Final Project: A CNN for Eye Tracking
ECSE 6965: Deep Learning
Andrés Vargas

1 Introduction

In this project, I implemented a deep convolutional network for eye tracking in Tensorflow. The network architecture was inspired (but not copied from) [1]. I used Tensorboard to visualize the network structure and iteration-wise error. My network achieved a validation error of 1.673 cm, which is better than the state of the art acheived in [1].

2 Data and Hardware

The dataset contains 48000 training samples and 5000 validation samples. Each sample contains 5 features: face, left eye, right eye, face mask, and labels. The network, shown in the next section, requires extensive computational power. For this reason, I ran the training routine on a remote server to which I had access through my research group at RPI.

3 Architecture and Results

Figure 1 shows the network architecture. Operations that begin with a "C" are convolutions, "P" is for pooling, "D" is for densely connected layers, and "F" (these nodes are smaller) stands for flatten. The characters after the first character indicate channel. For instance, "PRE" stands for "pool right eye" and "DM" stands for "dense mask". Notice that the eyes each have 3 convolutional layers, with each convolution followed by a 2×2 max pool with stride of 2. The convolution layers have filter sizes 5×5, 3×3, and 3×3, with number of filters 18, 36, and 72; respectively. The face has two convolutions, each followed by a 2×2 max pool with stride 2, with filter sizes and number of filters equal to those of the first two eye channels. All filters in this network use a stride of 1.

After the eye convolutions, the outputs are flattened (FRE and FLE) and then concatenated (FE). The face convolution output is also flattened (FF). Then, FE and FF are each passed through two densely connected layers of size 1000, each one with a RELU activation. The mask never gets convolved, but instead is immediately flattened (FM) and then passed through two densely connected layers of size 625, each one with a RELU activation. DE1, DF1, and DM1 (eye, face, and mask channels; respectively), are then concatenated and passed through another size 1000 densely connected layer with dropout probability of 0.5 and a RELU activation. From there, we linearly map the (batchsize x 1000) dimensional "final_flat" to the (batchsize x 2) dimensional "pred". The error is defined as the batch mean of the norm of the difference between pred and the ground truth. For learning, I used Adam Optimizer with a step size of 0.001 and a batch size of 500, and minimized the Euclidean error. The results can be seen in Figure 2.

Since the batch size was 500, each epoch was comprised of 96 iterations. Figure 2 shows that as the number of epochs increases, the validation error becomes progressively larger than the training error because the network has seen the training data over and over again. Nevertheless, dropout probability of 0.5 in the final densely connected layer did a good job of preventing significant overfitting. At the end of the 25th epoch, the training error was 1.132cm and the validation error was 1.673cm.

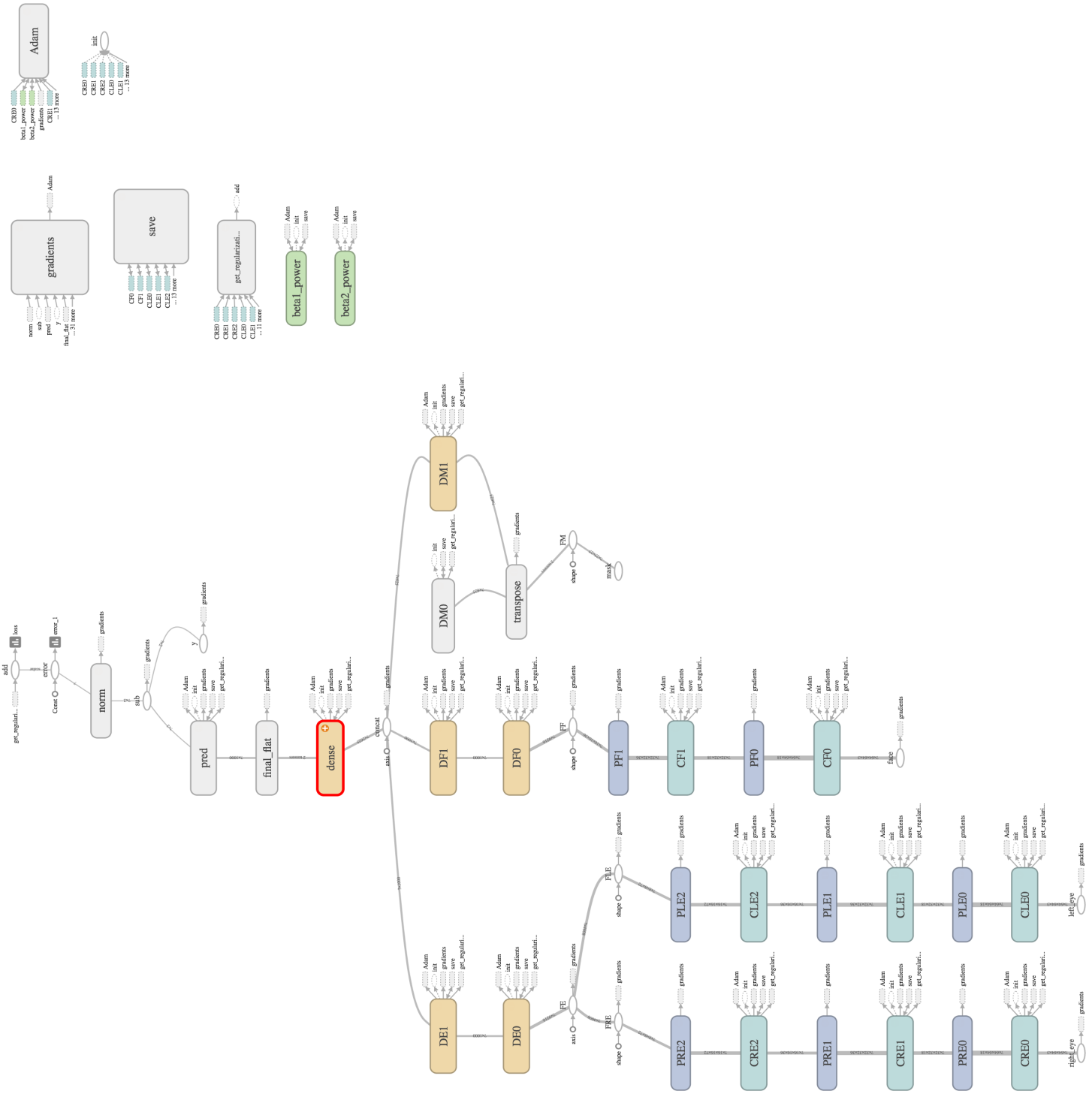

Figure 1: Tensorboard Visualization of Network Architecture.

4 Discussion and Conclusions

Before arriving on the architecture presented above, I attempted an architecture that learned basically learned nothing. I am now convinced that this earlier architecture failed because the filter sizes were much too large and there were not enough parameters: with a 32×32 filter with a stride of 4, the original images for the face and eyes are each reduced to 8×8 images after the first convolutional layer. In this reduction, a **ton** of very useful information is lost. Although it might be argued that the large size of the filter means that the model has more

parameters, and thus may not be as prone to underfitting, this increase in the number of parameters is useless if we are losing this much information in the first convolutional layer. Also, in the original architecture, the first convolutional layer only had 3 filters, and there were no densely connected layers between the convolutional outputs and the predictions. The mask only had one densely connected layer of size 625. Lastly, soon after observing the poor performance of the first architecture, I realized that my filter had even dimension; which is typically frowned upon because it this means that the filter cannot be centered on a single cell of the input.

I accounted for all these problems by using smaller filters, more convolutional layers, more filters in each convolutional later, and odd dimension filters. Also, I included more densely connected layers after the convolutions. By being more intelligent about the model architecture, my final architecture was able to achieve a validation error of 1.673cm, which is better than the state of the art 1.71cm achieved in [1].

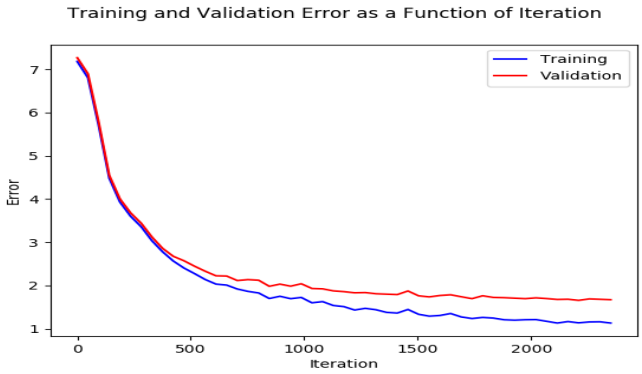

Figure 2: Training and validation error of final architecture over 25 epochs. The network achieved a final validation error of 1.673cm. Note, this plot is not directly from Tensorboard. I downloaded the error data from Tensorboard and remade the plots in MATLAB so I could have more control over the labels and other visualization subtleties.

References

[1] Kyle Krafka and Aditya Khosla and Petr Kellnhofer and Harini Kannan and Suchendra Bhandarkar and Wojciech Matusik and Antonio Torralba *Eye Tracking for Everyone*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.