Introduction

Eye-tracking has important applications in computer vision, medical diagnoses, and other areas. However, most eye-tracking solutions today suffer from high cost, custom hardware, or lack of testing in real world conditions. Gaze estimation on phones with cameras offer a solution through the benefits of widespread usage, fixed position of the camera relative to the screen, and rapid development of mobile camera technology. For this task, we attempt to replicate “Eye Tracking for Everyone” (Krafka, et al.) while introducing new input features and architectural design elements.

Model/Architecture

Features: images of size [144, 144, 3] for 3 input features, 1 face grid mask, histogram of gradient on face images

Pipeline: 3 CNNs for the 3 images, connected with face grid mask and histogram of gradient on face via FC layers

Training: from scratch on Tensorflow, learning rate = 0.00001, optimizer = SGD

Implementation Details

CNN:
16 pixels in cell
11 x 11 / 96 relu conv, 2 x 2 pool
2 cells in block
5 x 5 / 256 relu conv
Stride 8
3 x 3 / 384 relu conv, 2 x 2 / 2 pool
1 x 1 / 64 relu conv

Evaluation

Metric for training: RMS distance (in cm) from location of the true gaze fixation

Scenarios
- (Input) Large variability in pose, appearance, and illumination
- (Features) HoG, OpenCV vs. Mechanical Turk bounding boxes
- (Architecture) Adding pooling layers and batch normalization
- (Hyperparameters) Changing optimizers, learning rates, LR decay

Preliminary Findings

Results are heavily dependent on number of samples and subjects. More training required

Future Direction
- Need to efficiently create HoG and OpenCV face/eye detection features
- Data processing of samples is currently a bottleneck
- With the previous iTracker architecture as a template, continue to tweak architecture to balance convergence time with accuracy