ECSE 4965/6965 Final Project

Spring, 2017

Mobile Eye Gaze Estimation with Deep Learning

Due: May 10th, 2017, 11:59 PM EDT

1 Overview

In this final project, you will leverage on the learned Deep Learning techniques through the semester to tackle a challenging problem — mobile eye gaze estimation. Eye gaze estimation is to predict the direction or position you are looking at. Specifically for this project, we focus on mobile eye gaze estimation, that is, to predict the gaze position on the phone/tablet screen. Overall, our input is the face image captured by the frontal camera of the phone, and the output is a 2D position (x, y) on the phone screen. The following instructions will guide you through building an accurate mobile eye gaze estimator.

2 Original Dataset and Models

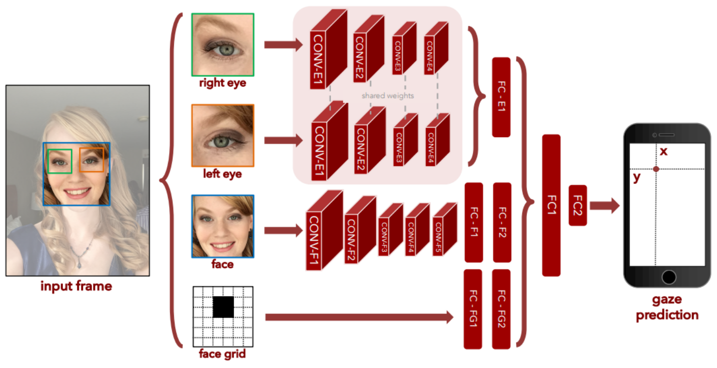

Figure 1: Overall architecture.

The original dataset comes from the GazeCapture project. The dataset involves over 1400 subjects and results in more than 2 million face images. Their model architecture is illustrated in Fig. 1.

The network involves 4 pathways. Besides face images, they also add eye crops for both left and right eyes as eye gaze is mostly affected by eye appearances. Besides, as the gaze prediction (x, y) on the screen also depends on the head/eye position, a face mask indicating face position in the whole image is also used. More details can be referred to the project page or their papers.

3 Our dataset

Due to the limitation of computation power, we create a much smaller dataset with 48000 training samples, 5000 validation samples and 7000 reserved testing samples. Each sample contains 5 items: face, left eye, right eye, face mask and labels. The dataset can be downloaded here. And you can use following code to read the data.

```
import numpy as np
npzfile = np.load("train_and_val.npz")

train_eye_left = npzfile["train_eye_left"]
train_eye_right = npzfile["train_eye_right"]
train_face = npzfile["train_face"]
train_face_mask = npzfile["train_face_mask"]
train_y = npzfile["train_y"]

val_eye_left = npzfile["val_eye_left"]
val_eye_right = npzfile["val_eye_right"]
val_face = npzfile["val_face"]
val_face_mask = npzfile["val_face_mask"]
val_y = npzfile["val_y"]
```

4 Potential Models

The goal of this project is to estimate eye gaze as accurate as possible. You can use whatever you have learned in the class or online to build a proper deep eye gaze estimator. The model architecture and hyperparameters are all up to you. You are free to use a similar architecture with four pathways as in the original paper, or you can design your own model architectures. Besides, you also need to choose proper hyperparameters like filter size, number of filters, etc.

4.1 Architecture Visualization

For this project, as everyone might use different architectures, you are required to visualize your model architecture through TensorBoard. After launching TensorBoard, navigate to GRAPHS on the top panel, and download the png image. It is also suggested to explore other utilities of TensorBoard. Fig. 2 shows an example of tensorboard visualization with four pathway input.

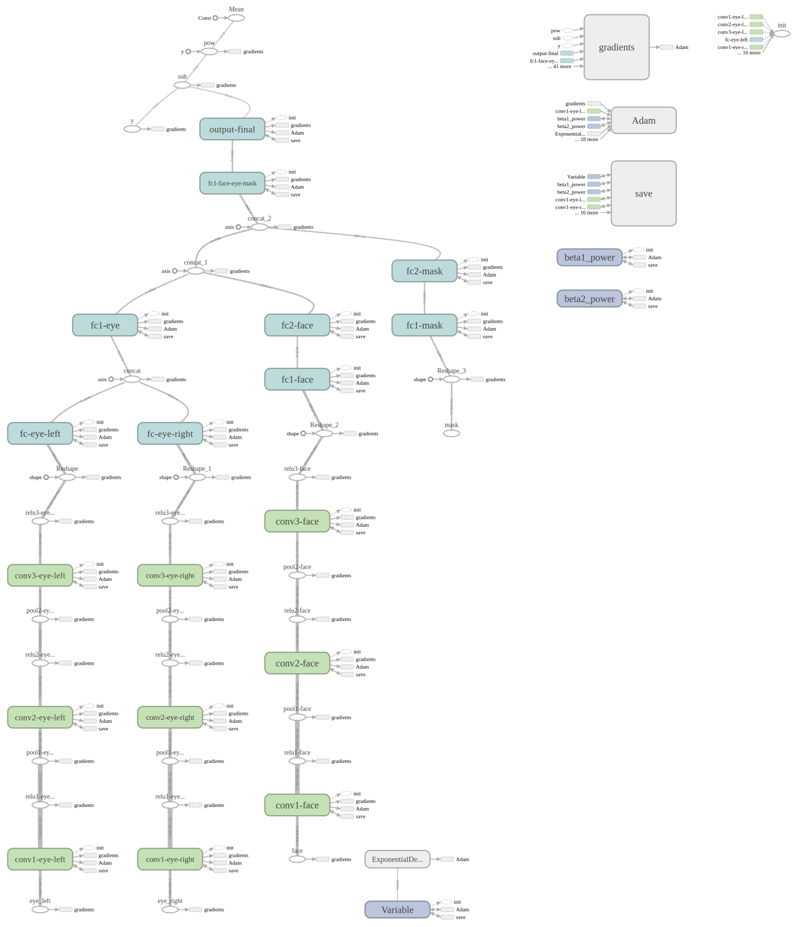

Figure 2: Four pathway graph tensorboard visualization.

4.2 Model evaluation

Different from previous assignments for classification, we need regression for this project. Assuming we have N testing samples, and your prediction $\mathbf{y}_p$ should be of size $N \times 2$. To evaluate the model, we use the mean Euclidean distance between $\mathbf{y}_p$ and ground truth position $\mathbf{y}$:

```
err = np.mean(np.sqrt(np.sum((yp - y)**2, axis=1)))
```

However, you are not required to use the above error as loss functions.

4.3 Fair comparison

We do have considered the case for students with less computation powers, which restrict them from using a complex model and getting better accuracy. For fair comparison, we first limit the training data size to 48000, to penalize complex models which leads to over-fitting. Second the grading rubric makes sure simple models (certain prediction error) can get at least 70% model evaluation credits.

4.4 Computational time and Expected Error

We have done benchmark testing with only one pathway (left eye) input, 3 convolutional layers and 2 pooling layers. Fig. 3 shows an example convergence curve. Note it is not converged yet and more training epochs should give better results.

When training on a Lenovo t440s machine with 8GB of RAM, it takes around 6 hours after 25 epochs, and reaches an error of 3 cm.

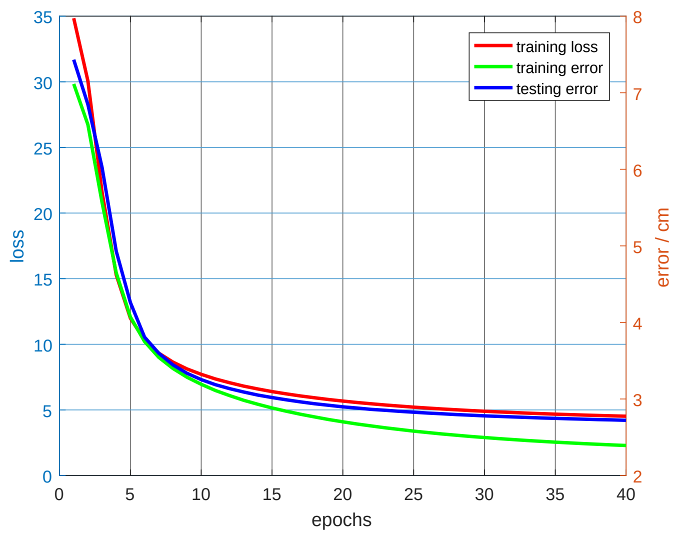

Figure 3: Loss and error for training and testing.

4.5 Model Saving

In order to evaluate your models, we need access to the placeholder for all 4 pathway inputs as well as the prediction operator. Notice this does not mean you need to use all 4 pathway inputs. If you choose to only build models with one pathway, then you need to create placeholder for other 3 pathways and add them to collection.

```
tf.get_collection("validation_nodes")
tf.add_to_collection("validation_nodes", eye_left)
tf.add_to_collection("validation_nodes", eye_right)
tf.add_to_collection("validation_nodes", face)
tf.add_to_collection("validation_nodes", face_mask)
tf.add_to_collection("validation_nodes", predict_op)
saver = tf.train.Saver()
save_path = saver.save(sess, "my_model")
```

5 Submission

You must submit the following items for this project:

- Your source code
- Your saved model ”my-model”
  - With graph collection with all four pathway input and the prediction nodes
  - Make sure to validate your model by running

    ```
        python validation_script.py my-model -v
           train_and_val.npz
    ```

- Write up
  - Loss and accuracy plots (Fig. 3)
  - Model architecture visualization (Sec. 4.1 and Fig. 2)
  - Explanation and justification

6 Grading Rubric

- 5 pts for code quality
  - Appropriate TensorFlow functions are used.
  - Readable and commented where necessary.
  - Not plagiarized.
- 30 pts for final error
  - error ≥ 4.0: 5 pts with a model that loads.
  - $4.0 >$ error ≥ 3.0: 10 pts.
  - $3.0 >$ error ≥ 2.6: 20 pts.
  - error < 2.6: rank with all students and assign proper credits. The first place get full credits.

- 25 pts for write up
  - 5 pts for clear explanation and justification.
  - 5 pts for loss and error plot.
  - 10 pts for clear and well-organized model architecture visualization.
  - 5 pts for architecture comparison or different pathway comparison.

7 Questions

Any questions on the dataset and the model, contact Kang Wang at wangk10@rpi.edu.