Final Project

Teng Liu, RIN: 661088506

1 Explanation and Justification

1.1 Introduction and Explanation

In this project, I designed a deep model for estimating the position of eye gaze on the phone/tablet screen. Specifically, the model consists four paths: three paths of deep convolutional neural network (CNN), and one path of deep neural network. The convolution layers downsize the data, extract the features, and share the weights. The fully connected layers follows the traditional neural network structure. There are four inputs: left eye, right eye, face, and face mask, each of which is fed into the corresponding path in the model. The output is a 2-dimensional coordinate, so the cost function is chosen to be the mean square error of the Euclidean distance from the true eye-gaze position. Fig. 1 shows the dimension and related operations of each layer. Detailed TensorBoard architecture visualization is shown in Section 3.

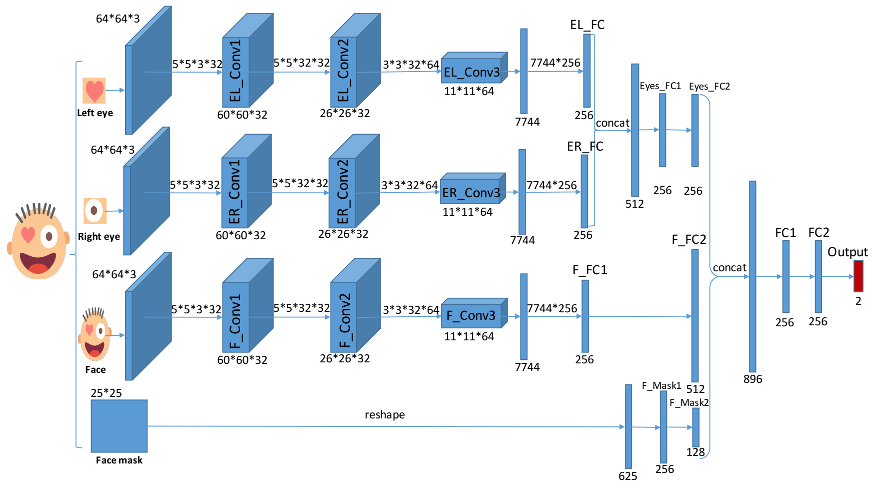


Figure 1: General architecture of the model. All convolution layers have the stride size of 1. After every convolution layer, there is a ReLU operation, followed by a max pool operation with size 2 and stride 2. The input image is for illustration only, and is not the real data.

1.2 Justification

I justify the above design as follows. Since we are provided with four inputs, taking use of all of them is a natural first attempt. After trying different architectures, including one path and two paths, four-path model indeed performs the best. Detailed architecture comparison can be found in Section 4. The input images are normalized by dividing 255, subtracting the mean, and dividing the standard deviation. This normalization is written in the first function in the submitted python code. Following the discussion in [1], I removed the last convolution layer from the left-eye path, the right-eye path, and the face path. This is because the images provided are 64*64 instead of the original 224*224. The model in [1] will result in a much smaller dimension in the fully connected layer, given the smaller image size. I also added the fully connected layer to the eye paths, and a fully connected layer before the output layer. This is to compensate for the removal of the convolution layers, which may simply the model too much and may jeopardize the performance. In order to convolve with as many pixels as possible without downsizing too much, all filters used in the convolution layers have the stride size of 1. To add non-linearity, ReLU is used as the activation function after each convolution layer. Max pool is applied to the output of ReLU to further downsize the data while keeping the important features.

2 Loss and Error Plot

The batch size is set to 512. The initial learning rate is set to 0.001. After each epoch, the learning rate is decreased by multiplying 0.85, which I referred to as the decay parameter. This adaptive learning rate results in a faster convergence and a lower test error, as shown in Fig. 2. After about 30 epochs, the performance converges. The final loss, train error, and test error is 0.9818, 1.1211 cm, and 1.8496 cm respectively (Please see the terminal logs in the attachments for detailed data).

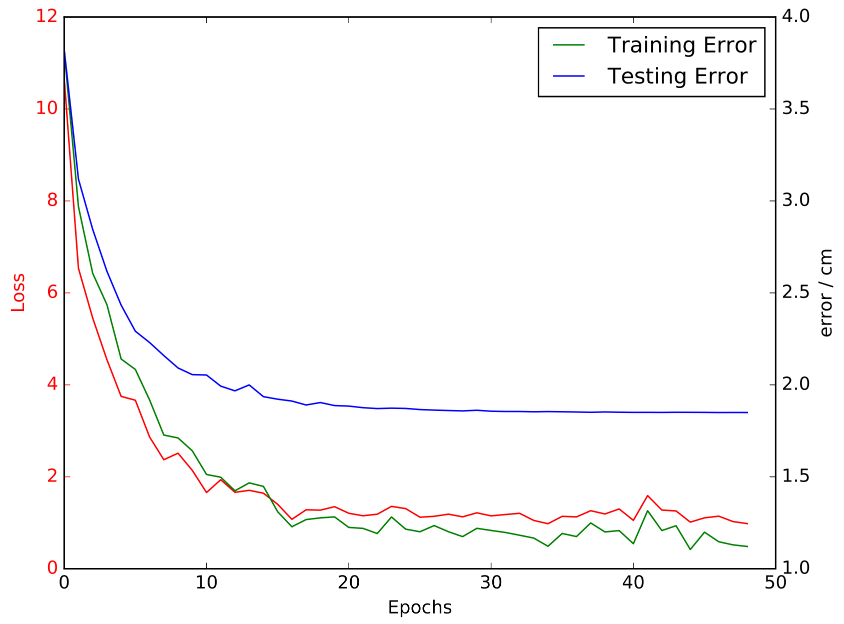


Figure 2: Batch size = 512. Learning rate $\eta = 0.001$. Decay parameter is 0.05. The performance converges after around 30 epochs. After 50 epochs, the loss, train error, and test error is 0.9818, 1.1211 cm, and 1.8496 cm respectively.

3 Model Architecture Visualization

TensorBoard is used to visualize the architecture of the model. Since I used *tf.get_variable()* to initialize the weights, the weight tensors are also shown on the graph. To make the architecture more clear, I removed the weight tensors from the main graph in Fig. 3. The output layer has two branches: the left branch corresponds to the loss calculation, and the right branch corresponds to the error calculation.

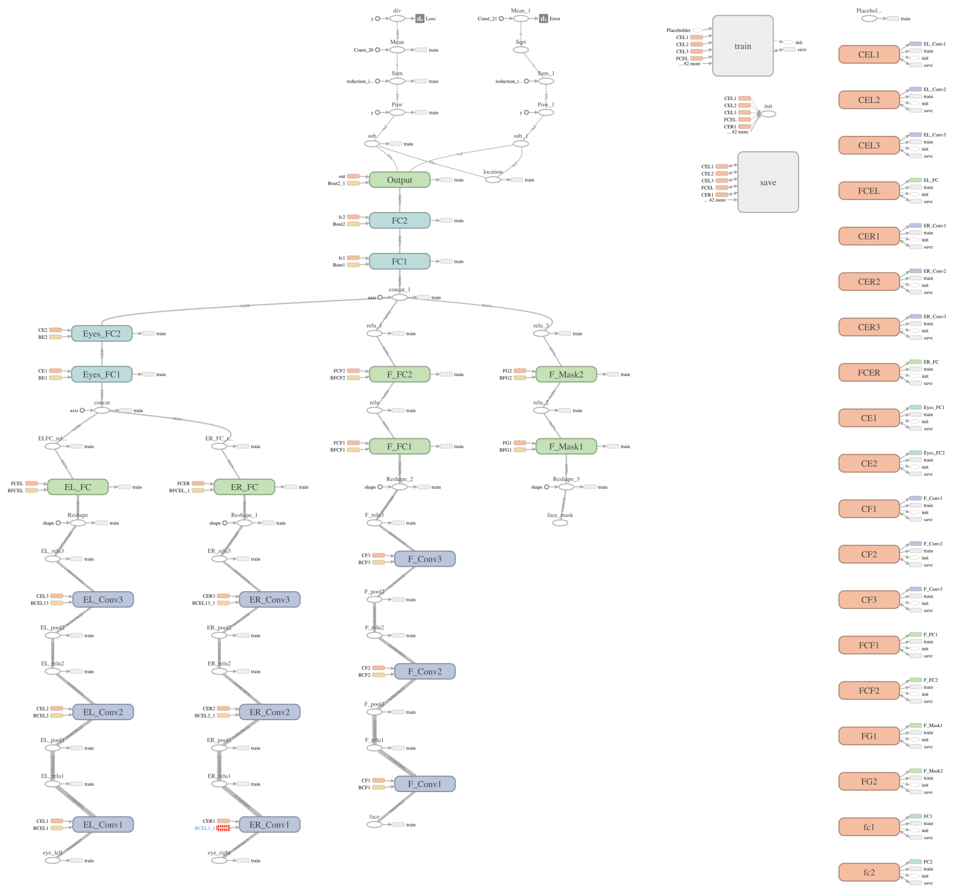


Figure 3: Four-path Model architecture visualization using TensorBoard.

4 Architecture Comparison

I compare the four-path model with the two-path model and the one-path model. In the two-path model, only the images of the left eye and the right eye are fed into the model, while the face images and the face masks are discarded, as shown in Fig. 4. In the one-path model, only the images of the left eye are fed into the model, as shown in Fig. 5.

To check which architecture performs the best and to determine the value of the hyper-parameter at the same time, I applied the coordinate optimization algorithm. Specifically, I tested three learning rate 0.0002, 0.001 and 0.005 on one-path model, two-path model and four-path model, and use the TensorBoard to visualize the training error and the loss, as shown in Fig. 6 and Fig. 7. Fig. 8 shows the log scale of the loss, where the horizontal axis is chosen as "Wall" instead of "Step", which is more clear than Fig. 7.

From the comparison figures, we note that the loss and the error of the four-path model with the learning rate 0.001 not only decreases the fastest, but also converges to a better performance. The two-path model with the learning rate 0.001 ranks the second in terms of performance. With less paths, both the one-path model and the two-path model have a higher error and a higher loss than the four-path model at any iteration. However, all of the considered architectures have reasonably good performance with about 1 cm of difference in the error after convergence. Therefore, all of them are applicable based on the size of the data. If the data size is relatively large, the four-path model can be used since it is more complex than the other two models. If the data size is relatively small, the one-path model can be used in order to prevent over-fitting. Otherwise, the two-path model is a good choice to use.

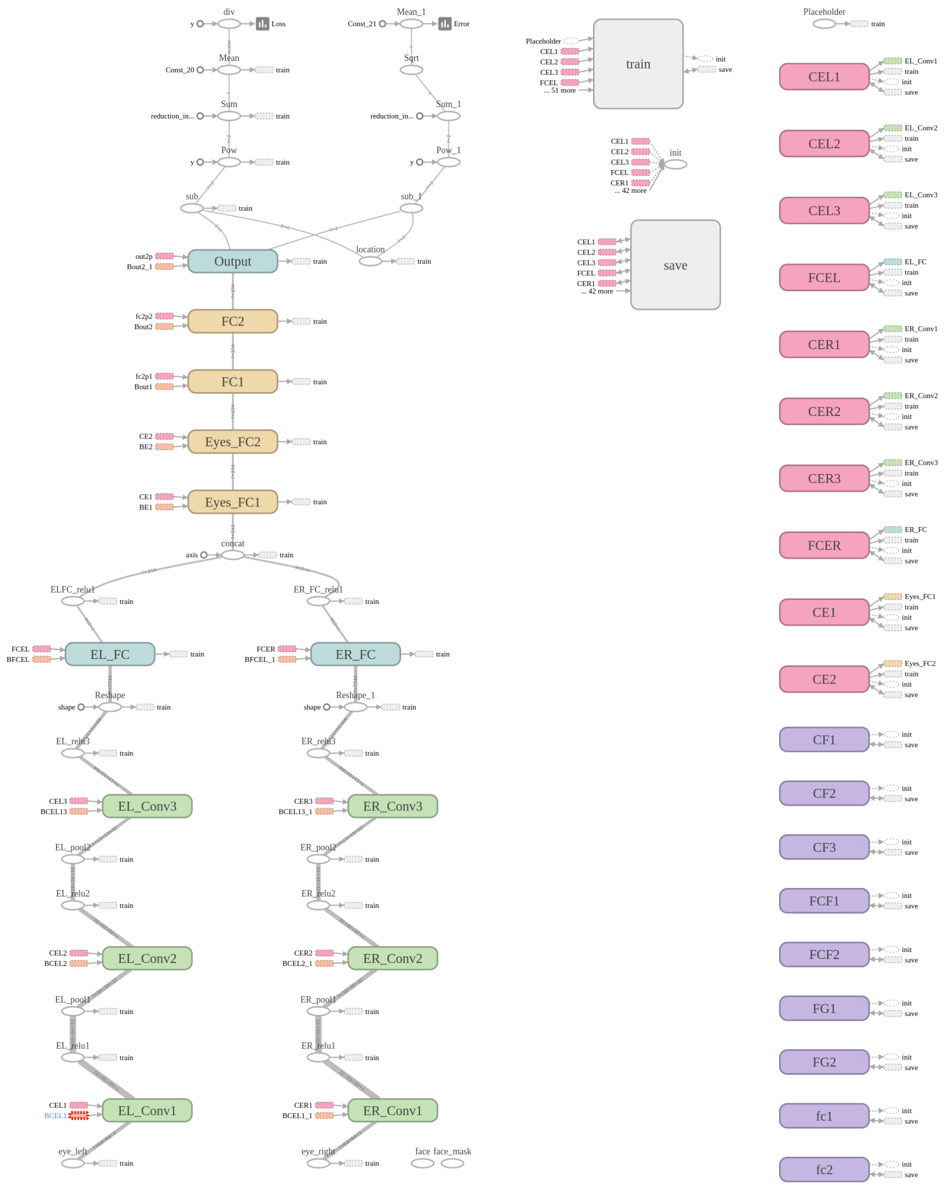


Figure 4: Model with two paths.

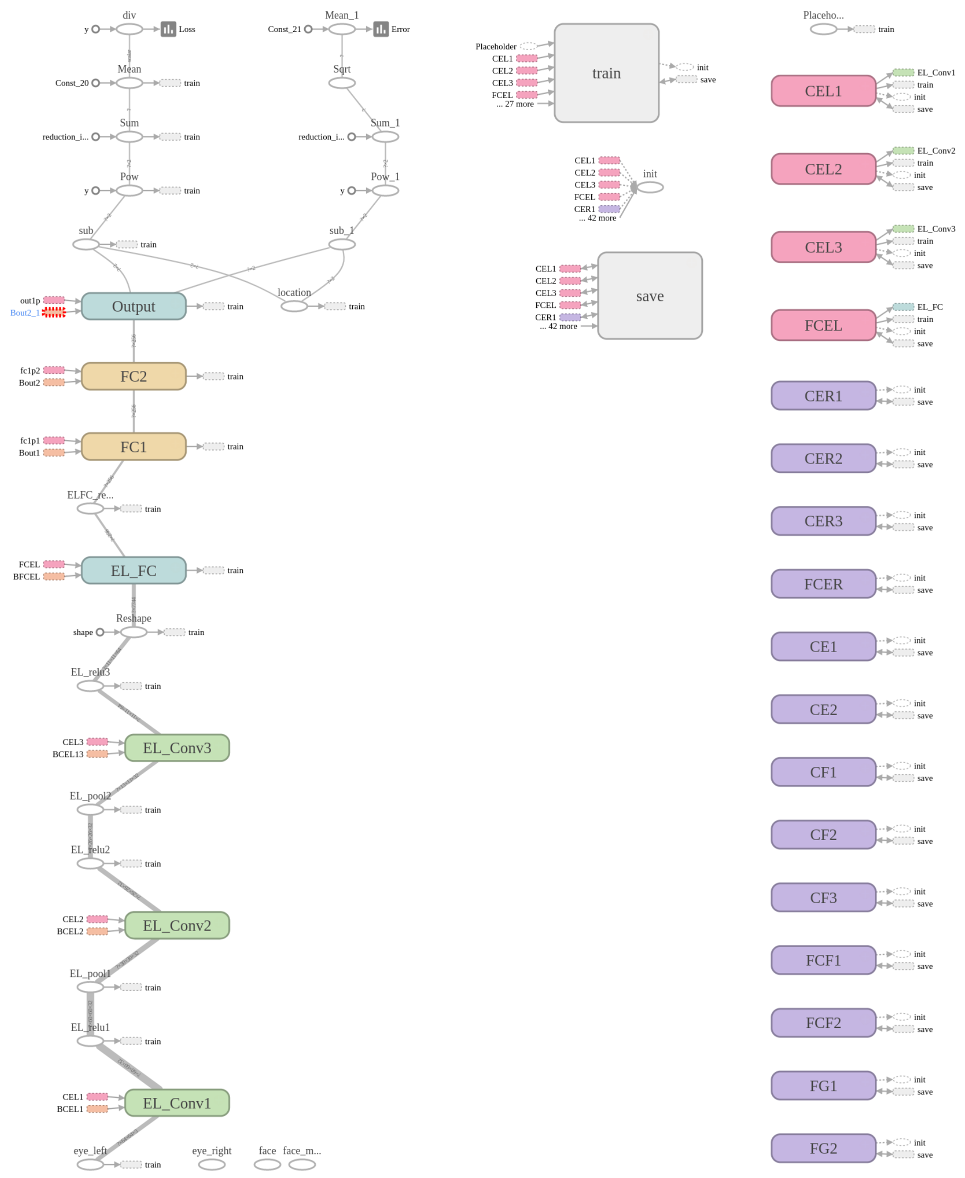
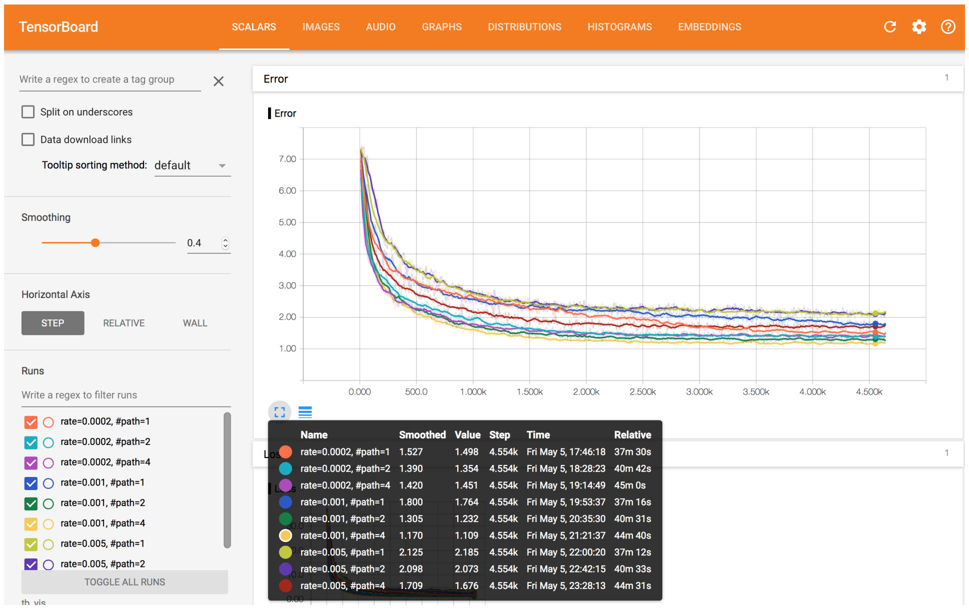
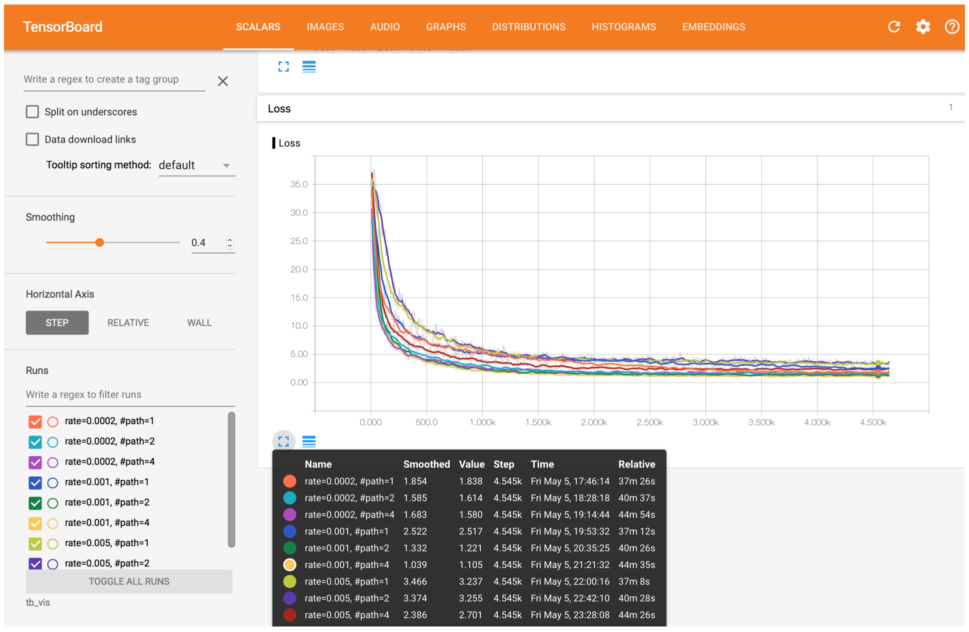


Figure 5: Model with one path.



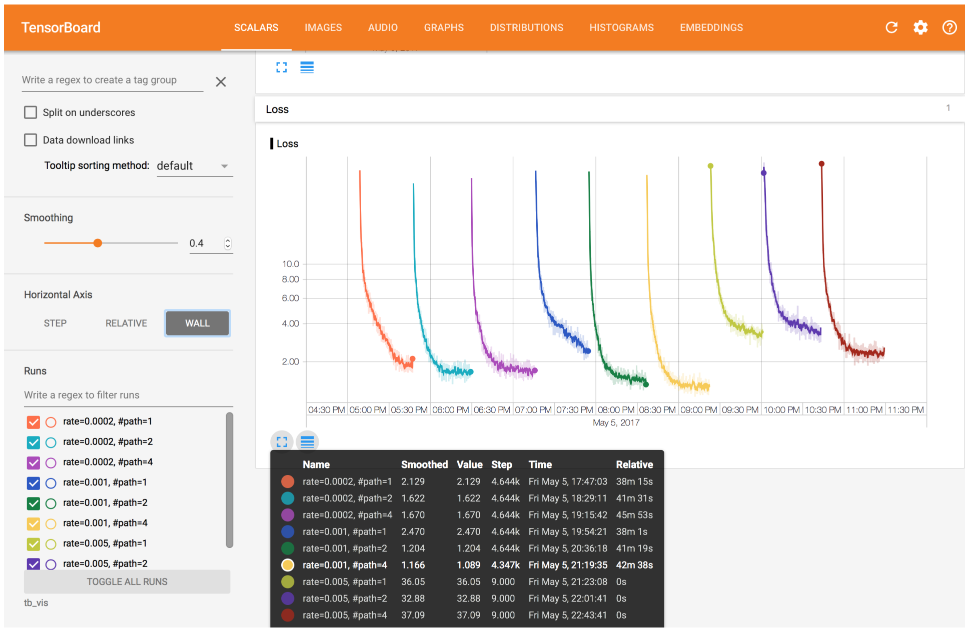
	Name	Smoothed	Value	Step	Time	Relative
●	rate=0.0002, #path=1	1.527	1.498	4.554k	Fri May 5, 17:46:18	37m 30s
●	rate=0.0002, #path=2	1.390	1.354	4.554k	Fri May 5, 18:28:23	40m 42s
●	rate=0.0002, #path=4	1.420	1.451	4.554k	Fri May 5, 19:14:49	45m 0s
●	rate=0.001, #path=1	1.800	1.764	4.554k	Fri May 5, 19:53:37	37m 16s
●	rate=0.001, #path=2	1.305	1.232	4.554k	Fri May 5, 20:35:30	40m 31s
●	rate=0.001, #path=4	1.170	1.109	4.554k	Fri May 5, 21:21:37	44m 40s
●	rate=0.005, #path=1	2.125	2.185	4.554k	Fri May 5, 22:00:20	37m 12s
●	rate=0.005, #path=2	2.098	2.073	4.554k	Fri May 5, 22:42:15	40m 33s
●	rate=0.005, #path=4	1.709	1.676	4.554k	Fri May 5, 23:28:13	44m 31s

Figure 6: Comparison of the training error among different architectures and different learning rates. X-axis is "Step".



	Name	Smoothed	Value	Step	Time	Relative
●	rate=0.0002, #path=1	1.854	1.838	4.545k	Fri May 5, 17:46:14	37m 26s
●	rate=0.0002, #path=2	1.585	1.614	4.545k	Fri May 5, 18:28:18	40m 37s
●	rate=0.0002, #path=4	1.683	1.580	4.545k	Fri May 5, 19:14:44	44m 54s
●	rate=0.001, #path=1	2.522	2.517	4.545k	Fri May 5, 19:53:32	37m 12s
●	rate=0.001, #path=2	1.332	1.221	4.545k	Fri May 5, 20:35:25	40m 26s
●	rate=0.001, #path=4	1.039	1.105	4.545k	Fri May 5, 21:21:32	44m 35s
●	rate=0.005, #path=1	3.466	3.237	4.545k	Fri May 5, 22:00:16	37m 8s
●	rate=0.005, #path=2	3.374	3.255	4.545k	Fri May 5, 22:42:10	40m 28s
●	rate=0.005, #path=4	2.386	2.701	4.545k	Fri May 5, 23:28:08	44m 26s

Figure 7: Comparison of the loss among different architectures and different learning rates. X-axis is "Step".



	Name	Smoothed	Value	Step	Time	Relative
●	rate=0.0002, #path=1	2.129	2.129	4.644k	Fri May 5, 17:47:03	38m 15s
●	rate=0.0002, #path=2	1.622	1.622	4.644k	Fri May 5, 18:29:11	41m 31s
●	rate=0.0002, #path=4	1.670	1.670	4.644k	Fri May 5, 19:15:42	45m 53s
●	rate=0.001, #path=1	2.470	2.470	4.644k	Fri May 5, 19:54:21	38m 1s
●	rate=0.001, #path=2	1.204	1.204	4.644k	Fri May 5, 20:36:18	41m 19s
●	**rate=0.001, #path=4**	**1.166**	**1.089**	**4.347k**	**Fri May 5, 21:19:35**	**42m 38s**
●	rate=0.005, #path=1	36.05	36.05	9.000	Fri May 5, 21:23:08	0s
●	rate=0.005, #path=2	32.88	32.88	9.000	Fri May 5, 22:01:41	0s
●	rate=0.005, #path=4	37.09	37.09	9.000	Fri May 5, 22:43:41	0s

Figure 8: Comparison of the loss among different architectures and different learning rates. Y-axis is in log scale. X-axis is ”Wall”.

References

[1] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2176–2184, 2016.