Eye Spy

Are saccadic eye movements triggered by frame content?
Computer Vision

Katie Lamb – `katlamb`, Emma Spellman – `espell`, Nicholas Seay – `nseay`

June 12, 2019

---

1 Abstract

It is not well understood what thought process and procedures humans go through when they look at and analyze images. Our choice to gaze at certain locations in an image and then jump, or saccade, to other locations may be motivated by either the composition of the image itself or instead by brain signals independent of the image. Our model attempts to detect features in images that motivate jumps in the location of a subject's gaze. While the model is just a starting point in a complex research problem, the low accuracy of our predictions and model's lack of ability to pick up on image features for prediction indicates that saccadic eye movements are primarily triggered by the brain instead of features of the image.

2 Introduction

When humans look at an image they either fixate on a point or moving object, or their eyes will saccade to a new point, meaning they will make a rapid jump to a new location in the image. It is not well understood what motivates saccades, whether its features of the image or primarily signals from the brain independent from the image. With help from Dan Birman, a PhD candidate in Psychology at Stanford, we sought to build a machine learning model that could predict where one will saccade to, which would indicate there are features in an image that motivate saccades. This is a common question in vision science, and our meetings with Dan helped provide direction for the project. Modeling where eyes are drawn to is important for understanding graphics, product design, and user interfaces. It also provides further understanding of human vision and the brain as well as potentially providing insight into animal vision. It is important to understand if research should focus more on the viewpoint and image perception of the subject, or instead on the thoughts and brain signals that trigger saccades.

Existing models that address this question mostly focus on fixation points instead of saccades, predicting which group of points eyes will latch upon. Our goal for our model differs in that we try to predict, given a current point of fixation in a frame of a video, where the eyes will saccade to and next fixate upon in the next video frames. Additionally, many existing models have certain restrictions, like only working with black and white images and only considering simple features, like contrast and orientation of an image.

3 Methods

Our model works by passing video frames cropped around a central point of fixation into a convolutional neural network. The network outputs a distribution of probabilities assigned to a 4x4 grid corresponding to sections of the image, indicating the likelihood of the viewers gaze saccading to that position if they are currently fixating upon the given point.

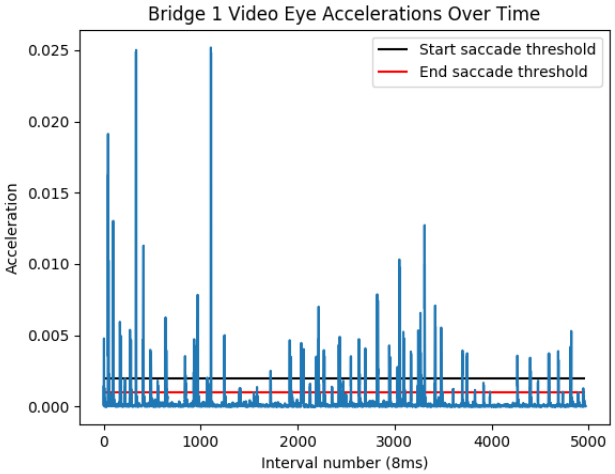

Figure 1. The acceleration in pixel/microsecond^2 between the start and end of 8ms intervals of one subject watching the Bridge 1 video.

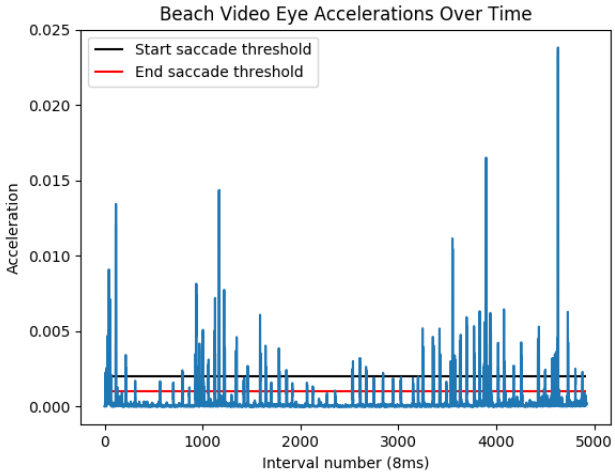

Figure 2. The acceleration in pixel/microsecond^2 between the start and end of 8ms intervals of one subject watching the Beach video.

3.1 Data

Our dataset is from the Universität zu Lübeck: Institut Für Neuro- Und Bioinformatik which has a set of eye movement data from people watching movies, trailers, and short clips. The data entries are organized as a time stamp given in microseconds from the start of the video, and the x and y pixel location of the eyes in the frame at that time. The data is grouped by media type, such as Hollywood movies, stop motion movies, and natural scenes. We chose to use the data from natural scenes, as often produced movie scenes have specific motions or features that are intended to draw the viewers eye. Using natural scenes also aligns with a primary motivation for the project – to gain a better understanding of how the human brain and vision interacts with its surroundings. Our data cleaning process involved two steps:

1. **Extracting saccades:** To determine what time frames in the data a saccade begins and ends, we used an algorithm based on acceleration that is implemented in previous studies from Araujo, Kowler, & Pavel, 2001, and Ehinger, et al., 2009. The algorithm works by breaking up the eye movement data into overlapping intervals. Our intervals were of size 8 milliseconds, so in consecutive intervals the second interval begins 4 milliseconds after the onset of the first interval. Since our data has discrete time frames and is not continuous, if it did not align precisely with the 8 millisecond interval, we took the eye location of the measurement from the next closest time frame. Next, we found the euclidean distance between pixels from the eye location at the beginning of the interval to location at the end of the interval, and divided this distance by the time of the interval to get velocity for each interval. Finally, acceleration between intervals can be found by computing the difference in velocities. A saccade starts when the acceleration between two intervals surpasses a threshold, and the saccade ends once the acceleration dips below another threshold.
Looking at the acceleration data shown in Figure 1. and Figure 2., we empirically determined the threshold to start a saccade as greater than .002 pixels/s^2 and the threshold to end a saccade as less than .001 pixels/s^2. The prior studies cited above that use this algorithm use a stricter ending threshold to account for the eyes overshooting a fixation point at the end of a saccade, and needing to be sure the location is the true fixation point at the end. Once the saccades were determined, the pixel location at the beginning of the saccade and the time frame of this location is one data entry, with its label being the location and time frame of the eyes at the end of that saccade.

2. **Producing images:** Once the pixel locations of the start and ends of saccades were found, these locations needed to be translated into images that can be fed to the CNN. With guidance from Dan, we created the images in a way that simulates how an eye looks at a point. The eye focuses on one point that is in full resolution, and then on the periphery the image looks blurry to the eye. To mimic this, we took three crops of the appropriate video frame, centered on the pixel location at the beginning of the saccade, from the time frame of this location.

The three crops are of size 128 x 128, 256 x 256, and 512 x 512 pixels, and are all compressed to size 64 x 64. We don't use a crop of original size 64 x 64 because the eye does not see on a perfect pixel by pixel resolution. By compressing each to 64 x 64, we blur each image by varying amounts, simulating how the periphery looks blurry to the eye. These three crops from each saccade are inputs to the CNN. To help with feeding images into the network, we combined the three crops into one image that made up one data point, which was later split up again when inputting the images into the encoder. The labels were the pixel that the eye saccades to for each data point. If the label pixel was outside the 512 x 512 crop area, which was a rare occurrence, we moved it in to the edge of this crop area. These label points were downsampled with the images as well, and all label points a row and column number between 0 and 63. An example input example and labeled pixel is shown in Figure 3.

Figure 3. Training entry of 3 downsampled crops pieced together. Red dot in the right frame indicates the area of the label pixel where the eyes will saccade to.

Figure 4. The same image entry with the green area indicating the model's prediction of the pixel location where the eyes will saccade to.

4 Model

4.1 Architecture:

Since the network receives three frames of varying resolution as input for a specific x_i, we use three SingleCropEncoders, which are each structured the same. Their outputs are then concatenated and fed into a decoder. Our first version of this decoder was a three layer decoder which outputted a matrix of the same shape as the initial input images. The entries in the matrix were probabilities giving predictions of how likely it is that the viewer saccaded to the corresponding pixel in the input image. However, we found that our model was not complex enough to predict among 64^2 possible pixels. Therefore, the second version of our model used a 2 layer CNN which outputted a 4x4 matrix representing sections of the image on which to make predictions.

SingleCropEncoder: The encoder consists of 3 ConvReluMaxpool layers. Each conv layer uses 16 filters and a padding of 1. The maxpool is 2x2 with a stride of 2. Therefore, the input starts as (N,C,64,64) and is convolved to shape (N,16,32,32), (N,16,16,16), and finally (N,16,8,8). Three of these were used in the full model, one for each crop around our fixation point.

ConvTransposeNet: This initial version of our decoder utilizes two transpose convolutions. The first uses a ReLU activation while the second uses sigmoid. Since the three encoders input is concatenated along the channel axis, the input for this half of the model is (N,48,8,8). The size then changes to (N,16,16,16), and ends

as (N,1,64,64). To achieve this, the first ConvTranspose uses 16 filters of size 4x4, stride of 2, and padding of 1. The second uses a single filter of size 6x6, stride of 4, and padding of 1.

ConvNet: This second version of our decoder utilizes two normal convolutions, as we no longer wanted to increase the size back up to the original image size. Both use a kernel size of 3, padding of 1, and stride of 1 to retrain the size of the image representation. The first conv in a ConvReluMaxpool layer uses 8 filters. The second, which is only followed by a softmax over the height and width dimensions, uses only 1 filter. Since the input for this half of the model is still (N,48,8,8), the size changes to (N,16,4,4) and ends as (N,1,4,4).

Loss: The output of the model is a probability distribution across the image frame, indicating the probability of saccading to each pixel. In the best case the highest probability is assigned to the label pixel location. An example output is shown in Figure 4, where the model predicts the highest probability of saccading to a location up and left of the true label. We use cross entropy loss with the probability outputted at the location of the true label.

4.2 Pre-training:

For the baseline model, we pretrained our SingleCropEncoders using data from CIFAR-10 which we resized from 32 x 32 to 64 x 64 pixels. At present, the pretraining can be much improved, both in accuracy and in quality. In an attempt to improve our encoded representation, we attempted using a pretrained vgg-16 model. However, this was also not a good option, because although the representation is much better, it is far too complex. After resizing the input to be 256x256, the vgg model outputted the same 8x8 height and width dimensions needed for our decoder. However, the channel width is 512, which increases the number of parameters to be more than our number of examples in the first layer of a modified post-vgg decoder alone. Training was very slow and unfruitful. We may decide to pretrain using CIFAR-100 in the future so that our network recognizes the lower-level features of more salient classes, without exploding the number of parameters the network must handle. CIFAR-10 notably does not have a class for humans, which are sure to be something that the people viewing the movies for the dataset instinctively notice and identify.

4.3 Results and Accuracy:

The final test set accuracy we obtained was 22.39% when predicting from the 16 sections of the image. This is certainly better than randomly guessing a location in the image, but not high enough to indicate from this model that saccades are triggered by features of the image. Perhaps the model must be more complex to learn more about the images, and more research on the topic may provide indication of specific steps or layers to add that simulate eye perception better. However these results are showing that it may be that saccades arent determined by the images being looked at, but external factors in the brain. The purpose of the model was to find out if there are features in images not seen or understood by humans that are influencing saccades. Since we did not obtain high accuracy, this might indicate that research on saccades by neuroscientists and psychologists should focus more on the brain's activity than vision and perception.

5 Future Work

One area that immediately stands out as a weakness in the model is our pre-training, which gets 60.19% validation accuracy. This is low, so we know pre-training is an area to address for improving the model. We could also experiment more with when to unfreeze these weights, or which decoders to unfreeze. It may be necessary only to unfreeze the SingleCropEncoders for the more downsampled crops (256 and 512) rather than unfreezing all/none of them. Better hyperparameter tuning or choices of optimizers might yield better results as well. We only experimented with Adam and SGD with momentum optimizers. Additionally, other research papers reference special layers to add on to models which specifically help simulate human vision specific problems, such as weighting pixels in the center of an image higher (T. Judd, 2009). While our initial experiments adding these layers did not yield better accuracy, more research on the topic may illuminate ideas for improving the model. After perfecting this version of the model, a very clear extension would be incorporating temporal elements. Our model received preselected saccade frames, but a more complex model may simply receive short clips, and predict both when and where the viewer may saccade.

6 Contributions

We divided the preliminary work into cleaning the training data and writing the model. Nick researched deep learning methods used in eye fixation detection experiments, and wrote the model and pre-training. Katie handled saccade detection and constructing the dataset, as well as research on methods for saccade detection. Emma helped with constructing, generating, and formatting data points. After combining the preliminary model and data, as a group we tested and tuned the model, as well as researched possible improvements.

7 References

C. Araujo, E. Kowler, M. Pavel. "Eye movements during visual search: The costs of choosing the optimal path." 2001.

K. Ehinger, et al. "Modeling Search for People in 900 Scenes: A combined source model of eye guidance." 2009.

M. Dorr, T. Martinetz, K. Gegenfurtner, E. Barth. "Variability of eye movements when viewing dynamic natural scenes." *Journal of Vision*, 10(10):1-17, 2010.

T. Judd, K. Ehinger, F. Durand, A. Torralba. "Learning to predict where humans look." *2009 IEEE 12th International Conference on Computer Vision,* 2009.

Dataset: Eye Movement Dataset from Universität zu Lübeck: Institut Für Neuro- Und Bioinformatik (https://www.inb.uni-luebeck.de/index.php?id=515)

Pre Training: CIFAR 10 dataset collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, 2009.

Repository: https://github.com/katherinelamb/eye-movement-prediction.

We would like to recognize Dan Birman for his guidance throughout the project.