Deep learning Final Project

Gyanendra Sharma

May 10, 2017

1 Architecture and Model explanation

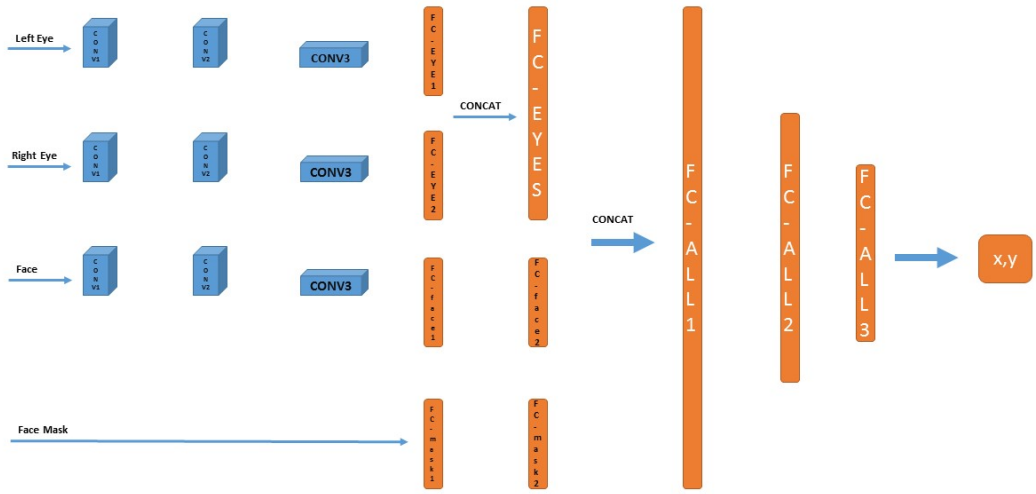

Figure 1: High level architecture diagram of the Deep network for mobile eye gaze estimation.

The architecture that I designed for this project is shown in figure 1. Three inputs; left eye, right eye and face go through three layers of convolution layers first whereas the face mask input directly goes to the fully connected neural layer. I used 32 filters of size 5x5 for the first convolution layer for each of the three inputs. The second convolution layer was also composed of 32 filters with size of 5x5 and the final convolution layer was 64 filters each of size 3x3. The left eye and right eye share the same weights for the convolution layers. However, the face has independent weights even though the shape of the layers is similar to that of eyes. A more detailed architecture based on **tensorboard** is shown in figure 4.

Each of the left eye and right eye go through a fully connected layer before the outputs get concatenated. The output of the convolution layer 3 is a tensor of size 11x11x64. So, we flatten this tensor and feed it to the fully connected neural network which outputs a tensor of shape 256x1. Once this is applied to both eyes, the outputs are concatenated and fed to another fully connected layer which produces a output of tensor 256x1 from the input tensor of shape 512x1.

The output of convolution layer 3 from the face input also goes through two fully connected layers independently. The two hidden layers are both of size 256 nodes thus, producing an output tensor of shape 256x1 after the operation.

The mask layer input is of size 25x25 which is flattened and fed to the first fully connected layer of

size 256. The output of this layer is fed to another fully connected layer of size 128, which produces and output tensor of shape 256x1.

Once all the individual inputs are processed separately, as explained above and in the diagram, all inputs are concatenated to produce an output tensor of shape 2x256 + 128 = 640x1. This output is fed through a fully connected neural net architecture with two hidden layers of size 512 and 256 consecutively. The final out layer has two nodes corresponding to two classes of output i.e x and y.

2 Error and Loss Plots

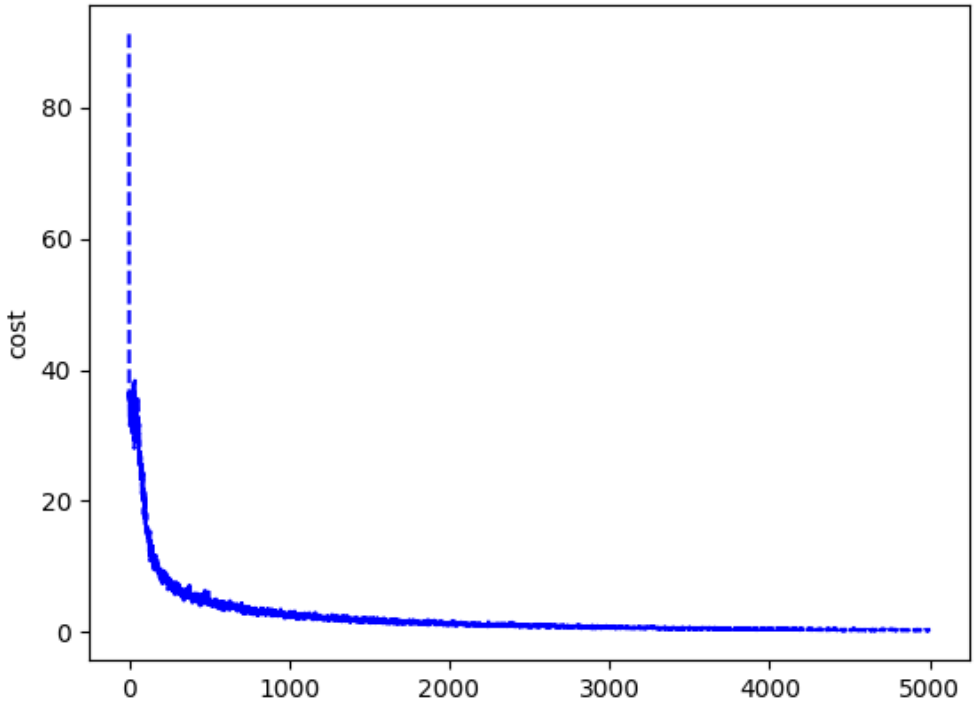

Figure 2: Loss over 5000 iterations. Batch size of 512.

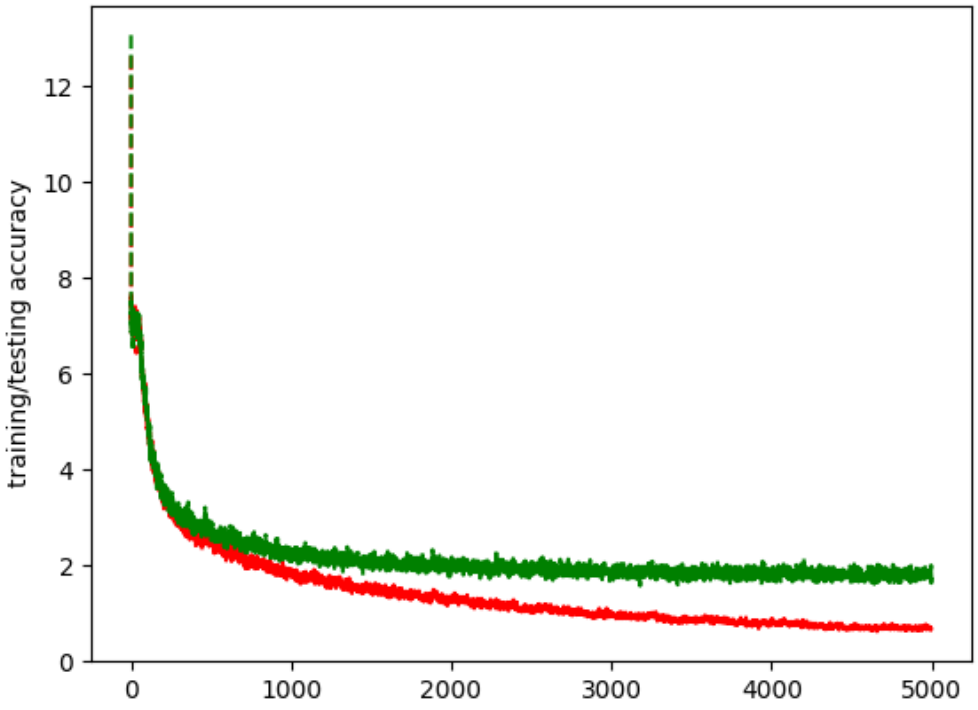

Figure 3: Training and testing accuracy over 500 iterations. Batch size of 512. **Green:Test Accuracy, Red:Training Accuracy**

Here, as seen in figure 2 and 3, we see that the loss values as well as the training and testing error values converge pretty well over time. The Final validation error for my model was **1.79cm.**

3 Tensorboard Visualization

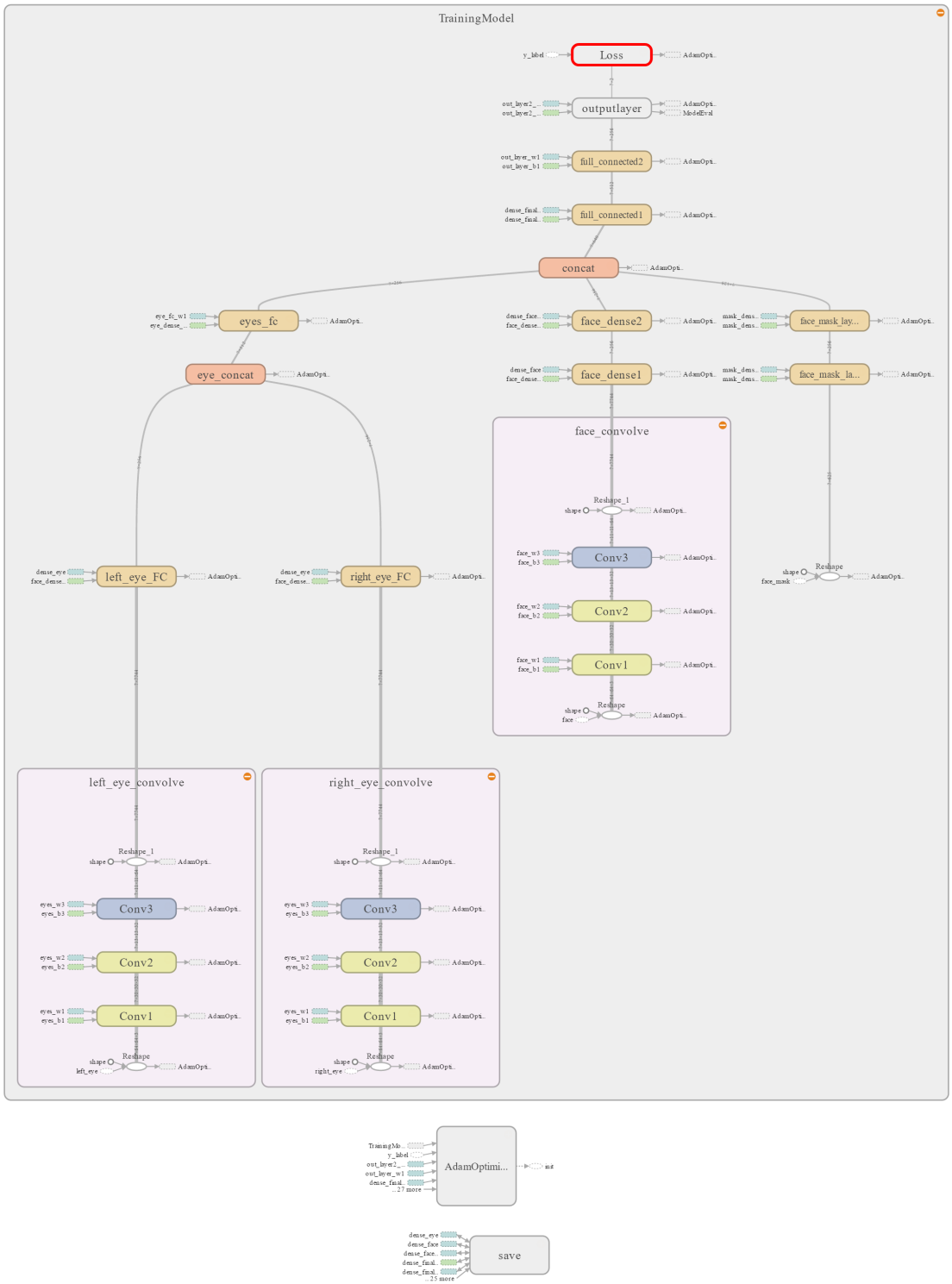

Figure 4: Tensorboard visualization of the deep network architecture for eye gaze estimation.

Note: I am submitting an accompanied file *tensorboardVis.png*, if you want to zoom and look at more details such as the tensor shapes for different inputs/outuds.

4 Architecture/Different pathway comparison

The model I designed is based on the paper given in the assignment. However, different pathway was chosen based on the results I got in designing the architecture.

My initial architecture had 4 convolution layer for each of the left eye, right eye and face inputs. In addition, I only had one fully connected layer after the convolution operations for each individual inputs. Mask input was also passed through a single fully connected layer. Once all inputs were concatenated, I had a single hidden layer and one output layer to get the output x and y values. My hope was that with more convolution layers, the information from left eye, right eye and the face would be preserved and weighted more than the mask layer. However, having very few fully connected layer resulted in loss of information through the layers and thus my output suffered considerably.

So, I decreased the number of convolution layers to three and added fully connected neural networks for both the individual inputs as well as the concatenated outputs. Adding fully connected layers throughout the network made sure that information wasn't lost while moving through the networks and thus ended up improving my result considerably.