap. From heatmap, we then infer important landmarks to track the eye, iris, and pupil.

$$X_t^{'} = SA(LN(X_{t-1})) + X_{t-1} \tag{4}$$

$$X_t = FF(LN(X_t^{'})) + X_t^{'} \tag{5}$$

$$class = MLP(LN(X_6^0)) \tag{6}$$

Results and Discussion

As can be seen in the result of eye detection, the performance is not good. The red region should be in the eye region which is close. However, it is still off. It could be tied to the model performance, but the reason is simpler. The celeb dataset contains the cropped face and in our example, it is not. So, to improve the performance, we need to use an additional layer that will crop the face, but due to lack of time, we did not implement it.

On the other hand, the performance of the updated visual transformer for gaze estimation is superior. The only mistake it is making is the same mistake human can make, such as in the second picture. While being powerful, the network had an adequate size of 1.8M of parameters. The network was trained on 60K+ generated images, which is good for data-hungry transformer, but could be more, but again time and compute constraints.

Future Works

From the accomplished work on this project, we can mention several things that could be done in the future in order to improve the results and expand the scope of this project. The first thing that could be done is to implement higher quality datasets that would allow for higher accuracy while training the model, potentially leading to wider scope of use

Fig. 2. Predicted accuracy of pupil direction.

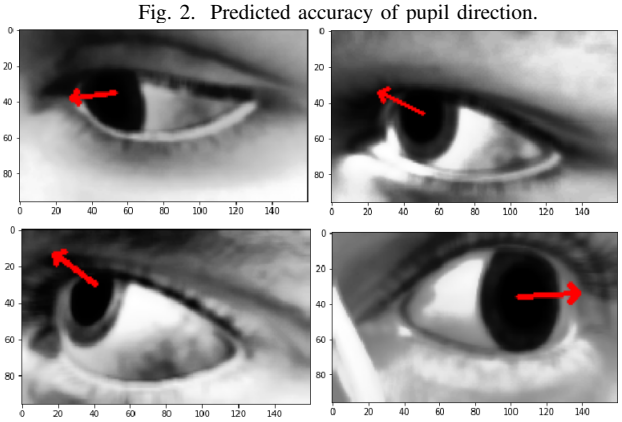
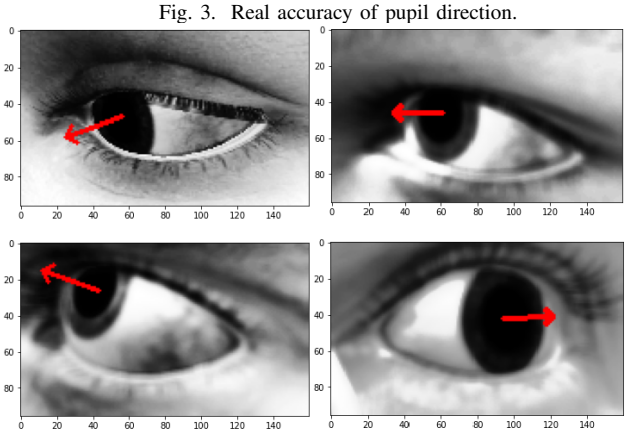
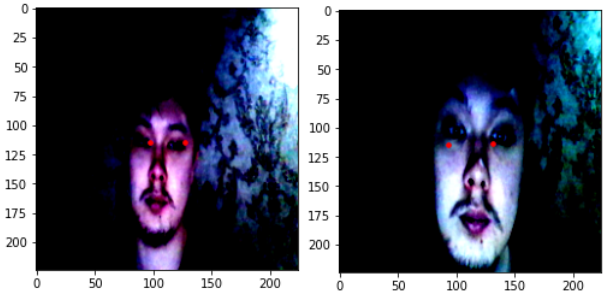


Fig. 3. Real accuracy of pupil direction.



for this project. We could also use our time to generate more than a million sample images. This will lead to very high performance. The next thing is the implementation of the face cropper for better eye localization performance. Also, all of the networks should be quantized for better speed. As the current implementation of the gaze detection is relatively robust, using more optimized functions and methods could lead to better performance in general. After these improvements, it would be possible for this project to get a more user friendly interface, potentially allowing it to be used by broader audiences. Thus, we should connect every part done effectively. Later, implementing the cross-platform version of the app, and releasing it on different platforms and their corresponding app stores, would be one of the final implementations of where this project could be implemented.

Fig. 4. Result of eye detection.



Conclusion

We considered one of the best ways of eye and pupil tracking using the gaze approximation method, by using the machine learning algorithm and several datasets, allowing us to reach satisfactory results. The machine learning model was trained by using the datasets mentioned above.

Using the MobileNet backbone, that is considered to be relatively lightweight, but still sufficient for our work, we were able to obtain a satisfactory result. Now that we have obtained the eye regions, our task was to train the Transformer Network that required a lot of data, for which we have used the UnityEyes utility. That resulted in a powerful model, which however lacked the speed, which can be easily solved by implementing the quantization and using the optimized compilers that would run the model in native C++ style.

References

[1] D. A. Robinson, "A method of measuring eye movemnent using a scieral search coil in a magnetic field," *IEEE Transactions on Bio-medical Electronics*, vol. 10, no. 4, pp. 137–145, 1963.

[2] H. D. Crane and C. M. Steele, "Generation-v dual-purkinje-image eyetracker," *Appl. Opt.*, vol. 24, no. 4, pp. 527–537, Feb 1985. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-24-4-527

[3] Z. Liu, P. Luo, X. Wang, and X. Tang, "Large-scale celebfaces attributes (celeba) dataset," *Retrieved August*, vol. 15, no. 2018, p. 11, 2018.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[5] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *arXiv preprint arXiv:2005.12872*, 2020.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[9] I. Tenney, D. Das, and E. Pavlick, "Bert rediscovers the classical nlp pipeline," *arXiv preprint arXiv:1905.05950*, 2019.

[10] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," *arXiv preprint arXiv:2006.16236*, 2020.

[11] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

ADDITIONAL INTEREST

On the left, you can see a normal heatmap, which is produced from the landmark. In our network, we also tried to predict heatmap and then use it to predict landmarks back. However, the heatmap that is generated by the network is very intricate and strange. Probably, the network found it is own way to communicate heatmap. It can be researched more.

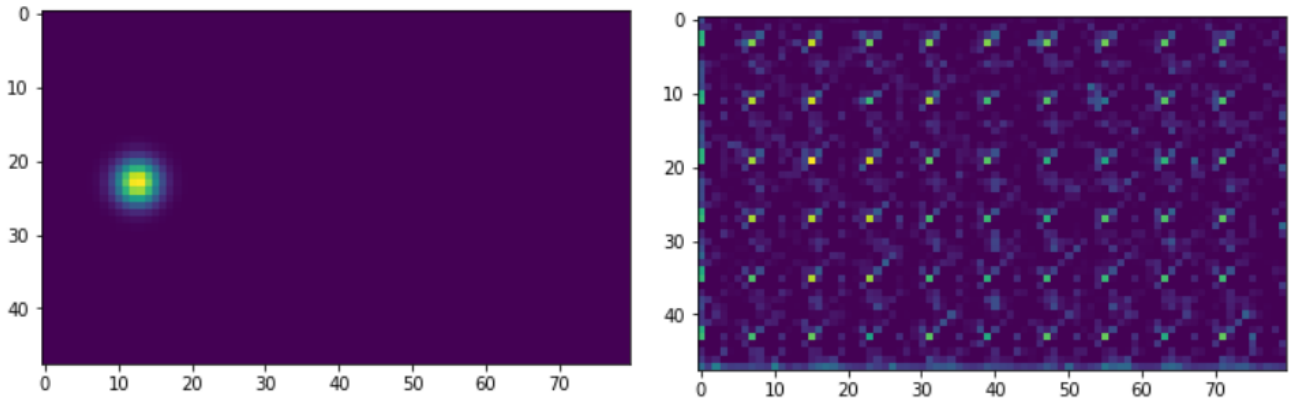


Fig. 5. Result of eye detection.