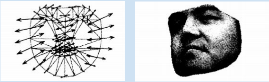
Gaze Estimation on Mobile Devices with Deep Convolutional Networks

Team Gazelle: Matthew Kim, Owen Wang (CS 231A), Natalie Ng (CS 231A)

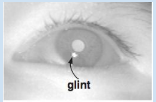
Introduction

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.

Previous Approaches



Shape-based models
(e.g. Active Appearance Model)



Corneal reflection-based models

Problem Statement

Input: images of faces looking at the screen of a mobile device with a camera
Output: (x, y) coordinates in cm where the model estimates the gaze to be where the origin is the location of the camera

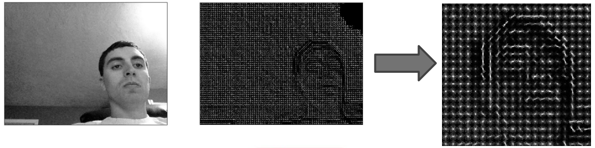
Approach: OpenCV for features including bounding boxes for eyes and face, CNN architecture on Tensorflow to train on features

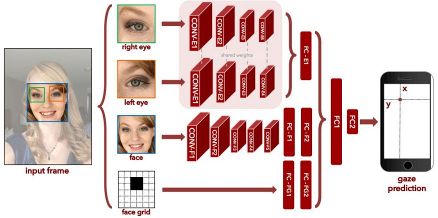
Dataset
MIT CSAIL "GazeCapture" dataset has 2.5 million images from 1474 crowd-sourced participants with corresponding dot locations to evaluate output coordinates on.

The entire dataset is hundreds of GB, with JSONs for four features per subject: left eye, right eye, face, and face grid mask. 13 fixed dot locations per device orientation for each subject

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





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

Implementation Details

HoG:

16 pixels in cell

2 cells in block

Stride 8

CNN:

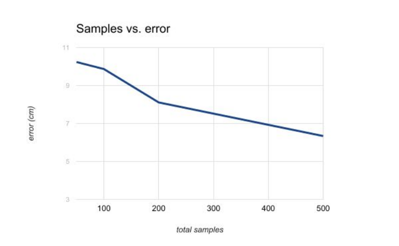
11 x 11 / 96 relu conv, 2 x 2 / 2 pool

5 x 5 / 256 relu conv

3 x 3 / 384 relu conv, 2 x 2 / 2 pool

1 x 1 / 64 relu conv

Preliminary Findings



Method	Error
Center	7.54
Gazelle (so far)	6.34
AlexNet	3.09
iTracker	2.58

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