Predicting familiarity from gaze data

Dan Graakjær and Jens Egholm

Copenhagen University
Dan G. Kristensen - nbs151@alumni.ku.dk
Jens E. Pedersen - xtp778@alumni.ku.dk

Abstract. Eye-tracking is a cheap tool for efficiently measuring and analysing responses from the visual system to visual or verbal stimulus. It has been shown that eye movement follows typical pattern based on input stimuli and familiarity of the stimuli. In this article we explore the predictability of eye movements of known and unknown visual stimuli, asking whether gaze data can be used to recognise the familiarity of faces. Using LSTM deep learning networks, we build four models to predict face familiarity based on fixations and fixation coordinates. We conclude that the performance is too poor to support the hypothesis and that further study with more participants, data and computational power is required to disprove the hypothesis.

Keywords: eye-tracking, face recognition, neural networks

1 Introduction

Noton and Stark showed that eye movements follow certain pre-learned patterns when recognizing stimuli [9]. Once a stimuli has been learned, the eye will follow the same patterns when met with the same stimuli in the future [2, 9]. These patterns have clear functions to aid us in learning tasks, and are employed differently for different types of stimulus [2]. Human faces have been extensively used in experiments because of the distinct features in focused areas of interests (AOI) that provides somewhat clear scanpath patterns [2, 3, 9]. However, Henderson, Williams and Falk questions whether the initially learned pattern is employed for every successive stimulus [2]. Instead, there is evidence that "eye movements become more restricted during recognition than during learning" [2, p. 104], indicating that eye movement patterns are different when a stimuli is being learned and when it is being recognised.

Judd et al has shown that machine learning models can be built to predict where humans will look, based on the AOIs of some given stimulus [5]. This proves the claim that there is some form of systematism to the human gaze [3,5,9,11], and that at least parts of it can be learned by an intelligent algorithm.

The hypothesis in this paper is that familiarity with a given stimulus can be determined solely based on gaze data. Specifically we will examine number of fixations and fixation coordinates to predict whether persons are familiar or unfamiliar.

1.1 Fixations

Fixations are time periods where the eye stands still for approximately 200-300 ms [3].[1] Fixations have shown to be associated with deeper processing in the brain, compared to brief visual verification such as pilots routinely checking instrument panels [3,11]. However, it is unclear whether fixation length is different in learning versus recognition [2]. In that respect fixation duration is unlikely to aid in familiarity prediction.

However Henderson et al. "found a greater concentration of fixation time on fewer critical features in recognition than in learning" [2, p. 104], indicating that the number of fixations and number of highlighted AOIs are bigger indicators. Recalling the scan path concept it is similarly plausible that the sequence of the fixations might be of importance, since scan paths visit AOIs in a certain order [3].

1.2 LSTM

Neural networks have shown a remarkable capability to recognise patterns in large amounts of data and are being applied in a vast array of domains [8, 13]. Early networks were brittle and fragile to large changes in input, causing the invention of back-propagating networks, that constantly re-evaluates the effect of stimuli changes [12,13]. Back-propagating networks suffers shortcomings when applied to larger networks where gradient descent algorithms are incapable of escaping local extrema [12]. To counter this, suggestions were made on how to retain and share network state, capturing the best properties and making them re-occur over the lifetime of the network [8, 12]. Networks of this type are dubbed recurrent neural networks. Hochreiter and Scmidhubner built on this idea with a model for Long Short-Term Memory (LSTM) [14]. The model 'remembers' previous configurations and retains the best configuration for future iterations [7, 14].

LSTM networks build on a more complicated model of a neuron named an LSTM unit. The unit is more complex than a perceptron because the input-output flow is accompanied by three gates, which can influence the activation of the neuron [13, 14]. The gates influences the input, output and 'forgetting rate' of the LSTM unit, using information threshold that persists beyond the units, giving it a more permanent and stable state (ibid.). This architecture has won several competitions in within pattern recognition and are unbested in many domains [13].

1.3 Neural network optimization

Historically gradient descent has been a popular choice as a method to train neural networks because it is quicker to adapt than simple linear adaptation [12]

[1] In fact the human eye constantly performs *tremors* and never stands completely still [3].

[8]. To avoid falling into local extrema, stochastic gradient descent (SGD) are heavily used as a noisy approximation [12]. Another method called resilient back-propagation (Rprop) attempts to correct weights in neural networks in back-propagating networks by taking previous weights into account [8]. Inspired by both the SGD and Rprop algorithms, root mean square propagation (RMSprop) applies gradient descent while adapting to the magnitude of recent gradients [15]. This approach has proved efficient for training recurrent neural networks due to its back-propagating and stochastic abilities (ibid.).

1.4 Neural network regularization

Excessive training on the same data can lead to generalisation problems [12]. Too many iterations of a training set can train the machine learning model to adapt to specific features of the training dataset instead of global features represented by a separate test dataset. This type of overfitting leads to drastic performance drops when cross-validating the model. A common strategy to avoid overfitting in neural networks, and LSTMs in particular, is to randomly drop features in the network [12]. This forces the network to redundantly reinforce input patterns and throw away less important structures, making it more robust and less prone to data-specific feature adaptation.

2 Method

2.1 Participants

Ten participants, aged 20-38 years took part in the experiment. Of the ten participants, five were female and five were male. While some of the participants used glasses, no significant eye conditions were reported by any of them.

2.2 Stimuli

The stimuli consisted of 103 images of different faces. 3 of the 103 images acted as preparation images and were not included in the analysis. The 100 images belonged to three different categories: Famous, non-famous and domain. The famous category consisted of 25 images of very famous and easily recognizable people, while the non-famous category contained 25 images of non-famous people and the domain category consisted of 50 images of people who were famous within different domains, i.e. music, politics or sports.[2] A domain category ensures that the images are not rated the same by each of the participants and thereby providing diversity in the data.

All images were normalized to control for spurious effects of colour, orientation and size. All images were grayscaled and aligned both with regards to

[2] The exact stimulus is shown in Appendix C: Stimulus on page 5

rotation and eyes, based on three AOIs: left eye, right eye and nose. The scaling was done using OpenCV.[3]

The experiment resulted in gaze data from 10 participants over 100 images each, providing 1000 data points.

2.3 Experimental design

All gaze data was recorded using a Tobii T120 Eye Tracker and the Tobii Studios Software and was afterwards processed using Python.[4]

The experiment itself was conducted using a manuscript (see Appendix B: Experimental design and protocol) where each participant was welcomed and given a short briefing on the experiment. He/she was placed in front of the eye tracker which was then calibrated. Here they were presented with a preparation segment (which was not included in the final analysis) and then a segment for each of the 100 images. One segment consists of a fixation point (1.5 seconds), an image (3 seconds) and a questionnaire where they would rate the familiarity of the person in the image (until an answer was given). The order of the segments were randomized, but were the same for each participant. The experiment was concluded when the participant had gone through all 100 segments. In the questionnaire the participants rated the familiarity of a person using a Likert scale shown in table 2.3.

1. Not at all familiar
2. Slightly familiar
3. Somewhat familiar
4. Moderately familiar
5. Extremely familiar

A pilot study was performed, where the fixation point was set to 2 seconds and the image for 5 seconds. It was found that the participant expressed a high degree of fatigue after the first 70 pictures (500 seconds + response time $\approx$ 20 minutes). This was visible by several data points missing in the gaze-data and the participant's verbal expressions. For that reason we decided to decrease the image and fixation point time to 1.5 and 3 seconds respectively.

2.4 Preprocessing

The output gaze data for each participant was cropped to the time where the participants were actively looking at images, and paired with the questionnaire response for that image.

[3] The script is available in Python, see Appendix A: Code on page 5.

[4] For a more detailed runthrough of the experimental design, see Appendix A: Code and Appendix B: Experimental design and protocol on pages 5-5.

Since the participants rated each image using a Likert scale, there is five degrees of familarity in our data set. The purpose of this study is to predict familarity and not degree of familarity, so the labels were transformed, reducing the number of classes from 5 to 2, taking a binary classification approach to the data instead. To ensure as little loss of information as possible, we employed two different approaches:

1. Only the label 'Not at all familiar' was recognized as truly unfamiliar and the rest was labeled as familiar
2. The label 'Not at all familiar' is labeled as unfamiliar (0), all vectors with label 'Slightly familiar' or 'Somewhat familiar' removed from the data (two smallest and most semantically vague categories). Vectors with label 'Moderately familiar' and 'Extremely familiar' labeled as familiar (1).

Table 1. Distribution of ratings

Not at all familiar	Slightly familiar	Somewhat familiar	Moderately familiar	Extremely familiar
394	120	75	123	288

By taking a binary approach, we ensure that the data is within the scope of our hypothesis and get a much better prediction accuracy as it will be extremely difficult predicting an exact rating between 1-5. Looking at table 1, the two extremes 'Not at all familiar' and 'Extremely familiar' are the most represented categories in the data, with 'Not at all familiar' being the most dominating. In the new distribution, after the binary transformation, the 'Familiar' category is the most dominant category with 606 occurrences in the first binary category and 486 in the second one.

Table 2. Distribution of ratings after transformation

	0 (Unfamiliar)	1 (Familiar)
Binary	394	606
Binary no 2 and 3	394	486

After the labels were transformed, the most important features of the data set were extracted: The fixation coordinates and the number of fixations. Each fixation coordinate accounts for 8-9 milliseconds of the recorded data, meaning that it is possible to have several fixation coordinates in sequence with same value (the more identical fixation coordinates in sequence means a longer fixation). The number of fixations in an image can then be estimated as the number of unique fixation coordinates in sequence. Another important thing to note is that the number of fixation coordinates varied a lot between the 1000 data points. In most cases, the number of coordinates was around 300, but a few of

the data points had below 200 coordinates. This raised several issues: Firstly, if a data point has too few coordinates, it indicates that the eye tracker had problems reading the participants eye movements correctly. If this is the case, then the sample might not be representative of the participants eye movements in the experiment and could contaminate the analysis. Secondly, since each data point represents the participants fixation point over time, we want the same number of coordinates for each participant. We cannot compare two data points if one of them is significantly longer than the other, as time may very well be a compromising factor. Therefore all data points with fewer coordinates than 213 were removed from the data set and the rest of the data points were sliced, resulting in a regular array of length 213, corresponding to an unknown number of fixation points over a timespan of around 1900 milliseconds for each of the images.

2.5 Data models

Four models could now be constructed combining either binary or binary without the second and third category, with the number of fixations or all fixation coordinates. All four models employ LSTM networks [13], using the deep learning framework Keras [6]. To avoid overfitting each layer drops 50% of the LSTM features.

All hyperparameter tuning was performed using Hyperas [4]. The parameters being trained was single, double or three-tiered layered LSTM networks, number of LSTM cells for each layer (from 2 to 128), the activation function for the final network layer and the optimizer (RMSprop, adam or SGD).[5]

2.6 Metrics

For assessing the models three values are used: precision, recall and F1 score. Precision denotes the amount of correct guesses out of the total amount of guesses made. Recall tells us how many of the positives are actually false negatives, or, how many relevant items have been selected. F1-score is a mean of how well the amount of correct positives compare to the amount of all positive results from the classifier. The distinction between precision and recall is necessary to know how well the model not only classifies the correct results, but also capture the wrong results. This property is especially important to avoid false positives [1].

Fig. 1. Formula for calculating F_1 score [16].

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

All scores range from 0 to 1, where higher is better. If the models attain a precision of 1, it correctly predicts all samples correctly.

[5] See Appendix A: Code on page 5 for more information.

Cohen's Kappa When dealing with unbalanced classes in classification, it is important to take into consideration the expected vs. observed accuracy of ones model. This means that the class frequency has to be considered a factor when evaluating ones results. In the case of this study, we have two different data splits:

1. Binary, where 60% of the instances belongs to the familiar class (606/1000)
2. Binary no 2 and 3, where 55% of the instances belongs to the familiar class (486/880)

This means that while the models might report accuracies above chance, which is 50% when dealing with binary classes, the unevenly distributed instances will have influenced these numbers and above chance will now have a higher threshold. The point of the Kappas coefficient is to take these points into consideration when evaluating the model. Here the distance between the observed accuracy (precision) and the expected accuracy (the performance of a totally random classifier). If a large distance is observed, one can conclude that the model is performing above chance and the model can be considered successful. While the interpretation of the kappa values are relative to the specific case, some generalisations can be made. According to Fleiss, J.L. [10] a kappa value of 0.75 can be considered as excellent, 0.40 to 0.75 is fair to good, and below 0.40 is poor.

3 Results

Table 3 shows the precision, recall and F1 scores for the models. While the F1 values for both models look acceptable at first glance (>0.75), the precision and recall values indicates an underlying issue. The high recall in all cases seems to be a symptom of a too inclusive model, which in this case means that too often the model will classify instances as belonging to the 'familiar' class. This also explains why precision is relatively high while the model performs poorly, as the dominant class frequency will influence the precision.

Table 3. Precision, recall and F1 score respectively for all four models.

	Number of fixations	Fixation coordinates
Binary	0.647 / 0.862 / 0.751	0.658 / 0.961 / 0.780
Binary no 2 and 3	.663 / 1.0 / 0.792	0.643 / 1.0 / 0.782

To better understand the performance of a model when dealing with unbalanced classes, it's necessary to take the class frequency into consideration and get an idea of the distance between the expected vs. actual accuracy. Calculating the Cohen's kappa coefficient will do exactly this and give a metric based on this distance:

Table 4. Cohen's kappa coefficient

	Cohen's kappa
Fixations and binary	0.027
Fixations and binary no 2 and 3	0.0
Coordinates and binary	0.116
Coordinates and binary no 2 and 3	0.0

Table 4 reports a very low kappa value for all of the models. As per the metrics section, a kappa value below 0.40 can be considered as poor. In this case all kappa values are below 0.4, indicating low distance between the expected vs. observed accuracy. In sum none of the models have a significantly higher accuracy than expected, when taking the class frequencies into consideration.

4 Discussion

While the f1 scores are relatively high, the precision and Cohen's kappa shows us that the model is imperfect. The models work solely with a binary dependent variable, where 60% of the data is within a single category. In absence of clear patterns in the data, the models have reverted to the simplest baseline. This does not entirely disprove the theory however. Cohen's kappa informs us how much is attributed to chance, which, by definition, cannot be much different than the precision scores extracted from the models (around 60/40).

A number of choices have been made in setting up the experiment and constructing the models that should be investigated. Two choices is particularly interesting: the discrete dependendent variables and the number of participants.

All models were predicting a binary dependent variable, which leaves out a large part of the complexity captured by the concept of *familiarity*. Having, say, a numeric dependent variable would force the model to take this into consideration and reveal any subtleties between the two extremes. This also questions the method with with *familiarity* is measured. Choosing a Likert scale might not be the most optimal way of rating it. Instead it would be interesting to examine a scale with 10 or even 100 discrete steps.

Second choice regards the number of participants. 10 participants is considered a small experiment, and the size and homogeneity of the group results in low external validity. For a truly comparable experiment the participants should be increased and diversified.

A final point for discussion is the use of deep learning frameworks. LSTM networks have proven to discover and retain patterns in large datasets. If the patterns are hard to distinguish and possibly even individual, this simple network (compared to the human brain) will have a hard time to discover the correlations. A future study should execute the simulations on a much more powerful computer, allowing for a wider range of hyperparameters for both the LSTM cells and number of layers.

5 Conclusion

This article set out to conduct an eye tracking experiment with the hope of finding a correlation between eye movement patterns and familiarity of visual stimuli. A study was conducted using 10 participants that rated 100 images to be more or less familiar. Four models was constructed to extract relevant patterns from the data, but all showed a mismatch between their precision and recall, leading to the conclusion that none of the models were able to improve beyond baseline. While this study argues for the rejection of the hypothesis, no strong generalization should be built upon these findings due to low external validity and the improvements that could be made to the data processing, experiment scale and machine learning setup.

References

1. Agresti, Alan and Finaly, Barbara: Statistical methods for the social sciences. Pearson Prentice Hall, 2009, ISBN 978-0-13-027295-9
2. Henderson, John M. and Williams, Carrick C. and Falk, Richard J.: "Eye movements are functional during face learning", in Memory & Cognition (33), 2005 pp. 98–106
3. Holmqvist, Kenneth and Nyström, Marcus: Eye tracking, A comprehensive guide to methods and measures, Oxford University Press, 2011, ISBN: 978-0-19-873859-6
4. Hyperas - Hyperparameter optimization framework, `http://maxpumperla.github.io/hyperas/`, retrieved 1/1/2018
5. T. Judd, K. Ehinger, F. Durand and A. Torralba, "Learning to predict where humans look," 2009 IEEE 12th International Conference on Computer Vision, Kyoto, 2009, pp. 2106-2113.
6. Keras - Deep learning framework for Python, `https://keras.io`, retrieved 1/1/2018
7. Kristensen, Dan G., Pedersen, Jens E. and Kenhof, Tobias S. W. Y.: Dominant modalities in LSTM networks, Cognitive Science 2, IT & Cognition, Copenhagen University, 2017
8. Nilsson, Nils J.: "The quest for artificial intelligence - A history of ideas and achievements", Cambridge University Press 2009.
9. Noton, D., & Stark, L. (1971). Scan paths in eye movements during pattern perception. Science, 171, 308-311
10. Fleiss, J.L. (1981). Statistical methods for rates and proportions (2nd ed.). New York: John Wiley.
11. Rayner, K. (1998). Eye movements in reading and information processing:20 years of research. Psychological bulletin, 124(3), 372.
12. Russel, S. and Norvig P., "Artificial Intelligence, A Modern Approach", Third edition, Pearson New International Edition, Pearson, 2014
13. Schmidhuber, J.: Deep Learning in Neural Networks: An Overview, Neural Networks (61), pp 85-117, 2015
14. Schmidhuber, Jürgen and Hochreiter, Sepp: "Long Short-Term Memory", Neural Computation, volume 9, issue 8, November 15, pp. 1735-1780, 1997
15. Tieleman, Tijmen and Hinton, Geoffrey: "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", Coursera: Neural Networks for Machine Learning, 2012.
16. Wikipedia article: "Precision and recall", `https://en.wikipedia.org/wiki/Precision_and_recall`, retrieved 1/1/2018

Appendix A: Code

All code is available on GitHub at `https://github.com/Jegp/facerecognition`.

Face alignment

The scripts for the face alignments are available under the `face-alignment` repository. The `align_faces.py` script locates AOIs of a face, grayscales the colours while scaling and rotating the picture so all faces are centered around the same location with the same distance to the AOIs.

Model construction

The models were all built with Keras [6] and the hyperparameters were found using Hyperas [4]. The code for the model building can be found at the above GitHub in the file `model.py`.

All models were persisted for the sake of replication and can be found on the above GitHub under the names `model_[input]_[output].model` where `[input]` is either *fixations* or *xy* and `output` is either *binary* or *binary no 2 and 3*.

Model evaluation

The most optimal models are stoder in `*.model` files in the GitHub directory. The are evaluated in the Jupyter Notebook file `Evaluation.ipynb`. The file can be viewed directly from GitHub at `https://github.com/Jegp/facerecognition/blob/master/Evaluation.ipynb`.

Appendix B: Experimental design and protocol

1) Instructions read out to participants
   a) "In this experiment we will ask you to look at faces of more or less familiar people. For each face you will be asked to rate how familiar the face is, ranging from "Not at all familiar" to "Extremely familiar". The experiment is expected to last 20 minutes. Before the experiment we will calibrate the eye-tracker. Please find a position where you can sit comfortably for 20 minutes to avoid re-calibration. Do you have any questions before we begin?"
2) Calibrate eye-tracker
3) Instructions on screen
   a) "In this experiment we will ask you to look at faces of more or less familiar people. For each face you will be asked to rate how familiar the face is, ranging from "Not at all familiar" to "Extremely familiar"."
   b) "Press key when ready."
4) Pilot phase
   a) 1x:
      i) 1,5s - Cross
      ii) 3s - pilot image from domain category, non-randomized selection
      iii) Until selected - answer
   b) "Everything clear? Press any key when ready to start."
   c) 1x (without informing the participant):
      i) 1,5s - Cross
      ii) 3s - pilot image from domain category, non-randomized selection
      iii) Until selected - answer
5) Testing phase
   a) 100x:
      i) 1,5s - Cross
      ii) 3s - testing image, randomized selection
      iii) Until selected - answer
6) End
   a) "Thank you for participating."

Answer:

When answering, participants will select the most appropriate score that, in their opinion, describes the familiarity of the person in the photo. We use this *Linkert scale* for familiarity.

Data:

Custom data set of 100 (+2) photos was created using free online resources. Each image belongs to one of three categories - famous, domain, unknown. Famous and unknown categories contain photos of people that should be universally recognizable or unrecognizable, respectively. Domain category includes photos

that depict people who are famous and recognizable, however, only if the participant is familiar with their domain (i.e. sports, politics,...). Domain category ensures that an overlap in familiarity/unfamiliarity will exists in the dataset, however, it will not be at a cost of increased confusion (i.e. domain people are either very familiar or almost unknown).

Checklist

- Phones on silent
- Ask questions before start
- Leave the room to prepare for next participant

Literature

https://www.researchgate.net/publication/262886751_Vision_Research_a_Practical_Guide_to_Laborato
https://www.researchgate.net/publication/254913339_Eye_Tracking_A_Comprehensive_Guide_To_Met

Appendix C: Stimulus

The stimulus was found using the Google search engine. Attribution, links and the actual stimuli is available through the GitHub website listed in Appendix A: Code.

Attributions and links can be found in the `stimulus.md` file.

Actual stimuli can be found in the `domain`, `famous` and `unknown` folders.