Behavior Research Methods (2019) 51:840–864
https://doi.org/10.3758/s13428-018-1133-5

gazeNet: End-to-end eye-movement event detection with deep neural networks

Raimondas Zemblys[1] · Diederick C. Niehorster[2] · Kenneth Holmqvist[3,4,5]

Published online: 17 October 2018

Abstract

Existing event detection algorithms for eye-movement data almost exclusively rely on thresholding one or more hand-crafted signal features, each computed from the stream of raw gaze data. Moreover, this thresholding is largely left for the end user. Here we present and develop gazeNet, a new framework for creating event detectors that do not require hand-crafted signal features or signal thresholding. It employs an end-to-end deep learning approach, which takes raw eye-tracking data as input and classifies it into fixations, saccades and post-saccadic oscillations. Our method thereby challenges an established tacit assumption that hand-crafted features are necessary in the design of event detection algorithms. The downside of the deep learning approach is that a large amount of training data is required. We therefore first develop a method to augment hand-coded data, so that we can strongly enlarge the data set used for training, minimizing the time spent on manual coding. Using this extended hand-coded data, we train a neural network that produces eye-movement event classification from raw eye-movement data without requiring any predefined feature extraction or post-processing steps. The resulting classification performance is at the level of expert human coders. Moreover, an evaluation of gazeNet on two other datasets showed that gazeNet generalized to data from different eye trackers and consistently outperformed several other event detection algorithms that we tested.

Keywords Eye movements · Event detection · Deep learning · Fixation · Saccade · PSO

Introduction

In eye-movement research, the goal of *event detection* is to robustly extract events, such as fixations and saccades, from the stream of raw data samples provided by an eye-tracker. For a long time, two broad classes of algorithms were used: First, the *velocity-based algorithms* that detect the high-velocity saccades and often assume the rest of the data to be fixations. The most well-known is the I-VT algorithm of Bahill, Brockenbrough, and Troost (1981), and Salvucci and Goldberg (2000), but the principle can be traced back to algorithms by Boyce from the 1960s (Boyce, 1967). The *dispersion-based algorithms* instead detect fixations by defining a spatial box that the raw data must not exit for a certain minimum time, and usually assume the rest to be saccades. The best known is the I-DT algorithm of Salvucci and Goldberg (2000).

Velocity and spatial boxes are examples of *hand-crafted features*, i.e., transformations of the eye-tracking data that are designed beforehand by algorithm developers in order to provide different descriptions of the data that the specific algorithm uses. The last decade has seen several improved event-detection algorithms, as researchers have gained access to new descriptions of the eye-movement signal by hand-crafting new features and developed bespoke algorithms to exploit these features for event detection.

For instance, Engbert and Kliegl (2003), Nyström and Holmqvist (2010), and Mould, Foster, Amano, and Oakley (2012) use adaptive thresholds to free the researcher from having to set different thresholds per trial when the noise level varies between trials. Nonetheless, these algorithms still only work over a limited range of noise levels (Hessels, Niehorster, Kemner, & Hooge, 2017). Another

✉ Raimondas Zemblys
r.zemblys@tf.su.lt

[1] Research Institute, Siauliai University, Šiauliai, Lithuania

[2] Humanities Laboratory and Department of Psychology, Lund University, Lund, Sweden

[3] Department of Psychology, Regensburg University, Regensburg, Germany

[4] Department of Computer Science, University of the Free State, Bloemfontein, South Africa

[5] Faculty of Arts, Masaryk University, Brno, Czech Republic

recent development is the work by Larsson, Nyström, and Stridh (2013) and Larsson, Nyström, Andersson, and Stridh (2015). These authors developed new features that have enabled the automatic detection of smooth pursuit and post-saccadic oscillations in clean data recorded at a high sampling frequency that contain these events intermixed with fixations and saccades. Furthermore, Hessels, Niehorster, Kemner, and Hooge (2017) have presented a novel largely noise-resilient algorithm that can successfully detect fixations in data with varying noise levels, ranging from clean data to the very noisy data typical of infant research. These algorithms are designed to solve a specific problem—smooth pursuit detection, or noise resilience—using algorithmic rules and user-settable thresholds specifically designed for that problem. However, these novel algorithms still rely on increasingly complex features that may be computationally expensive to calculate and often come with an increasing number of either tunable or hard-coded parameters, which in turn may require experience and insight from the end user. On the positive side, the thresholds and exploited features are often possible to interpret and implement. For extensive overviews of event detection algorithms and detection methods they use, see Andersson, Larsson, Holmqvist, Stridh, and Nyström (2017), Hein and Zangemeister (2017), and Holmqvist and Andersson (2017).

A new development is the appearance of event detection methods based on machine learning techniques. Up till now, these detectors have still made use of the same hand-crafted features as the above classes of algorithms, but the thresholding and classification are learned from the data and performed automatically by the event detector. Zemblys (2016) compared ten machine learning algorithms for event detection while other examples include work using support vector machines Anantrasirichai, Gilchrist, and Bull, (2016), convolutional neural networks (Hoppe & Bulling, 2016) and random forests (Zemblys, Niehorster, Komogortsev, & Holmqvist, 2018). Hoppe and Bulling (2016) show that a simple one-layer convolutional neural network (CNN), followed by max pooling and a fully connected layer outperforms (agrees more closely to human experts) algorithms based on simple dispersion and velocity and PCA-based dispersion thresholding. Zemblys et al. (2018) has shown that IRF, a random forest-based event detector, which uses features adopted from existing event detection algorithms, outperforms previous state-of-the-art algorithms. In fact, in clean SMI HiSpeed eye-tracking data (sampled at 500 Hz with average noise level 0.042 degrees RMS), performance almost reached that of a human expert. Detection performance of Zemblys et al. (2018) was furthermore stable down to 200Hz, with the algorithm still performing quite well at even lower sampling frequencies. Moreover, classification performance as evaluated by human experts was largely robust to the increasing noise—only when the average noise level exceeded 0.5 degrees RMS, the performance started to degrade noticeably.

IRF by Zemblys et al. (2018) and these other machine learning methods still use hand-crafted features, but they do not require any thresholds to be set by the user and once trained, they are computationally very fast. Although their paper's title suggests differently, the algorithm by Hoppe and Bulling (2016) is still not entirely end-to-end as it uses a hand-crafted feature. Specifically, they use a Fast Fourier transform to extract a number of frequency components (features) first and then use these as input to their algorithm. However, as the authors write: "it would be conceptually appealing to eliminate this step as well" (Hoppe & Bulling, 2016, p. 12). In addition, neither Hoppe and Bulling (2016) nor Zemblys et al. (2018) directly take into account long-term temporal dependencies of samples—their event detector models are not aware of what event label they assigned to previous samples and cannot learn temporal dependencies such as, e.g., that a post-saccadic oscillation (PSO) can only occur after a saccade. Here we propose to use an entirely end-to-end deep learning-based approach, which takes raw eye-tracking data as input and classifies it into fixations, saccades and PSOs. In contrast to the above-mentioned machine learning-based algorithms, the deep learning approach we present in this paper automatically learns all features and appropriate thresholds from the data. It furthermore learns how the short- and long-term context of each raw data sample affects what the current sample can be classified as, thereby ensuring that event codes are produced in sensible sequences without the need for post-processing, as a human expert would produce them. It should be noted that this paper aims to present a proof of concept in the form of a framework for developing end-to-end event detectors by means of deep learning techniques. The reader is encouraged to use this framework to build their own event detector which is ideally suited to the data they have, and tuned to detect the events they are interested in.

Using supervised deep learning to develop an event detector for eye-movement data comes with an important bottleneck: many hours of eye-movement data must be hand-coded to serve as training data for the algorithm. Having too little hand coded training data would not lead to a good event detector when training a deep network, because it would simply overfit the data and would not be useful on unseen trials. Preparing training data therefore requires experts in eye-movement research to invest the time to look through substantial stretches of data carefully. This is very expensive and does not guarantee the same labeling quality in the beginning and the end of the coding. It is very possible that the coder becomes tired after long stretches of coding and makes mistakes (see Fig. 11 in Appendix B). Moreover, Hooge, Niehorster, Nyström, Andersson, &

Hessels (2017) report that not all coders are consistent during the entire coding session and instead change their criteria when labeling onsets and offsets of the fixations.

In this paper, we present a completely end-to-end eye-movement event detection algorithm, *gazeNet*. We use a supervised learning approach, i.e., we take manually annotated eye-tracking data as input and train an end-to-end deep learning-based event detector. Furthermore, we solve the problem of not having enough training data by extending a relatively small dataset consisting of the annotated eye-tracking recordings by first training a generative neural network, *gazeGenNet*, to produce synthetic labeled eye-tracking data. We then use this synthetic data as input to a deep learning network that trains an event detector to take raw eye-movement data as input and produce a eye-movement event classification similar to that of an expert, without any need for calculating hand-crafted features and post-processing steps, and without the need for the user to set any parameters. The resulting event detector furthermore operates in a fundamentally different way than the vast majority of the previous event detectors, which use an hierarchical approach, where they first detect events of one target type such as fixations and then in a later step process the remaining data. In contrast, gazeNet detects all events simultaneously, considering all event types at once and deciding which is the most likely for each sample while also taking into account its context. Finally we test the resulting event detector on another set of gaze data that was manually annotated by human experts and compare its performance to two threshold-based and two machine learning-based eye-movement event detection algorithms.

Method

Data

The data we are using in this study are manually annotated eye-tracker data from the Humanities Lab, Lund University (hereafter called the *Lund2013* dataset[1]). It consists of monocular eye-movement data of participants viewing images, videos, and moving dots. The eye-movements of all participants were recorded with the SMI Hi-Speed 1250 eye-tracker, running at a sampling frequency of 500 Hz. Two domain experts then manually segmented data into fixations, saccades, post-saccadic oscillations (PSO), smooth pursuit, blinks and undefined events. A comprehensive description of the *Lund2013* dataset and the coding process can be found in Larsson et al. (2013).

[1] Available for download at http://www.humlab.lu.se/en/person/MarcusNystrom/ or https://github.com/richardandersson/EyeMovementDetectorEvaluation

In this study, we did not use the video and moving dot trials from the *Lund2013* dataset to avoid trials that contain pursuit, which is out of the scope of this paper. We only used 20 unique image-viewing trials, which presumably contain only fixations, saccades, PSOs and unlabeled events. We will hereafter refer to this subset as the *Lund2013-image dataset* or just *dataset*. The Lund2013-image dataset consist of recordings of four subjects looking at four different images; some of the subjects looked at the same images multiple times, yielding a total of 20 trials. Trials were 4 or 10 s long. Most of the Lund2013-image dataset—14 of the 20 trials were independently coded by two eye-tracking experts - coders *MN* and *RA*. The remaining six trials were only coded by coder *RA*. We will use these six trials as training data (*trainSet*) to train the generative neural network used for manufacturing more labeled eye-tracking data. Further, we randomly selected three trials which were coded by both of the coders and used them for validation (*valSet*) when training *gazeNet*. The remaining 11 trials, coded by both of the coders were used for testing the performance of the resulting event detector (*testSet*). For a full list of files and their assignments to the different sets see Table 9 in Appendix A. We also found an obvious labeling mistake in one of the validation trials, that we have fixed by reassigning 75 samples from the saccade to the fixation class (see Fig. 11 in Appendix B).

Any eye-movement recording will yield highly unbalanced data in the sense that the vast majority of samples belongs to fixations, because of their longer duration compared to saccades and PSOs. For example, Zemblys et al. (2018) report that nearly 89% of samples belong to fixations in their dataset from a fixate-saccade experiment, compared to on the one hand (Tinker, 1928) reporting an average of 94% and on the other Hooge et al. (2017) reporting only 71.1% ($\sigma = 2.7\%$) in image-viewing data, manually annotated by 12 expert coders. The relatively low percentage of fixation samples in Hooge et al. (2017) is most likely a result of their use of infant eye-tracking data. In the subset of the *Lund2013* dataset we use, the percentage of fixation samples, according to the human coders, is 77.8%, somewhat higher than reported by Hooge et al. (2017) (see Table 1). After we exclude samples coded as smooth pursuit (the coders blindly coded the trials and it was thus possible they indicated pursuit in trials that couldn't have contain pursuit) and other categories that we do not include in this study, the remaining samples were coded as around 85% fixation, almost 10% saccades, and 5% PSOs. Note that we only use those excerpts of the trials where both coders identified samples being either fixation, saccade or PSO.

Machine learning algorithms tend to work poorly when facing unbalanced datasets, i.e., with datasets where one or more classes are underrepresented. To alleviate this problem, we have designed a *heuristic data augmentation*

Table 1 Distributions of samples among event classes in our dataset. RA and MN are abbreviations for human coders, who coded the data

Dataset	Fixation	Saccade	PSO	Smooth pursuit	Blink	Undefined	Number of samples	Duration in seconds
Lund2013-image	77.78%	8.93%	4.96%	3.13%	5.03%	0.17%	151639	303.28
Lund2013-image*	84.84%	9.75%	5.41%				136078	272.16
trainSet (6 trials)								
RA	78.42%	9.75%	3.68%	6.90%	0.90%	0.35%	23941	47.88
RA*	85.57%	10.45%	3.98%				21888	43.78
valSet (6 trials in total)								
RA	82.76%	10.69%	5.84%	0.31%	0.40%	0.00%	11973	23.95
RA*	83.35%	10.77%	5.88%				11888	23.78
MN	83.79%	10.57%	5.26%	0.00%	0.38%	0.00%	11973	23.95
MN*	84.06%	10.64%	5.30%				11888	23.78
testSet (22 trials in total)								
RA	74.09%	8.57%	5.01%	4.83%	7.42%	0.08%	51876	103.75
RA*	84.72%	9.63%	5.65%				45207	90.41
MN	78.63%	8.14%	5.24%	1.05%	6.70%	0.24%	51876	103.75
MN*	85.20%	9.03%	5.76%				45207	90.41

*After leaving only the samples, where both coders identified them being either fixation, saccade or PSO. These are the data that we use in this study

procedure to enable gazeGenNet to see more samples of the underrepresented classes during training (see section "Architecture and training protocol"). Furthermore, gazeNet uses a weighted cross-entropy loss function to deal with the majority class bias (see section "Training gazeNet – an end-to-end eye-movement event detector").

Performance evaluation

How to best evaluate the performance of event detection algorithms has been extensively debated in recent years. A common assumption is that human experts can and should serve as a gold standard. If so, the output of any event detection algorithm should be compared to human coders. However, Komogortsev, Gobert, Jayarathna, Koh, and Gowda (2010, p. 2643) write that "this type of classification technique [manual coding] is susceptible to human error and can be open for biased interpretation with limited generalizability". Andersson et al. (2017, p. 619) also voice concerns, saying that "a Human-Algorithm comparison, however, often assumes that humans behave perfectly rationally and that, consequently, any deviation from perfect agreement is due to the mistakes of the algorithm. Thus, a question highly related to this approach is how reliable the human coders are." This question was answered for fixations by Hooge et al. (2017), who found that the majority of their twelve human coders code fixation onsets and offsets almost the same. The coders' settings differed with respect to each other by up to 20 ms on average when coding fixation onsets and up 15 ms when coding offsets. However most of the coders were much more similar than that (see Fig. 9 in Hooge et al. 2017). Some human coders ignored small saccades, while others code for them. It is very likely that PSOs would also be coded differently by different people, which was not investigated by Hooge et al. (2017). Nevertheless, Hooge et al. showed that among twelve expert coders, RA and MN (the same experts as who coded the Lund2013 dataset) are both close to a median expert coder of fixations in almost all respects. For this reason, we were comfortable with using the two human coders RA and MN as a baseline in this study. All mentions of "accuracy" and "performance" in this paper thus refer to how similar the output of the event detector in question is to the hand-coded labels provided by these two expert human coders.

In order to evaluate the performance of our algorithm, we employ several different metrics from the literature and propose a novel event-matching procedure for event level classification evaluation which generalizes the event-based F1-score procedure from Hooge et al. (2017) to handle more than two event classes. These metrics allow us to check whether our algorithm is able to find all the events indicated by the human coders, and also how accurate the algorithm is in labeling the onset and the offset of these events. All evaluations in this paper are performed on the whole-dataset level, calculating a single score based on all events in all trials. To ensure a fair comparison between gazeNet and the two human coders (section "Results"), exactly the same samples were used for all evaluations. Specifically, we only evaluated data excerpts that were coded as fixations, saccades or PSOs by both of the coders. We also exclude the first sample in each excerpt because gazeNet takes differentiated data as input, and this operation reduces the number of samples by one. Last, all excerpts shorter than 101 samples were removed from the dataset because gazeNet was not trained on excerpts shorter than 100 samples of differentiated data. In section "Comparison to other algorithms and datasets", gazeNet was evaluated together with 4 other event detection algorithms on the Lund2013-image-test dataset, genSet and 2 other eye-movement datasets. For this evaluation, none of the data are removed from the evaluation to evaluate the algorithms in a more naturalistic setting and assess how artifacts in the algorithms' output affect their overall performance.

Accuracy paradox In the case of unbalanced data, all accuracy based measures, including very common ones—precision, recall, F1-score—are sensitive to the proportions of samples in each class and suffer from the *Accuracy Paradox*. It implies that a predictive model with a high accuracy might in fact have a low predictive power. For example, the event detection algorithm could predict all the samples to belong to the majority class, in our case—fixations. The sample-to-sample accuracy of such event detector, if measured by any accuracy based measure, will be high and would reflect the sizes of the classes of events rather that the predictive power of an algorithm. It is obvious that such an algorithm would not be useful, despite that the evaluation measure gives it a high score.

A common metric that accounts for the proportions of samples in different classes is Cohen's kappa (Cohen, 1960). Cohen's kappa measures inter-rater agreement and is a number between –1 and 1, where 1 means perfect agreement and 0 means no agreement between the raters, other than what would be expected by chance. In other words, it compares an observed classification accuracy with an expected accuracy (random chance).

For an illustration of the advantage of using the kappa statistic for the evaluation of the performance of event detection algorithm, consider a simple case of the ground truth data, where 80% of the samples are labeled as fixations (F), and the rest 20% are saccades (S), say—FFFFFFFFSS. Now assume the algorithm labels data as FFFFFFFSSS, i.e., mislabels 10% of the samples. The F1-score for such a prediction is 0.93 for the fixations and 0.8 for saccades. That is, the fixation score is punished less because it is the majority class. Either way the F1-scores are still high, despite of 12.5% of fixation samples labeled incorrectly, and the output containing 33% more saccade samples. Cohen's kappa in this case is 0.74. Consider another case, where the algorithm labels data as FFFFFFFFFS, i.e. does not detect 50% of the saccade samples. In this case, the F1-score is 0.94 for the fixations and 0.67 for the saccades, while kappa is 0.62. In conclusion, Cohen's kappa reflects better what damage was done when mislabeling a few samples and suits better for the evaluation of unbalanced data. While in turn the F1-score makes the algorithm look better than it really is. Note that when evaluating the detector's performance at the event level instead of the sample level, the distribution of data across the classes is less unbalanced. Nevertheless, Cohen's kappa is also a robust measure to use in this case, and will thus be used throughout the paper.

Sample-to-sample vs. event-level Cohen's kappa

Following Andersson et al. (2017), Zemblys et al. (2018) and others, we perform sample-to-sample comparison using Cohen's kappa (κ_s). Sample level κ_s indicates the accuracy of the detection of onset and offset of events, but κ_s does not tell the whole story. For example, a single sample in the middle of one fixation misclassified as a saccade would have a minimal impact on the κ_s score, but frequent misclassifications of this nature would have a huge impact on higher level analyses that are usually performed on eye-tracking data. Breaking large fixations into small ones, or missing small saccades and thus merging short fixations into a large one would dramatically change the distributions of fixation duration, mean saccade amplitude and duration, and other eye-tracking measures that eye-movement researches employ in their studies. Therefore we also calculate an event level Cohen's kappa–κ_e.

Novel event-level evaluation

Evaluating the algorithm's performance on the event level is a difficult task because it is unclear how to map the sequence of events predicted by an algorithm to the ground truth events labeled by human experts. Hoppe and Bulling (2016) used a majority vote approach and calculated the F1-score. For each of the ground truth events, Hoppe & Bulling looked at which of the classes comprise the majority of the samples in the algorithm's output for the samples spanned by the ground truth event. Such an approach does not guarantee a reasonable evaluation of event level classification in eye-tracking data. Consider a case where the algorithm splits a long ground truth fixation into three fixations by detecting small saccades in between (see Fig. 1, left). The majority of the samples in such a segment will be fixation samples and therefore the whole segment will be marked as a fixation. Consequently, F1-score or any other metric will show a perfect performance, despite of the algorithm's output being three small fixations and two saccades in between—considerably different from the ground truth.

Hooge et al. (2017) developed another event level F1-score and used it to compare human coders for assessing fixation coding performance. Hooge et al. looked for

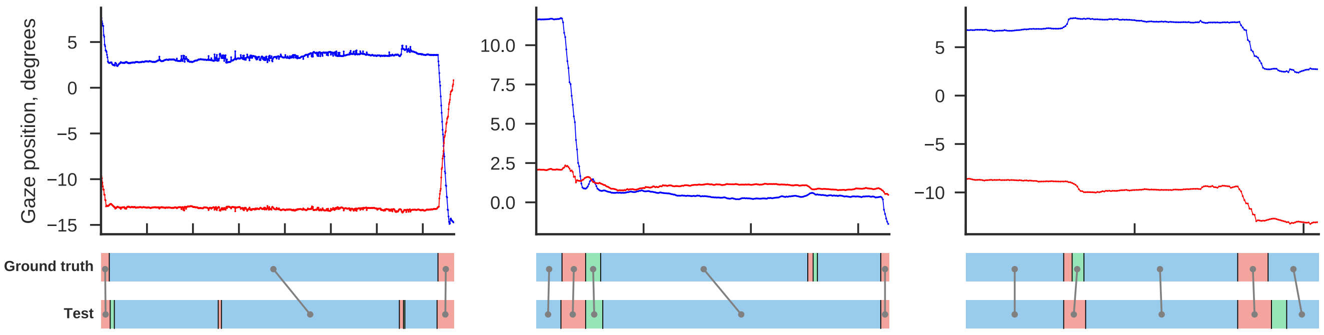

Fig. 1 Examples of the proposed event-matching procedure. *Blue*, *red*, and *green patches* are fixations, saccades and PSOs, respectively. *Blue* and *red lines* are horizontal and vertical gaze data, respectively. *Gray lines* show matched events. *Ticks* on the horizontal axis denote 200-ms intervals

overlapping test-events occurring earliest in time and labeled them as hits. The remaining unmatched ground truth fixations were labeled as misses, while the remaining unmatched test fixations were labeled as false alarms. This works when there is a single event class, but when there is more than one event type to match, the Hooge et al. approach does not work well. Consider a case where the ground truth segmentation consists of a saccade, followed by a fixation, while the algorithm finds a PSO after that saccade (see the second saccade in Fig. 1, right panel). The Hooge et al. event-matching procedure would match the ground truth fixation with the algorithm's PSO, because these are the events that overlap earliest in time. A modification to this approach could be to first look for matching fixations, then for saccades and so on. However this way we would unfairly assist the algorithm to appear better in the evaluation by asking the question "whether the algorithm detected the ground truth event" rather than "how well the algorithm classifies data". Consider the case where a segment of eye-tracking data is labeled as a fixation. Then at the beginning of that segment the algorithm labels a few samples to be a fixation, and the rest of the segment to be a saccade. Both, the original (Hooge et al., 2017) approach, and the hypothetical modified one, where we would first match fixations, then saccades, etc. would mark a hit and a false alarm. While in fact the algorithm's prediction is that the considered segment is a saccade, i.e., the algorithm does not only fail to detect the fixation, but also mislabels data as a completely different event.

In order to overcome the issues with the event-matching procedure by Hoppe and Bulling (2016) and generalize the method of Hooge et al. (2017), we propose using a combination of both. Instead of looking for overlapping test-events occurring earliest in time, for each ground truth event, we look for the algorithm event that has the largest overlap, and then match the events in the two streams. If the matched events are of different classes, the match is marked as a false positive or a false negative (depending on which event is evaluated), and if the events are of the same class, as a true positive (hit). The remaining unmatched events are then labeled as false positives or false negatives, depending on whether they, respectively, occur in the ground truth or the algorithm event stream. Using our approach in the example where the ground truth segmentation consists of a saccade, followed by a fixation, and the algorithm finding a PSO after that saccade (second saccade in the right panel of Fig. 1), the ground truth and algorithm's fixations get matched because they overlap more, while the algorithm's PSO is left unmatched and appears as a false positive in a subsequent calculation of the event level performance score.

Using our proposed approach for matching the longest overlapping events penalizes inaccurate performance at both the sample level and the event level classification. For example, let us say that the human expert labels a short saccade and a longer PSO after that saccade, while the event detection algorithm labels all these samples to be a saccade (see first saccade in the right panel of Fig. 1). Our event-matching procedure will match the experts PSO with the saccade of the algorithm, because these two events will have the longest overlap. As a result, the algorithm is penalized for two errors, 1) not detecting a PSO after the saccade and 2) mislabeling the PSO as a saccade. Technically our procedure treats this situation as if the algorithm did not detect a saccade. Although this is simply is not true, the algorithm just missed a PSO and mislabeled those samples as part of the saccade, we did not want to include more heuristic decisions to our matching procedure and help the algorithm to appear better. Another example where the event level performance score is heavily penalized is when the event detection algorithm splits or merges a fixation (see left and middle panels in Fig. 1). If, for example, the algorithm splits a ground truth fixation into two separate fixations by detecting a false saccade in the middle, our event-matching procedure will match the ground truth fixation with the longest fixation in the algorithm's output. As a result, the other fixation will be treated as a false positive, and together with a false-positive saccade will degrade the algorithm's performance score. Similarly, if the algorithm would merge two ground truth fixations by not detecting a saccade between them, the performance score will be penalized by adding two (or three, if there is a PSO after the saccade in the ground truth) false negatives. Python code implementing the event-matching procedure and computation of the event level Cohen's kappa (κ_e) is available from https://github.com/r-zemblys/ETeval.

Sample and event error rates

Cohen's kappa can be hard to interpret, and therefore we propose to use two more natural measures, that allow to compare sequences of different length: *Sample Error Rate (SER)* and *Event Error Rate (EER)*. These two measures are inspired by the Character and Word Error Rates in Automatic Speech Recognition systems. They are calculated as the Levenshtein (edit) distance[2] between two sequences, normalized for length. If both compared sequences have equal length, SER and EER both report the percentage of misclassified items, or in other words—classification accuracy. This is always the case for SER, as it performs sample-to-sample comparison. For EER, the length of compared sequences can differ, and therefore EER reports the number of edits needed to match human coders to the event sequence from the algorithm.

The advantage of using SER and EER is that they are easy to interpret and in contrast to κ_e, no hard decisions, such as whether to match largest or first overlap, are needed in its implementation. The disadvantage of both SER and EER, is that they suffer from the accuracy paradox (see discussion on page 5). We however report SER and EER together with the sample and event level Cohen's kappa for the quantitative evaluation of our proposed algorithm, as these measures together allow us to avoid the accuracy paradox that may bias the evaluation of our highly unbalanced testing dataset, and at the same time, present the accuracy of the algorithm in an easily interpretable way.

Event-timing evaluation

Hooge et al. (2017) proposed using relative timing offset (RTO) and relative timing deviation (RTD) measures, which capture the systematic relative difference and variance of that difference between settings of two coders. Hooge et al. present RTO and RTD as the missing links between agreement measures such as the F1-score or Cohen's kappa and the eye movement parameters. The advantage of RTO and RTD is that they show that two classifiers may produce similar events (high agreement), but differ in the detailed timing of their settings. RTO is calculated as the temporal difference between onset or offset of two matched events (hits), while RTD is the standard deviation of the RTO measurements.

We include these measures in our paper for a complete picture of gazeNet's performance, however we should note that because of the event-matching procedure we use in this paper, RTO and RTD measures can result in worse overall figures, compared to using the original approach of Hooge et al.. Take for example an extreme case, where the event detection algorithm splits a ground truth fixation into 3 parts, with the middle one being the longest (see right of Fig. 1). Our event-matching procedure will match the ground truth fixation with the second fixation in the algorithm's output. As a result, the relative timing offset of both the onset and the offset of that fixation will be very high. Similarly, the RTD measure will also increase, as it measures the spread of RTO across the whole dataset. (Hooge, Holmqvist, & Nyström, 2017) used a different matching procedure when calculating RTO for onset and offset of the events—for the onset they matched fixations overlapping earliest in time, while for the offset RTO calculations matching of fixations was done in the opposite order instead matching the last overlapping event. We, however, wanted to have the same event-matching procedure for all the calculations to avoid assisting the algorithm to appear better.

Generating synthetic eye-tracking data

The Lund2013-image dataset is much too small for training a deep learning model. Even a small network would overfit the training data and that will not be useful when detecting events in new data. Hoppe and Bulling (2016) used a manually annotated dataset consisting of 1626 fixations, 2647 saccades and 1089 pursuit movements (around 22 minutes of eye-tracking data) to train a single-layer convolutional neural network with max pooling and a classification layer on top to predict the event class for every sample (75% of this dataset was used for training). In comparison, the Lund2013-image dataset only consists of around 5 min of eye-tracking data with 951 fixations, 911 saccades and 719 PSOs when counting the coding by the two coders on the same trials separately (see Tables 1 and 3), and the network we aim to train is much deeper.

Getting enough data to train neural networks has been a challenge in many other areas. While data for tasks like image classification or speech recognition can be relatively

[2]See here https://nlp.stanford.edu/IR-book/html/htmledition/edit-distance-1.html for a thorough description on calculating Levenshtein distance.

easy acquired through crowd-sourcing, reliably coding long sequences of eye-tracking data requires experts in eye-movement research, is very expensive, and yet, as discussed above, does not guarantee the perfect outcome.

Methods to *synthesize* eye-movements already exist, but they are almost exclusively developed for use in graphics applications, where animation of the eyes in a virtual character is desired (e.g. Lee, Badler, & Badler, 2002; Ma & Deng, 2009; Yeo, Lesmana, Neog, & Pai, 2012). Only one study mentions the possibility to use the synthetic data for testing event detectors (Duchowski, Jörg, Allen, Giannopoulos, and Krejtz, 2016). Also, all these synthesizers are algorithmic models of eye movements built using features borrowed from eye-movement research. Hence, the produced data do abide to the selected research, but do not necessarily have the same signal properties as authentic eye-movement recordings, which is what we want to use for training. In addition, there are advanced models that generate eye rotations (e.g. Enderle & Zhou, 2010) based on the knowledge of the oculomotor system in the brain. In contrast, a pupil-CR eye-tracker produces a signal that is the difference in position of the P-CR feature pair, which in a non-linear manner relates to eye orientation. This signal also contains PSOs as a result of movement of the lens inside the eye ball, amplified by the interplay of the pupil and CR signals (Hooge, Holmqvist, & Nyström, 2016). In other words, models may generate correct eye-ball movements but do not generate an authentic eye-tracking signal as would be recorded from a real eye.

Our approach to generating artificial eye-tracking data is to employ a recurrent neural network for extending the Lund2013-image dataset by using it to train a generative neural network that we call *gazeGenNet*. With this approach, we aim to reproduce both the eye-movement characteristics in the dataset, including PSOs, but also its signal properties.

Recurrent neural networks

Recurrent neural networks (RNN) have been shown to be able to generate realistic images pixel by pixel (Van Den Oord, Kalchbrenner, & Kavukcuoglu, 2016), have shown excellent performance in character-level language modeling (Sutskever, Martens, & Hinton, 2011), handwritten text generation (Graves, 2013) or sample level audio generation (Mehri, Kumar, Gulrajani, Kumar, Jain, Sotelo, . . . , Bengio, 2016). Generating synthetic eye-movement sequences is no different from such examples: It can be seen as a 2D regression problem where we have an input vector g_t which consists of all historical pairs of gaze coordinates $[x_t, y_t]$, from which we want to predict a future vector g_{t+1} of the coordinates of next gaze position. We can then feed g_{t+1} back to the network, predict the next vector g_{t+2}, feed it back to the network, and iterate this forward-stepping procedure ad infinitum.

A central advantage of such recurrent neural networks is that every generated sample depends not only on the one previous sample, but on all the previously generated samples. This allows the network to model long-term dependencies, such as learning to generate PSOs only after saccades. Another great advantage of using RNNs in generating eye-movement data is that these networks do not use exact templates extracted from training data, but rather their internal states to make predictions. That is, RNNs reconstruct the training data in a complex way, and rarely generate the same sequence twice but instead generate data with a certain amount of variability, as would be seen in authentic eye-tracking data recorded from human participants.

Architecture and training protocol

Our synthetic eye-tracking data generation network is inspired by the handwritten text generation network by Graves (2013): a sequence-to-sequence LSTM (Hochreiter & Schmidhuber, 1997) with a Mixture Density Network (Bishop, 1994) as an output layer. A Mixture Density Network is in fact nothing else than a *Fully Connected* layer, but instead of directly using its outputs to make predictions about the next gaze sample, the network outputs a parameterized Mixture of Gaussians, consisting of parameters for the probabilistic distribution of the location of the next gaze sample. Intuitively, this Mixture of Gaussians could be interpreted as the number of choices the network has for the next raw data sample, given all the inputs (samples) so far. We implemented gazeGenNet in Tensorflow.[3] A more detailed architecture description can be found in Table 2.

Although normalizing input data is not a necessary requirement for training a neural network, normalized input (and also normalized input for each internal layer) leads to a faster convergence (Ioffe & Szegedy, 2015). To achieve that and following (Graves, 2013), we use gaze offset (velocity per sampling interval) instead of a position signal as an input. In addition during the training we randomly either swap or not horizontal and vertical channels, so our network sees more variations of data.

Our training data are highly unbalanced, which poses a great challenge for classification tasks because of the so-called accuracy paradox (see on page 5). If that happens, the model could predict all test samples to belong to the majority class, in our case fixations. The whole generated dataset might end up as a single fixation. To avoid this,

[3]We used the starter code from https://github.com/hardmaru/write-rnn-tensorflow

Table 2 Architecture and hyper parameters for gazeGenNet. T - timesteps in the sequence, M - number of Gaussian Mixture components, N - number of event classes

Architecture		
RNN	Type	LSTM
	Layers	3
	Neurons	128
	Bidirectional	False
	BatchNorm	False
	Dropout	0.25
FC	Layers	1 (readout)
	BatchNorm	False
	N	3
	M	20
Hyper parameters		
	Optimizer	RMSprop
	Grad clip	10
	Learning rate	0.001
	Batch size	50
	T	100
	Training steps	2000

we perform *heuristic data augmentation*, which allows us to generate realistic and more frequent saccades and PSOs amongst the fixations. First, we split our training set into short trials (each 102 samples long or 201 ms at the data's 500 Hz sampling frequency), such that every trial includes a saccade. Second, we perform *saccade amplitude histogram equalization*, i.e we include large saccades–which occur less often–multiple times in each training epoch, such that each saccade amplitude occurs equally often. Such a heuristic data augmentation ensures that during the training, our generative network sees more saccade samples (17.59% compared to 10.6% in the original training set), and equal number of saccades for each amplitude, and no sequences with only fixation samples. Note that the goal is not to balance the dataset at a sample level, but rather balance the number of saccades with different amplitudes. Consequently, augmentation pushes gazeGenNet away from generating long fixation sequences and helps it learn to generate more larger amplitude saccades that it would do otherwise if non augmented training data would be used.

Formally, the network input g_t is a vector (dx, dy, e), where dx, dy indicate the offset in gaze from the previous sample and e (also referred to as $(g)_3$ in the equations below) is a binary input, that has value 1 at the offset of an event (fixation, saccade or PSO) and 0 otherwise.[4] This simple coding scheme is sufficient because the end of one event is the start of the next, as there are no holes of non-events in the data.

The output of the network y_t is a vector which consists of the end of event probability e, along with a set of means μ^j, standard deviations σ^j (both vectors with two elements), correlations ρ^j and mixture weights π^j (both scalars) for the M multivariate (2D) mixture components and probabilities o^i of the input sample belonging to one of the N oculomotor events. Mathematically, this can be represented as follows:

$$y_t = \left(e_t, \left\{\pi_t^j, \mu_t^j, \sigma_t^j, \rho_t^j\right\}_{j=1}^{M}, \left\{o_t^i\right\}_{i=1}^{N}\right) \tag{1}$$

The event type for each sample (fixation, saccade, or PSO) is determined by picking the event with the highest probability in o^i. The probability density $Pr(g_{t+1}|y_t)$ of the next input g_{t+1} given the output vector y_t is defined by Eq. 2 Graves (2013). The next input sample g_{t+1} is then generated by sampling from this probability density.

$$\Pr(g_{t+1}|y_t) = \sum_{j=1}^{M} \pi_t^j \mathcal{N}(g_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j) \begin{cases} 1 \text{ if } & e_t > \tau \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

Here τ is a *decision boundary*—a threshold above which the output value e_t will signal that the ongoing event is to be terminated. We choose this threshold to be 0.5.

We train gazeGenNet through backpropagation using the RMSProp algorithm[5] (Graves, 2013). The sequence loss function $\mathcal{L} = \mathcal{L}_g + \mathcal{L}_e + \mathcal{L}_o$ that is optimized during the training process is the sum of 3 separate loss functions, which are defined as follows:

$$\mathcal{L}_g(\mathrm{g}_1, \mathrm{g}_2) = \sum_{t=1}^{T} -\log\left(\sum_j \pi_t^j \mathcal{N}(g_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j)\right) \tag{3}$$

$$\mathcal{L}_e(\mathrm{g}_3) = \sum_{t=1}^{T} - \begin{cases} \log e_t & \text{if } \ (g_{t+1})_3 = 1 \\ \log(1 - e_t) & \text{otherwise} \end{cases} \tag{4}$$

$$\mathcal{L}_o(\mathrm{X}, \mathrm{O}) = -\frac{1}{T}\sum_{t=1}^{T} o_t \ln x_t + (1 - o_t) \ln(1 - x_t) \tag{5}$$

In Eqs. 3, 4 and 5, T is time steps in the sequence, X in Eq. 5, refers to the event labels provided by the human coders, and O the predicted oculomotor event labels. $\mathcal{L}_g$ (Eq. 3) optimizes the parameters of the Gaussian mixtures, from which the next samples are drawn. $\mathcal{L}_e$ (Eq. 4) is responsible for terminating the ongoing event, while $\mathcal{L}_o$ (Eq. 5) is a cross-entropy loss function, which optimizes the

[4]In the actual implementation we had an additional input value, that was randomly selected to be 0 or 1. In case this extra input was 0, the end of event flag in the sequence was set to 0.

[5]RMSprop is an adaptive learning rate method first proposed by Geoff Hinton in his course: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.eps

predictions of the event class by comparing the generated events to the hand-coded ones.

We train this network for 2000 steps (210 epochs), until the loss value levels out. We do not use any validation here and hence we are most likely overfitting. However, the goal is not generalization. We wish to generate a large amount of synthetic data which are very similar to the training dataset, while still exhibiting considerable variation in fixation durations, saccade and PSO amplitudes, shapes, etc.

Results

After training the network, we generated 2000 synthetic eye-movement trials, each 5000 samples (10 s) long. When generating artificial samples, we restrict maximum fixation duration to 500 ms by manually changing $(g)_3$ (e_t, see Eq. 2) to 1 if fixation durations exceeds this threshold. Further, we removed all fixations shorter than 36 ms, all saccades shorter than 6ms (we obtained these thresholds from our hand-coded validation set), all saccades larger than 30°, and PSOs that do not occur after the saccade. That lead to removing of 10396 fixations (14% of all generated fixations), 14548 short and 72 large saccades (21% in total), and 5588 PSOs (14%; 4630 of these PSOs were originally following fixations, while the rest of them were following saccades that got removed because of not meeting the above criteria). The majority (78%) of the removed fixations had durations of four or less samples, while most of the short saccades (73%) were only 1 sample long. The largest saccade that got removed had an amplitude of 47 degrees, while most of removed large saccades were in in the range of 30–40 degrees. In addition, all sequences that are shorter than 101 samples were also removed.

Although the number of removed events seems to be quite large, this is not an indication of poor performance of gazeGenNet, but rather a result of our choices of how to interpret its output. First, our sampling process of the synthetic data is probabilistic, i.e., we sample the position (or rather the offset from the previous gaze sample) of the next gaze sample from the mixture of Gaussians generated by the network. It is then very possible that for example the network's *intention* is to start generating a fixation, therefore it labels a sample as such. However when we sample the position, it might be more similar to a saccade. When we feed back such sampled offset to the network it might take several steps until the network switches to another state and starts labeling samples as saccade, thus resulting in a short fixation before it. And second, the way we handle the *end of event flag* (e_t in Eq. 2) also affects the final output. We set e_t to 1 if it is higher than 0.5, signaling the network that the ongoing event needs to end. If for example the network was generating the fixation and we signal to terminate it, it is not necessary that the network will do that. gazeGenNet relies not only on the input, but also on its internal state, therefore it might continue generating the fixation. Then the next time we signal to terminate the ongoing event, the network might start generating PSO, because it has already seen two *end of event* flags. We conclude that deeper investigation is needed to determine why the gazeGenNet generates too short or too long events and the other issues flagged above, and posit that this is a very worthwhile area

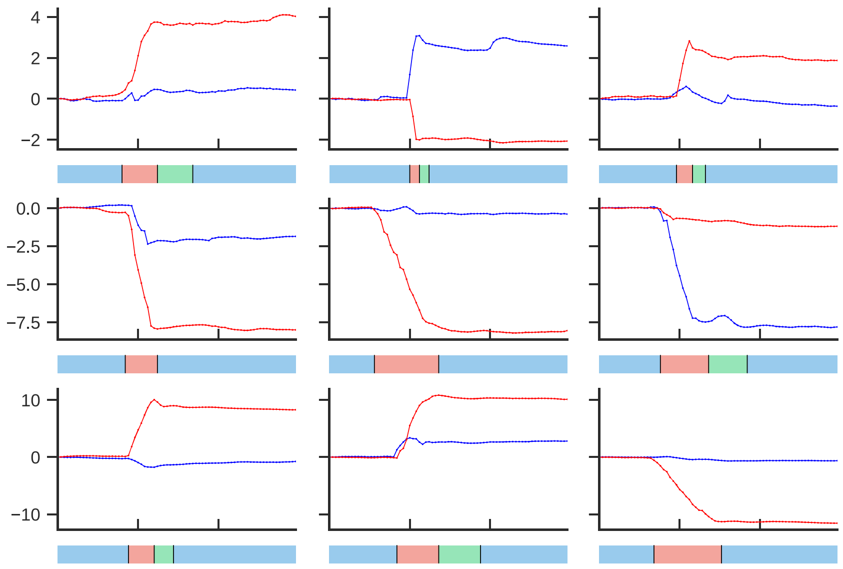

Fig. 2 Examples of synthetic data with time on the x-axis and position (in degrees) on the y-axis. *Ticks* on time axis denote 50-ms intervals. Horizontal gaze position - *blue*, vertical - *red*. *Red blocks* in the scarf plot underneath each panel indicate saccades, *green* - PSOs, and *blue* are fixation samples as labeled by the gazeGenNet

Table 3 Number of events in datasets. RA and MN are human expert coders, who coded eye tracking data. genSet is the synthetic dataset generated by gazeGenNet

Dataset	Fixations	Saccades	PSOs
Lund2013-image	951	911	719
(Hoppe & Bulling, 2016)	1626	2647	–
trainSet RA	171	174	106
valSet RA	92	90	78
valSet MN	90	88	69
testSet RA	296	279	228
testSet MN	302	280	238
genSet	62055	56264	34342

of study to solve the problem of generating the large datasets required for training deep learning-based event detectors. However, a further exploration is out of the scope of this paper. The sole purpose of gazeGenNet in our study is to generate gargantuan amounts of artificial data in order to deal with the overfitting problem when training gazeNet. The performance of the resulting event detector trained on this synthetic dataset underscores that the generated artificial data were indeed good enough to create a well-working event detector and this serves its purpose for the current investigation.

Figure 2 shows examples of raw data generated by our network. After having shown synthetic eye-tracking data side-by-side with real data to more than 100 eye-movement researchers at seminars, workshops and conferences and having obtained chance-level discrimination performance from this audience, we think it is safe to say that they look very realistic. Indeed, in the generated data, one can easily recognize saccades and PSOs, and also spot some drift and noise in the fixations, and even noise during saccades. Although not relevant for the event detection task, the scanpaths of the synthetic data trials seem to be very typical for an image viewing task (see Fig. 13 in Appendix D). Considering that trials were generated sample by sample, it confirms that the network was able to learn long-term dependencies and is able to generate realistically looking artificial eye-tracking data. To confirm this impression, in the rest of this section we analyze the properties of this synthetic eye-tracking data.

The above procedure resulted in a total of over 5 h of high-quality labeled artificial eye-movement data at 500Hz (hereafter *genSet*), which we will use to train gazeNet, the end-to-end deep learning-based event detector. Despite our use of data augmentation, the data generation network is still more likely to output fixation samples: 89.61% of this artificial data are labeled as fixations, 7.78% as saccades and only 2.61% as PSO. Our genSet has 62055 fixation, 56264 saccade and 34342 PSO events (see Table 3). In comparison to Hoppe and Bulling (2016) our genSet,

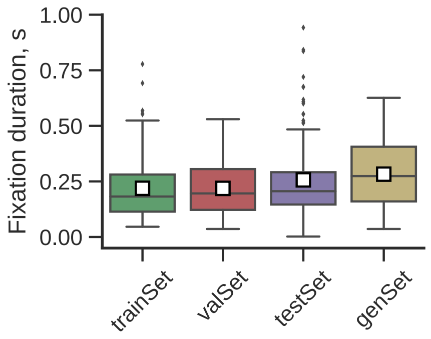

Fig. 3 Fixation durations in the datasets. White squares denote averages. Note that here we aggregate the data from both coders

totaling 326 min of eye-tracking data, has around 15 times longer duration, has 40x more fixations and 20x more saccades. Compared to our original hand-coded training data, we extended it by a factor of 450x.

In Fig. 3 fixation durations in all datasets are presented. Note that here we stack data from both coders. Average fixation duration in genSet is a bit higher, compared to that of training and validation sets—283 ms compared to 219 ms. However it covers a broader range of fixations durations. The testSet is in between with an average fixation duration of 257 ms. Around 15% of fixations in genSet have durations longer than the 500-ms threshold we set when generating the data. This means that gazeGenNet does not always rely on only the *end of event flag* (e_t in Eq. 2), but also on its internal state to generate the next sample. However the vast majority of these longer fixations are in the range of 500-510 ms, while only 21 fixations were longer than 510 ms.

Table 4 presents data quality measures of our datasets. Besides the root mean square of the sample-to-sample (RMS-S2S) and standard deviation (STD) measures, we also report magnitude ($\sqrt{RMS^2 + STD^2}$) and $\frac{\text{RMS}}{\text{STD}}$ measures (Holmqvist, Zemblys, & Beelders, 2017). We calculated these four measures using the procedure of Hooge et al. (2017), i.e., we slide a 200-ms window along each fixation (as indicated by the hand-coded or generated event labels), calculate these measures for each window and take the median. Note that this means that all fixations less than 200 ms are excluded from the precision evaluation. The values presented in Table 4 are averages of quality measures from all fixations. Compared to the training and validation

Table 4 Eye-tracking data quality measures

Dataset	RMS (°)	STD (°)	$\frac{\text{RMS}}{\text{STD}}$	Magnitude (°)
trainSet	0.027	0.121	0.221	0.124
valSet	0.026	0.138	0.190	0.140
testSet	0.038	0.150	0.256	0.155
genSet	0.029	0.146	0.200	0.149

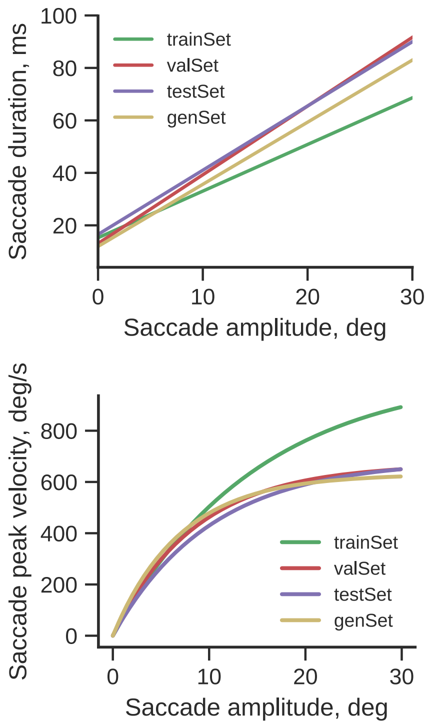

Fig. 4 Saccade amplitude–duration relationship (*top*) and main sequence relationship (*bottom*)

sets that were used to train gazeGenNet, genSet has slightly higher average RMS, STD and therefore magnitude values. However, the $\frac{\text{RMS}}{\text{STD}}$ value (0.2) is somewhere in between the two datasets, which means that gazeGenNet was able to produce the same type of noise (termed *trailing* by Blignaut & Beelders, 2012; Holmqvist et al., 2017) as was present in the original SMI Hi-Speed 1250 data.

Figure 4 shows the saccade amplitude-duration relationship and main sequence fits for all datasets. Interestingly, both the saccade amplitude-duration and the main sequence fits of genSet deviate from the data that gazeGenNet was trained on, but are very similar to fits of the validation and testing sets.

Compared to the training and generated data sets, the validation and testing sets have a larger number of saccades that are followed by PSOs—83% compared to 61% in training and synthetic sets (see Table 3). In addition to these differences in data, the metadata for confidence and tagtime per trial for the trainSet also differ from the validation and testing sets. Clearly, RA tagged events in those trials differently, more slowly, and indicated feeling less sure about his coding, but we do not know why.

We judged our labeled synthetic data to to be similar enough to the human eye-movement data that gazeGenNet was trained on for it to be usable for training the event detector. Note that, in the end, it is the performance of the eventual *gazeNet* event detector that is the final judge. gazeNet will be trained on these synthetic training data but evaluated against real manually annotated data. That evaluation will tell us whether our gazeGenNet yields input data of sufficient quality for training an event detector through deep learning.

Table 5 Architecture and hyper parameters for gazeNet. T - timesteps in the sequence, N - number of event classes

Architecture		
CNN	Type	“same”
	Layers	2
	Filters	8
	Kernel	2x11
	Stride	[1, 1]
	BatchNorm	True
	Dropout	0.25
	Activation	ReLU (clipped at 20)
RNN	Type	GRU
	Layers	3
	Neurons	64
	Bidirectional	True
	BatchNorm	True
	Dropout	0.25
FC	Layers	1 (readout)
	BatchNorm	False
	Bias	False
	N	3
Hyper parameters		
	Optimizer	RMSprop
	Learning rate	0.001
	Batch size	100
	T	100

Training gazeNet—an end-to-end eye-movement event detector

Our gazeNet architecture was inspired by Deep Speech 2, an end-to-end speech recognition neural network (Amodei, Anubhai, Battenberg, Case, Casper, Catanzaro, . . . , Zhu, 2015). gazeNet was implemented using the pyTorch[6] neural network framework (version 0.2.0_4) and the starter code from Sean Naren.[7]

Our network has two convolutional layers followed by three bi-directional recurrent layers with a fully connected layer on top. The convolutional layers use 2D filters with a size of 2×11 and are meant to extract deep features from raw input data, while the recurrent layers model event sequences and are responsible for detecting onsets and offsets of fixations, saccades and PSOs. Instead of the more

[6]https://pytorch.org/

[7]https://github.com/SeanNaren/deepspeech.pytorch

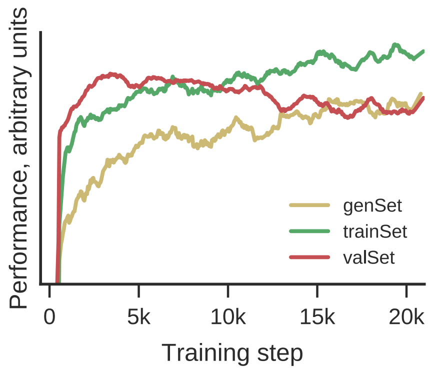

Fig. 5 Smoothed training performance curves for gazeNet. Note that when training gazeNet, the synthetic labeled data of genSet is used as training data. Only a subset (5%) of genSet was used to draw this training curve

conventional LSTM layers, we used gated recurrent unit (GRU) layers (Cho, van Merrienboer, Gülçehre, Bougares, Schwenk, & Bengio, 2014), because these usually exhibit the same performance with fewer parameters. A more detailed architecture description can be found in Table 5.

In the early stages of developing gazeNet, we assessed other architectures involving deeper and shallower networks with double and half the training parameters we used in this study and with different hyper-parameters (optimizer, learning rate, sequence length, etc.). As we did not perform a systematic analysis of these different architectures later on, and the aim of this paper was to provide a framework for how to approach building a deep learning-based event detector, the performance of gazeNet could likely be tweaked further by varying these hyper-parameters.

During the training we ran a validation every 50 steps, and used a L^2-norm composed from all four sample and event accuracy measures—S to determine the model's performance (see section "Performance evaluation" and Eq. 6).

$$S = \|\{\kappa_s, \kappa_e, 1 - SER, 1 - EER\}\|_2 \qquad (6)$$

We selected the model (for subsequent testing on the testSet) which had the highest validation score S. During the training we randomly added white noise from a range of [0, 0.5) degrees RMS to each of the sequences in the batch to further augment the training data and enable the gazeNet to work on noisier data. To deal with the majority class bias, gazeNet was trained using a weighted cross-entropy loss function, with the class weights calculated as 1 minus the fraction of each sample class in the trainSet (RA* in Table 1).

Smoothed training performance curves for the different datasets are plotted in Fig. 5. Note that gazeNet is trained using genSet as the training data, and not trainSet which was only used to generate genSet. Training curves are very typical for neural networks. Notice that the training curve (Fig. 5, yellow curve) based on the genSet data keeps increasing as training progresses, while the validation curve increases only up to a certain point, and then starts to decay: this is indicative that the network starts to overfit the data. The best performance on valSet was reached on the forth epoch after around 4k training steps. Unsurprisingly, the curve of trainSet increases together with genSet, because this was the data, based on which the genSet was generated. The similar trends in these two curves further speaks to the fidelity of our generated eye-movement dataset.

Results

Figure 6 shows examples of gazeNet's performance on the testing set. The left part of Fig. 6 shows almost perfect classification performance, compared to two human expert coders. One can spot very minor disagreements in the onsets and offset of the events, but all within a range of a few samples (2-4 ms).

The right side of Fig. 6 demonstrates cases where gazeNet does not agree with the human coders. The predicted PSOs after saccades #1 and #2 have shorter

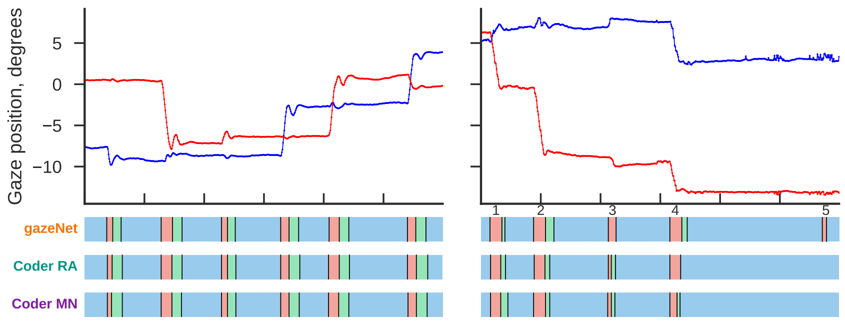

Fig. 6 Examples of data from coder RA and MN, together with events detected by gazeNet on the testing dataset. The *left panel* shows excellent performance of gazeNet, while the *right panel* shows examples of disagreement between gazeNet and the human coders. *Ticks* on the horizontal axis denote 200-ms intervals, the *vertical axis* shows position in degrees, the *blue line* indicates the horizontal gaze position and *red line* the vertical. The *blue*, *red*, and *green patches* in the scarfplot are fixations, saccades, and PSOs, respectively

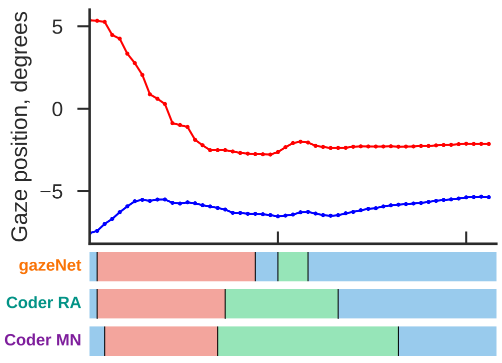

Fig. 7 gazeNet makes a mistake by misclassifying a few PSO samples as fixation, resulting in a PSO that follows a fixation. *Ticks* on the horizontal axis denote 50-ms intervals, the *blue line* indicates the horizontal gaze position, and *red line* the vertical. The *blue*, *red*, and *green patches* in the scarfplot are fixations, saccades and PSOs, respectively

and longer durations compared to coders RA and MN. Such disagreements will show up in the sample-based performance metrics below, but not the event-based metrics. Further, compared to human coders, gazeNet fails to detect a PSO after the 3rd saccade and tags part of the fixation as PSO after saccade #4. These disagreements will be counted as False Positives or False Negatives in both the sample- and the event-based performance metrics. And finally, gazeNet splits a long fixation into halves by tagging what appears to be noise as a small saccade, #5. Such a misclassification also hurts both the sample- and the event-based performance metrics.

It is worth noting however, that the human coders do not fully agree on the duration of PSOs following saccades #1, #2, and #3, and even on whether saccade #4 is followed by a PSO or not. This suggests that there is uncertainty about how to interpret these epochs in the data. Digging deeper, the authors think that the PSO after saccade #3 is an unclear case even if both coders tag a PSO here, because data after this saccade does not show an oscillating pattern, which is a common descriptive feature of the PSO. The authors have examined this case in a 2D plot (see Fig. 12 in Appendix C), and concluded that it is indeed very hard to tell whether what the experts coded as a PSO is indeed a PSO, or for example, the oculomotor delay in the vertical channel. As it impossible to tell who is right, perfect agreement of the algorithm with either of the coders is neither expected, nor should it be the goal. Especially for PSO events, disagreement between the coders and between the coders and gazeNet may be expected.

To enable gazeNet to learn temporal properties of the events, we have employed a recurrent neural network. In our case, the important temporal feature is that PSOs cannot occur after a fixation. We have however found one case in the testing data where gazeNet made a mistake and misclassified a few PSO samples as fixation (Fig. 7). That results in small fixation after a saccade, followed by a PSO. When comparing to human coders, this particular misclassification will degrade both sample and event level scores. In the confusion matrix analysis we provide below, this case will appear as a false-positive fixation. As any machine-learning-based event detector can *never* guarantee a certain situation will not occur in its output, such as a PSO after a fixation, we therefore encourage users to always check for such occurrences and deal with them as they see fit. It is however encouraging to see that only one such failure occurred in our output.

In Figs. 8 and 9, sample and event level confusion analyses are presented. On the sample level (Fig. 8), gazeNet's performance is nip-and-tuck with that of human coders for fixation classification. In saccade classification, gazeNet achieves 93-95% accuracy (agreement with the expert coders), compared to a mere 90% agreement between the two human experts themselves. In comparison, Hoppe and Bulling (2016) report 78% for fixations and 37% for saccades.[8] Sample level PSO detection accuracy is lower, but still at a human expert level—gazeNet scores 73% when compared to coder MN, and almost 76% when compared to coder RA, which is close to when the two expert coders are compared against each other (79%).

The event level confusion analysis in Fig. 9 also shows that the fixation and saccade detection performance of gazeNet is just as good as that of human experts and, in some cases, even better (in the sense that the algorithm's output agrees more with a human expert than the agreement between the experts themselves). For example, when compared to coder RA, coder MN misses two fixations and two saccades, while gazeNet misses three saccades and only one fixation. If coder MN is taken as a ground truth, then coder RA misses six fixations and six saccades, while gazeNet only misses three fixation and six saccades. The main difference in detecting events however again lies in PSO detection. Coders RA and MN tag 18 and 27 PSO events in the testSet, respectively, that the other coder does not code as PSO. gazeNet tags 37 false-positive PSOs when compared to coder RA, and 28 false positives when compared to coder MN. The number of false-negative PSOs when gazeNet's output is compared to both of the coders is the same—24. The higher number of gazeNet's false-positive PSOs raises the question whether PSOs detected by gazeNet are actually false positives, or whether the human coders have trouble identifying these events in raw data and have missed some PSO events that gazeNet identified. We did not want to act as metacoders, adding an additional layer of evaluation to this study; however, we provide example

[8]The Hoppe and Bulling (2016) data include smooth pursuit, therefore these numbers cannot be directly compared.

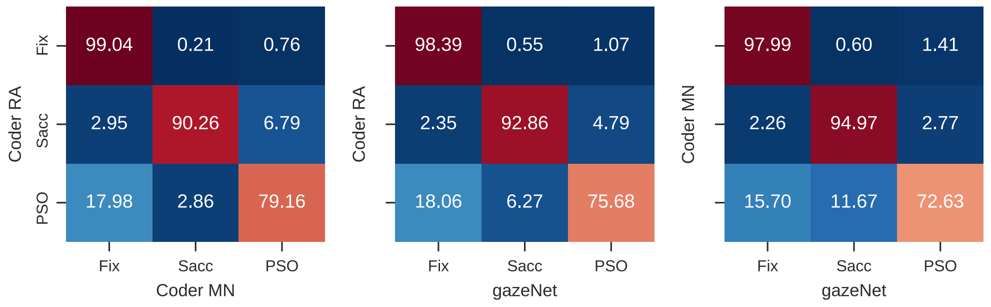

Fig. 8 Sample level confusion matrices. *Numbers* are normalised per row and show the percentages of correctly and incorrectly classified samples

plots of ambiguous PSO cases in Appendix E that allow the reader to form his/her own opinion.

In Table 6, sample-level Cohen's kappa (κ_s) and sample error rate values are presented for the human coders and gazeNet. Just like Figs. 8 and 9, these analyses show that overall the performance of gazeNet on training and validation sets is virtually the same as that of expert coders, and that its performance on the testing set is just a bit lower. On the testing set, gazeNet achieves a κ_s of 0.9 for fixations, which is on the higher end of human expert coder performance. In comparison, Hooge et al. (2017) reported that the sample-level Cohen's kappa when 12 experts coded fixations was in the range of 0.74–0.92. The values in Table 6 replicate the finding by Andersson et al. (2017) and Zemblys et al. (2018), that, like the two human coders, our algorithm performs best in classifying saccade and fixation samples, while the performance of PSO detection is considerably poorer. That again suggests that PSOs are difficult events to classify and probably more research is needed to define what a PSO is in eye-tracking data, and how the PSO should be treated.

Overall on the sample level, gazeNet agrees slightly more with coder RA, probably because it was trained using coder RA's data. At the event level (Table 7), however, these differences disappear and gazeNet appears to agree slightly more with coder MN instead. Event-level analyses also show that gazeNet's performance is at the level of human experts—the overall event-level Cohen's kappa (κ_e) for experts is 0.9, while gazeNet achieves κ_e of 0.86–0.87 for the testing set. In their comparison of 12 human coders, Hooge et al. (2017) only report event level F1-scores for fixations, which are in the range of 0.88–0.97. To compare gazeNet's fixation detection performance on our testing sets to Hooge et al.'s values, we also computed an event level F1-score for fixations. gazeNet's F1-score was 0.98, which is the same as the F1-score of the expert coders in this dataset.

We also examined the RTO and RTD of gazeNet's event detection results following (Hooge et al., 2017) and shown in Fig. 10. Compared to the human coders RA and MN, the onset of fixation of gazeNet is delayed on average by just under 4 ms. Fixations as detected by gazeNet on average terminate earlier and therefore saccades begin earlier, however the difference is negligible. The saccade offset and PSO onset of gazeNet is the same as for coder RA, while coder MN tags longer saccades and longer PSOs. Given that all these difference remain with 2 ms on average, these are only small details. Overall, the timing differences observed in the event detection output of gazeNet are well within the differences seen in a group of 12 human experts in Hooge et al. (2017) who coded fixations. They report differences between human coders in RTO values of up to 20 ms, which is five times larger than the maximum difference of 4 ms between gazeNet and the human coders for any of the events. The relative timing deviation values for fixations are however at the higher end of the values reported by Hooge et al.. The majority of the 12 coders

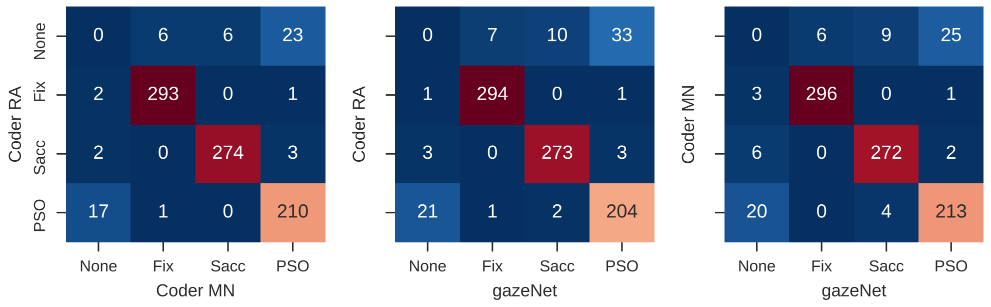

Fig. 9 Event-level confusion matrices

Table 6 Sample-level Cohen's kappa and sample error rate (SER) values for each event class. SER′ = SER*100

	Fixations	Saccades	PSO	All	SER′
Experts					
testSet	0.918	0.925	0.771	0.890	2.83
gazeNet vs:					
trainSet	0.909	0.929	0.757	0.892	2.75
valSet	0.901	0.920	0.725	0.875	3.29
testSet RA	0.900	0.914	0.743	0.873	3.26
testSet MN	0.890	0.904	0.718	0.860	3.54

had RTD around 20ms, which we also find here when comparing the two human coders. gazeNet has a larger RTD of approximately 30 ms, however this value is still within the range of human coders—Hooge et al. report fixation RTD values up to 37 ms.

Comparison to other algorithms and datasets

In Table 8 an evaluation of gazeNet's generalizability to other datasets is provided. The datasets we use for this evaluation are *GazeCom* (Startsev, Agtzidis, & Dorr, 2016; Startsev & Agtzidis, 2017) and *humanFixationEvaluation* (Hessels, Hooge, & Kemner, 2017 containing data from Hessels et al., 2016). We also compare gazeNet's performance to other event detection algorithms that are capable of detecting fixations, saccades and PSOs—the adaptive threshold-based algorithm by Nyström and Holmqvist (2010) (henceforth NH2010) and its recent modification by Friedman, Rigas, Abdulin, & Komogortsev, (2018) (referred to as MNH), and two versions of random forest based algorithm by Zemblys et al. (2018) (referred to as IRF and IRF-spec). In addition, we evaluate how these event detection algorithms perform on our generated dataset (genSet).

Data

The GazeCom (Startsev et al., 2016; Startsev & Agtzidis, 2017) dataset provides over 4.5 h of manually annotated

Table 7 Event-level Cohen's kappa and event error rate (EER) values for each event class. EER′ = EER*100

	Fixations	Saccades	PSO	All	EER′
Experts					
testSet	0.966	0.983	0.844	0.894	5.98
gazeNet vs:					
trainSet	0.972	0.978	0.654	0.840	10.2
valSet set	0.978	0.978	0.790	0.881	7.10
testSet RA	0.966	0.966	0.774	0.861	7.60
testSet MN	0.973	0.970	0.795	0.871	6.85

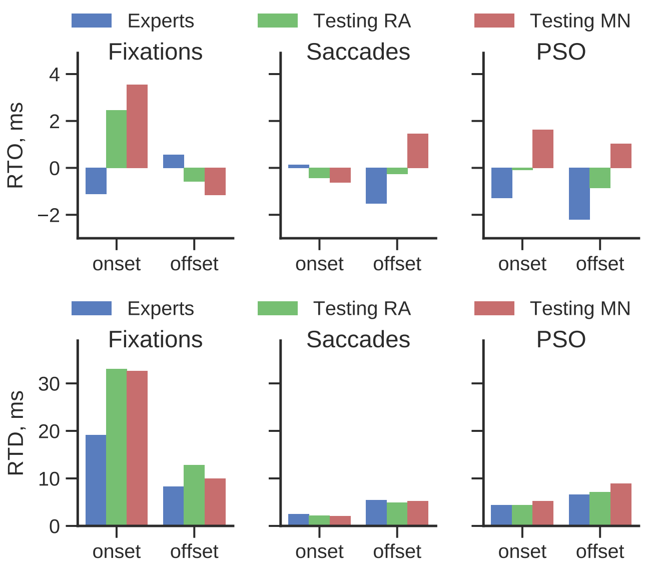

Fig. 10 RTO and RTD analysis. *Blue bars* compare the two human coders. *Green* and *red bars* compare gazeNet's events to the events coded by RA and MN, respectively

eye-tracking data that was recorded with an SR Research EyeLink II at 250Hz. Manual annotations (fixations, saccades, smooth pursuit, unknown and noise) were obtained from 2 coders prior to pre-labeling data by a set of algorithmic approaches and tie-breaking the disagreements (we only use the final labels in our evaluation). The humanFixationEvaluation dataset (Hessels et al., 2016; Hooge et al., 2017) was collected with a Tobii TX300 (300Hz) eye-tracker and derived from infant and adult participants. The total of almost 6 minutes of data were then hand labeled by 12 expert coders, who were asked to label only fixations. Because of the relatively small size of the dataset, we used the labels from all 12 coders in our evaluation.

Algorithms

Data from both datasets were resampled to 500Hz using first-order spline interpolation. Further, for each of the evaluated algorithms, all samples that were not coded as fixations, saccades or PSOs were set to values that the algorithm considers as a missing sample. In the case of humanFixationEvaluation dataset, the samples that were left uncoded were artificially set to saccade samples and only originally missing samples were considered as missing when evaluating the algorithms. Similarly, the output of each of the algorithms for all of the datasets was modified so any other event than fixation, saccade or PSO is labeled as *undefined*. Except for changing the sampling frequency parameter for the MNH algorithm, we did not modify any other parameters for NH2010 and MNH algorithms. The IRF algorithm we use in this study is an updated version of the original IRF described in Zemblys et al. (2018). For this updated version, we retrained IRF using hand labeled data

Table 8 Event and sample level Cohen's kappa for three machine learning algorithms and two traditional algorithms using hand-crafted features and thresholds, for three different eye-movement datasets and our genSet. The highest-scoring algorithm in each cell is printed in *bold*, while the runner-up is *underlined*

Dataset	Algorithm	Event level			Sample level		
		Fixations	Saccades	PSO	Fixations	Saccades	PSO
lund2013-image-test	gazeNet	**0.959**	**0.947**	**0.776**	**0.923**	**0.904**	**0.727**
	IRF	0.780	0.848	0.616	0.760	0.796	0.533
	IRF-spec	0.783	0.844	0.693	0.770	0.801	0.601
	MNH	0.837	0.759	0.598	0.679	0.675	0.540
	NH2010	0.639	0.798	0.350	0.498	0.655	0.310
GazeCom	gazeNet	0.915	**0.845**	–	**0.932**	**0.782**	–
	IRF	0.844	0.779	–	0.811	0.704	–
	IRF-spec	0.843	0.774	–	0.811	0.706	–
	MNH	**0.921**	0.771	–	0.813	0.724	–
	NH2010	0.647	0.745	–	0.480	0.650	–
humanFixationClassification	gazeNet	0.700	–	–	**0.792**	–	–
	IRF	**0.707**	–	–	0.684	–	–
	IRF-spec	0.701	–	–	0.670	–	–
	MNH	0.389	–	–	0.292	–	–
	NH2010	0.477	–	–	0.347	–	–
genSet	gazeNet	**0.918**	**0.884**	**0.719**	**0.851**	**0.902**	**0.766**
	IRF	0.719	0.702	0.436	0.446	0.774	0.484
	IRF-spec	0.720	0.701	0.465	0.453	0.778	0.529
	MNH	0.792	0.606	0.340	0.509	0.673	0.396
	NH2010	0.326	0.543	0.087	0.112	0.564	0.166

very similar to those originally used, covering the same wide range of sampling frequencies (30Hz–1250Hz) and additive noise levels as in Zemblys et al. (2018). Furthermore we have updated the post-processing procedure, as it has been reported[9] that in certain scenarios, the original post-processing code may behave differently than stated in the paper. Just like in Zemblys et al. (2018), we have also trained a specialist classifier (IRF-spec) using only high quality 500–1000Hz data with an additive noise level up to 0.04 degrees RMS.

Results

In Table 8 we provide the average event and sample level Cohen's kappa values for each of the datasets and algorithms, separately for fixations, saccades and PSOs. gazeNet outperforms all other algorithms in all tested datasets for all compared metrics, except for the event-level score for fixations in the GazeCom dataset where MNH scores marginally higher (0.921 compared to gazeNet's 0.915) and in the humanFixationClasification dataset where IRF scores marginally higher (0.707 compared to gazeNet's 0.700).

The other two machine learning algorithms—IRF and IRF-spec—perform quite well and in almost all the cases outperform both threshold-based algorithms. The main exception is in the performance of MNH in detecting fixations: at the event-level it outperforms IRF (but not gazeNet) in the lund2013-image-test, while in the GazeCom and genSet, MNH performs better than IRF at both the event and sample levels. MNH also seems to be on par with IRF at detecting saccades in the GazeCom dataset at both the event and sample levels. The good overall performance of MNH in the GazeCom dataset is likely due to the fact that MNH was fine-tuned for Eyelink1000 data, which is very similar to the GazeCom dataset. However, outside of its familiar territory, MNH, and also NH2010, are outperformed by the machine learning algorithms by quite a large margin. In particular this is evident for the humanFixationEvaluation dataset, where both the MNH and the NH2010 algorithms perform approximately twice as bad than IRF and gazeNet. Hooge et al. (2017) report that the average noise level in this dataset is 0.32 - 0.36 degrees RMS, which means it has a 10x

[9]See https://digital.library.txstate.edu/handle/10877/6874

larger noise than in the lund2013-image-test (see Table 4). Neither MNH nor NH2010 are able to cope with such noise, while both machine learning approaches seem to perform reasonably well (probably because they have seen data with such noise levels during training). Another area where the machine learning approaches are seen to excel is in PSO detection. Both gazeNet and IRF score better at detecting PSO events than the threshold-based MNH and NH2010.

We have also evaluated how the five algorithms perform on our synthetic dataset (genSet) that we used to train gazeNet. Compared to the lund2013-image-test scores, results for genSet show that overall, all five algorithms perform worse, which suggests that genSet differs from real human eye-movement data. However when compared to the results in other datasets, the obtained scores are still within a reasonable range. The final performance of the gazeNet algorithm on lund2013-image-test dataset is evidence that training on the imperfect genSet still leads to excellent detection performance for the final algorithm.

Discussion and conclusions

The main purpose of this work was to build the first fully end-to-end event detector for eye-movement classification. Our network takes raw differentiated eye-tracking data (sample-to-sample offset) as input and classifies it into fixations, saccades and post-saccadic oscillations without any post-processing. It consists of 2 convolutional layers followed by 3 bi-directional recurrent layers and a fully connected layer on top. The convolutional layers are meant to extract deep features from the raw input data, while the recurrent layers model event sequences and are responsible for detecting onsets and offsets of fixations, saccades and PSOs. Finally, the fully connected layer outputs probabilities of each sample being a fixation, saccade or PSO. Because we use simple numerical differentiation without any filtering, in contrast to Hoppe and Bulling (2016), our data are still raw and therefore our approach is completely end-to-end, i.e., no hand-crafted features with settings such as the number of FFT components were required.

The job given to gazeNet was to replicate the hand-coding produced by a human expert, and it came very close. The output of the resulting event detector is on par with the event classification of the two human expert coders at the sample and event levels, as measured by Cohen's kappa, the SER and EER measures (based on Levenshtein distances), and the RTO and RTD analyses proposed by Hooge et al. (2017). We know from Andersson et al. (2017) and Hooge et al. (2017) that hand-coded data represent neither the gold standard, nor the objective truth on what fixations and saccades are. Agreement between coders is nowhere close to perfect, most likely because expert coders often have different conceptions of what constitutes a fixation, a saccade or another event in data. For instance, Hooge et al. (2017) show that differences in onset and offset of fixations can be up to 20 ms for different coders. Importantly, in all discernible aspects, gazeNet is well within the variation of classification performance of the 12 coders in Hooge et al. (2017), and even similar to the variation in coding between the two very similar coders RA and MN, who were both medians in the data from Hooge et al. (2017).

A major challenge with deep learning and machine learning in general is the unbalanced occurrence of the sample classes. Figure 8 illustrates the accuracy paradox: confusion matrices show that expert coders, as well as gazeNet are best at labeling fixation samples. However, this is because the majority of samples belong to fixations, and therefore, a given number of misclassified fixation samples have a lower effect on the final score, compared to the same number of misclassified saccade or PSO samples. This needs to be taken into account when training similar event detectors. To alleviate the problem of the unbalanced input, we first made sure that our generative network sees more saccade and PSO samples than there originally were in the training data. And second, we used a weighted cross-entropy loss function when training gazeNet, and evaluated the detector's performance with measures that do not suffer from the accuracy paradox— sample and event level Cohen's kappa.

Our algorithm importantly shows that hand-crafting features is not necessary for event detection. It is fully possible to build an event detector with excellent performance without introducing any intricate calculations or overt thresholding on the original signal, as was the established practice from Boyce (1967) until the present day. In contrast, the only thing required for the featureless event detection paradigm we introduce here is labeled eye-tracking data. As such, when trained properly using a wide range of different data, a machine learning-based event detector can generalize well to data of different sampling frequencies and noise levels (Zemblys et al., 2018), still without the need to pre- or post-process data or manually set thresholds. The only bottleneck in achieving this is the availability of training data, everything else required is available: various machine learning algorithms and deep network architectures each best suited to specific tasks, a powerful hardware and software ecosystem (e.g., nVidia's CUDA, Google's TensorFlow, PyTorch, etc.) and a fast-growing machine learning community (Grace, Salvatier, Dafoe, Zhang, & Evans, 2017).

Producing this new event detector would have been almost prohibitively costly (in time) if we would have had to hand-code all of the 5h26min data that were used as input into the training phase. Needing only 44s of hand-coded data made it practical. We showed that a recurrent neural network can be trained to generate new labeled eye-movement data with very similar properties as the original data that it learned from: Fixation and saccade properties of the synthetic data are largely the same as in the hand-coded data we started with, and even the nature of the noise in the signal—the $\frac{\text{RMS}}{\text{STD}}$ value (Holmqvist et al., 2017)—is almost identical to that of the original data. Having the possibility to replicate hand-coded data and radically magnify the training set will open up the possibility to train many more event detection algorithms using this paradigm.

Interestingly, despite being trained on coder RA's data, at the event-level gazeNet agrees somewhat more with coder MN, especially in PSO detection (see Table 7). Figure 9 also shows how gazeNet agrees with human coders to the same level as expert coders between each other for fixations and saccades, but not PSO classification. The two expert coders find 18 and 27 PSOs that are not present in other coder's data (false positives), while gazeNet identifies 37 and 28 false-positive PSOs when compared to expert coders. This suggests that even human experts show considerable disagreement when coding PSOs and underscores the point that more work is needed to enable improved PSO detection.

Any coder uses his/her knowledge about eye-tracking data (supplemented by whatever background this researcher has) to label the fixations, saccades and other events in the eye-tracking data. Their decision processes are likely to be governed by some set of decision rules, possibly including thresholds, that the coders apply. However, we cannot know what goes on in the minds of coders, and even if they would introspectively ask themselves what they are doing, and come up with the answer that they employ certain thresholds, why should we trust that introspection? Even if internal thresholding rules are applied, how do we know whether they apply them consistently? We can only measure how similar each coder is to the output of the algorithm. However, the machine-learning algorithm has the advantage that during learning it sees a lot of instances of an event and thereby likely generates a more average and generalized representation of an event by removing some of the idiosyncratic event-to-event variation found in the human coding, but possibly also losing some of its situation-adaptiveness. By directly learning the appearance of an event according to the labeled training data, machine learning algorithms are able to agree more closely with human coders than conventional algorithms as shown in Table 8. The results in this table further suggest that although it is possible to tweak a conventional algorithm to work well for certain data and event types, when it comes to difficult cases such as PSO detection, machine learning based algorithms markedly outperform the conventional algorithms and are capable of better capturing the nature of the event as specified by the human coders in the training data.

The results presented in this paper indeed support the intriguing suggestion that during the training, gazeGenNet and gazeNet developed a generalized representation of the eye-movement events present in the training data. First, the main sequence relationship for the saccades in the data generated by gazeGenNet is closer to that of the validation and testing sets, than to that of the training data (see Fig. 4). This suggest that our training set (coded only by coder RA) is somehow different from the validation and testing sets (coded by both RA and MN) but that these differences are overcome by gazeGenNet. Second, for gazeNet, it was surprising to see that its coding output at the event level agreed more with human coder MN than with human coder RA on whose coding the algorithm was trained (see Table 7). This suggests that gazeNet learned something about the data that was not contained in RA's codings, but was present in MN's. Moreover, the evaluation of gazeNet on two other datasets, GazeCom and humanFixationClasification, showed that gazeNet is not only able to generalize to unseen data with other qualities, but also agreed more with the hand-coding provided in these dataset than the other event detection algorithms.

If there is some truth in the hypothesis that machine learning algorithms are able to learn generalized event representations that perhaps emphasize some basic underlying properties of the data, then it opens the question how noisy the hand-labeled training data can be, in order to still be able to learn an event detector from it that is capable of producing a good event classification. It is quite possible that we do not need experts to code the data, but instead any person with a few minutes of training can code eye-movement data to a sufficient standard that it can be used for training an event detector through deep learning or other machine learning techniques. If so, we can make use of crowd-sourcing platforms to get huge amounts of possibly noisily coded data, but good enough to still train a well performing event detector. Another option could then be to use unsupervised approaches (e.g., Houpt, Frame, & Blaha, 2017), or in fact any other, *good enough* event detection algorithm, or an ensemble of algorithms, to produce pseudo-labels for training data. In either case, the classifier trained on such data would need to be evaluated by human experts to judge how well such classifier is performing.

Future work

Besides fixations, saccades and PSOs, there are plenty of other events in eye-movement data waiting to be detected algorithmically. Nystagmus and smooth pursuit are classic problems for feature-based algorithms, with only a few specifically tailored algorithms existing that are able to deal with these events (see for example Komogortsev &

Karpov, 2013; Larsson et al., 2013, 2015) for pursuit and Juhola (1988), Turuwhenua, Yu, Mazharullah, and Thompson (2014) and Sangi, Thompson, and Turuwhenua, (2015) for nystagmus detection). A major challenge to the field is to produce an algorithm that codes events from head-mounted eye trackers, or eye trackers integrated in head-mounted display used for virtual reality, given that the data recorded with such systems contains a rich mixture of head and eye movements. Based on the recurring finding that PSOs are already quite a difficult event to classify consistently by a feature-based algorithms despite only involving an eye-movement data stream, we think that especially these situations involving further data streams (e.g., accelerometer, eye video, etc.) become too complex to write a hand-crafted algorithm for. We thus expect that the machine learning approach we present here is the method to pursue for producing an event detector that can deal with all these events at once.

A second important direction for future work is to further develop techniques suitable for generating labeled eye-tracking data. The current work was only possible due to gazeGenNet extending the length of the training set by 450 times, thereby solving the problem that deep learning-based event detectors need a large amount of training data. As this problem is expected to get only larger as more complex event detectors involving more event classes and data streams are developed, we expect that improvements of gazeGenNet will prove to be an important area of research. However, even when generating the training set, a common challenge for both traditional and machine learning-based methods that remains, is obtaining reliably coded eye-tracking data, on which the data generator can be trained and on which created algorithms can be tested. Given that we see that even expert coders have trouble coding PSOs in raw data, it would be not unlikely that coders will have even more trouble labeling fixations or smooth pursuit mixed with head movements, vergence, OKN, VOR and other complex eye-movements.

Conclusions

In summary, we have shown that an event detector constructed through deep learning based on only a few short segments of hand-coded data and working autonomously without the need of predefined feature extraction or post-processing nor the need to set any thresholds can perform as well as human expert coders in coding fixations, saccades and PSOs. This shows that constructing event detectors through machine learning is a promising approach for the future, and can hopefully enable the creation of event detectors that can deal with more types of eye movements simultaneously than a hand-crafted algorithm ever could. There is however a trade-off made when choosing the machine learning approach over the traditional approach of hand-crafting an event detector. Hand crafting provides the researcher with insight into how the resulting events are defined in terms of signal properties of the underlying eye-movement data. This insight is lost with the machine learning approach, which produces an event detector that is essentially a black box. Yet, the aim of performing event detection for the vast majority of researchers using eye tracking is to get properly labeled fixations, saccades and other events that they can then use as the basis of their analyses. Therefore, outside of the small crowd of researchers that may be somewhat unsatisfied with the machine learning approach because of their interest in event detection techniques and the signal properties of eye-tracking data, for the vast majority of researchers the machine learning approach is a valuable and, as we show here, a very well performing method of event detection.

How to use this algorithm

This paper primarily presents the algorithm as a proof of concept, and is not to be understood as an off-the-shelf algorithm that can be employed instantly with no preparation and no understanding of how it works. Readers who prefer to control thresholds and explore their effect on event detection should use another algorithm or implement a post-processing procedure that, for example, removes saccades with an amplitude below a certain threshold or fixations below a certain duration. That said, we do provide code for gazeNet (`https://github.com/r-zemblys/gazeNet`) and gazeGenNet (`https://github.com/r-zemblys/gazeGenNet`), along with trained models, and the reader is free to use it. We recommend carefully evaluating whether the detector provides satisfactory results on the reader's particular dataset. Be advised that this particular classifier only labels eye-tracking data as fixations, saccades and PSOs, therefore if the data contain other events, like smooth-pursuit, blinks, etc., these need to be removed first, or preferably, we encourage users to train their own classifiers on their own data following the framework set out in this paper. Our responsibility is to build the method and show that it is a viable avenue for developing eye-movement event detectors, and the responsibility of the user is to make sure that their use of our framework, code or classifier is appropriate for their data.

Acknowledgements We thank Richard Andersson and Marcus Nyström for making their manually labeled eye-tracking data publicly available. We also express our gratitude to two anonymous reviewers for providing useful comments and suggestions on earlier versions of this manuscript.

Appendix A: Training, validation and testing data sets

Table 9 List of files in Lund2013-image dataset and their assignment to the training, validation and testing sets

trainSet
TH38_img_Europe_labelled_RA
TH46_img_Rome_labelled_RA
TH50_img_vy_labelled_RA
TL44_img_konijntjes_labelled_RA
TL48_img_Europe_labelled_RA
TL48_img_Rome_labelled_RA
valSet RA
UH27_img_vy_labelled_RA
UH29_img_Europe_labelled_RA
UH47_img_Europe_labelled_RA
valSet MN
UH27_img_vy_labelled_MN
UH29_img_Europe_labelled_MN
UH47_img_Europe_labelled_MN
testSet RA
TL28_img_konijntjes_labelled_RA
TH34_img_Europe_labelled_RA
TH34_img_vy_labelled_RA
TL20_img_konijntjes_labelled_RA
UH21_img_Rome_labelled_RA
UH33_img_vy_labelled_RA
UL23_img_Europe_labelled_RA
UL31_img_konijntjes_labelled_RA
UL39_img_konijntjes_labelled_RA
UL43_img_Rome_labelled_RA
UL47_img_konijntjes_labelled_RA
testSet MN
TL28_img_konijntjes_labelled_MN
TH34_img_Europe_labelled_MN
TH34_img_vy_labelled_MN
TL20_img_konijntjes_labelled_MN
UH21_img_Rome_labelled_MN
UH33_img_vy_labelled_MN
UL23_img_Europe_labelled_MN
UL31_img_konijntjes_labelled_MN
UL39_img_konijntjes_labelled_MN
UL43_img_Rome_labelled_MN
UL47_img_konijntjes_labelled_MN

Appendix B: Labeling mistake correction

We found an obvious labeling mistake in the one of the validation trials (file *UH29_img_Europe_labelled_MN*. We fixed this error by reassigning 75 samples, [3197,3272) (zero-based index), from the saccade to the fixation class.

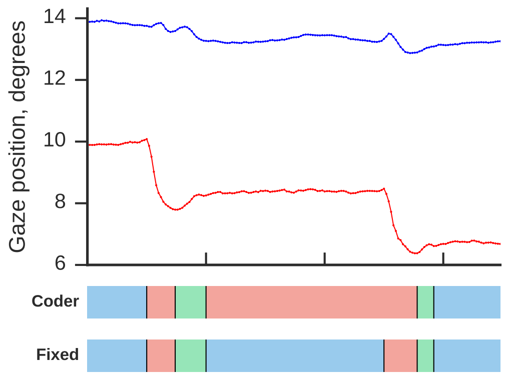

Fig. 11 Data fix in *UH29_img_Europe_labelled_MN* file. *Top scarf plot* - original classification by the expert, *bottom scarf plot* - fixed classification. *Blue*, *red*, and *green patches* are fixations, saccades, and PSOs, *blue* and *red lines* are horizontal and vertical gaze data, respectively. *Ticks* on the horizontal axis denote 100-ms intervals

Appendix C: Unclear PSO case

Figure 12 shows an example of an unclear PSO case. The expert classified the downward movement as a PSO, while gazeNet does not tag a PSO here and classifies all high velocity data as a saccade.

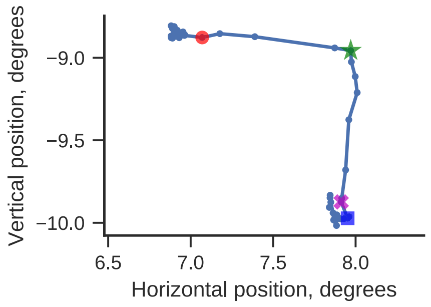

Fig. 12 Example of an unclear PSO case. The *red circle* denotes the saccade onset, as tagged both by expert RA and gazeNet. The *green star* and magenta x identify the saccade and PSO offsets according to the expert coder. The *blue square* shows saccade offset according to gazeNet

Appendix D: Synthetic data examples

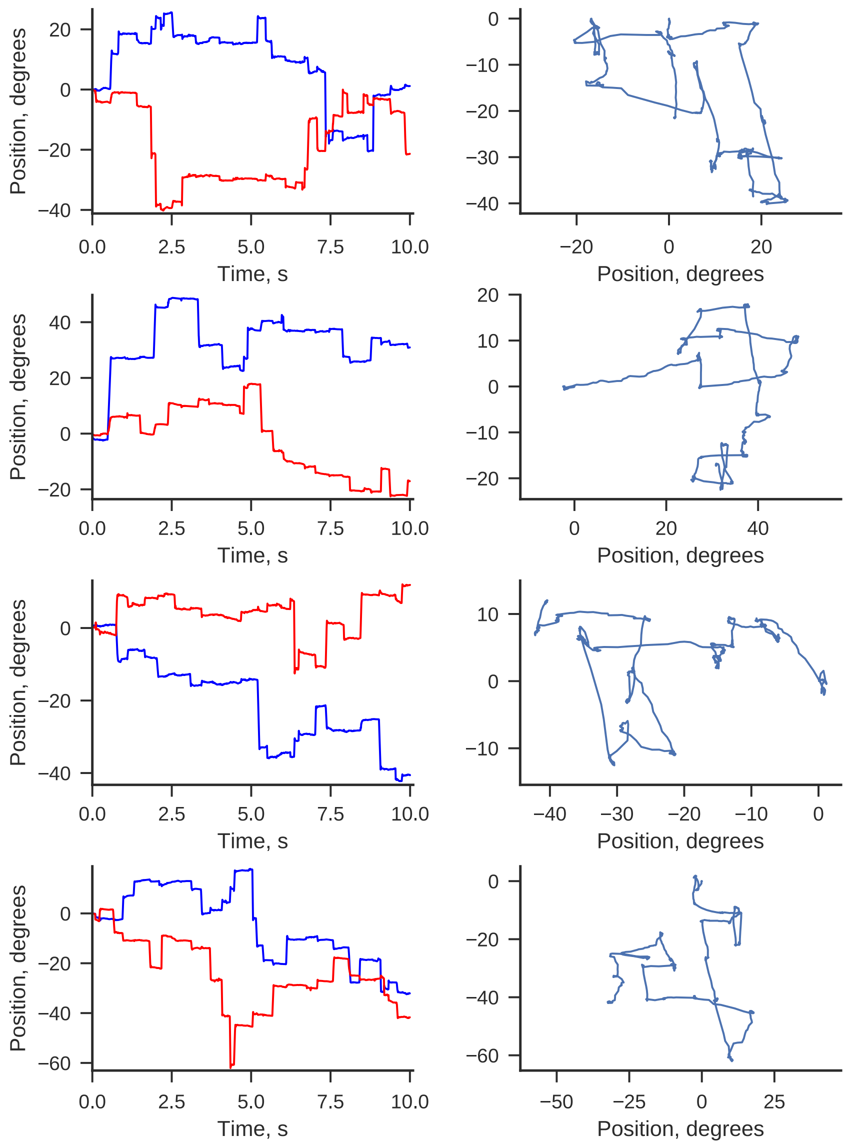

Fig. 13 Randomly selected examples of synthetic eye-tracking data. Horizontal (*blue*) and vertical (*red*) trials over time on the left, and a scanpath on the right

Appendix E: Examples of PSO codings

Figures 14, 15, 16 and 17 show position over time and 2D scanpath plots of example cases where gazeNet does not agree with expert coders in PSO classification. In each of the scarf plots, blue, red and green patches are fixations, saccades and PSOs. Blue and red lines in the position over time plots are horizontal and vertical gaze data, respectively. Ticks on the horizontal time axis denote 20-ms intervals. The red circles in the scanpath plots indicate the beginning of the example trials (plotted in gray), while thick colored lines connect onsets and offsets of PSOs as labeled by gazeNet and two expert coders. These examples are meant to illustrate how complex the eye-tracking data are and to allow the reader to form their own opinion of whether they agree more with how the human coders or how gazeNet have tagged the PSOs. The mean was removed from both channels of each segment for illustration purposes.

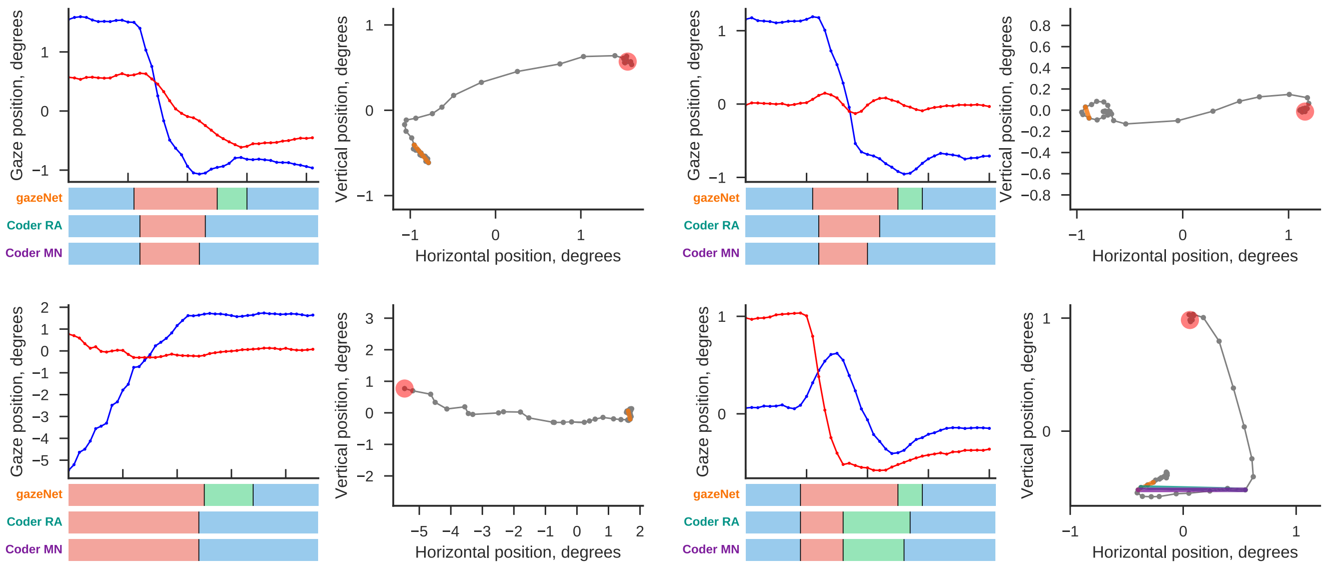

Fig. 14 Examples of gazeNet detecting false-positive PSOs as compared to both of the coders. Lower right example shows a case where coders' PSO is matched with gazeNet's saccade by our evaluation algorithm because of higher overlap

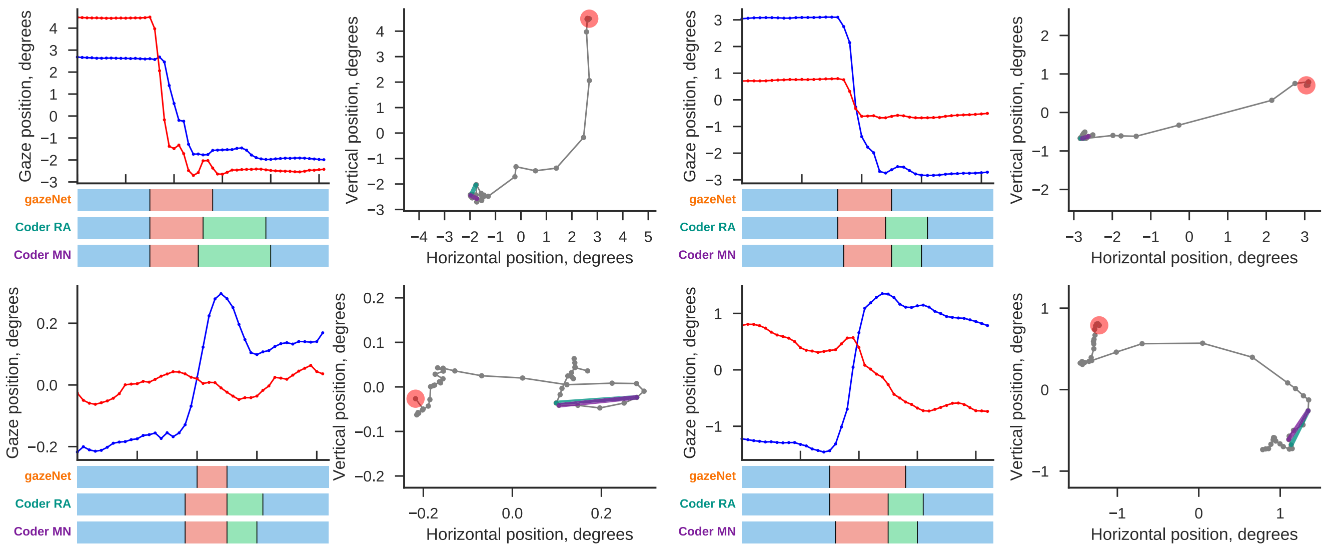

Fig. 15 Examples of gazeNet not detecting PSOs whereas both coders think that there is a PSO present in the data

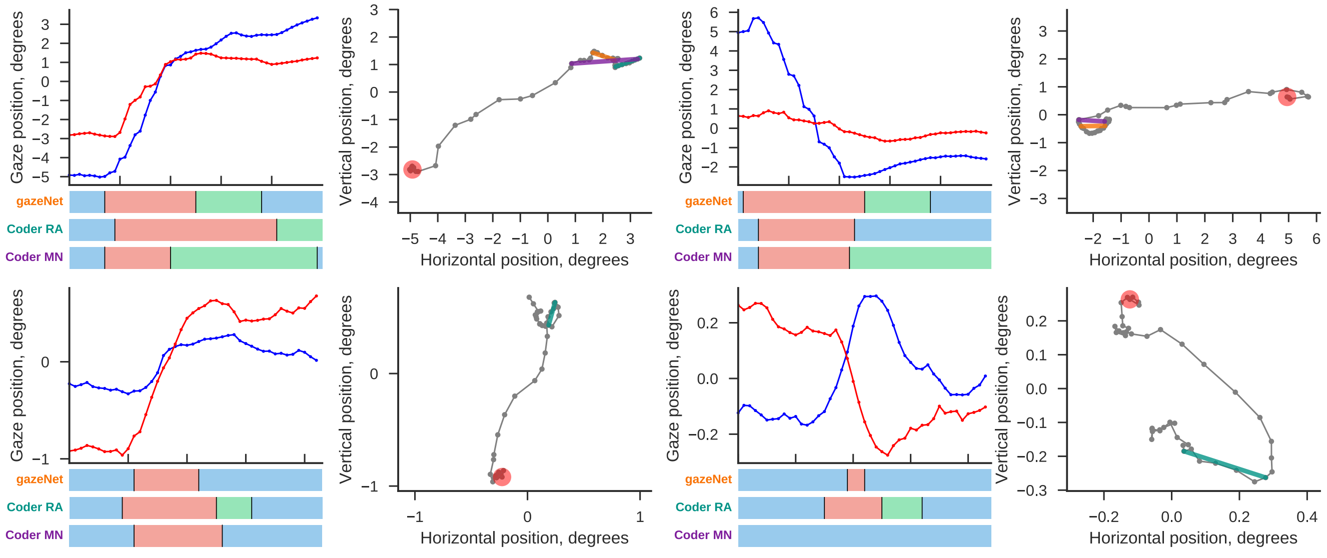

Fig. 16 Examples of gazeNet not agreeing with coder RA but agreeing with coder MN (except for the bottom right example)

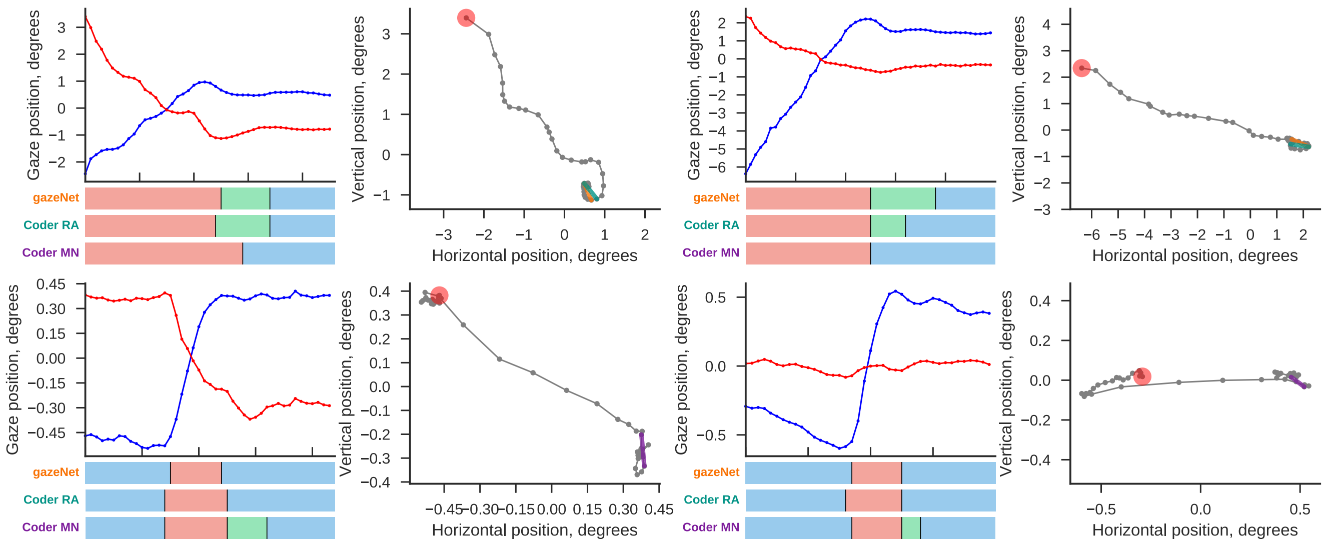

Fig. 17 Examples of gazeNet not agreeing with coder MN but agreeing with coder RA

References

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., …, Zhu, Z. (2015). Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. ArXiv e-prints.

Anantrasirichai, N., Gilchrist, I. D., & Bull, D. R. (2016). Fixation identification for low-sample-rate mobile eye trackers. In *2016 IEEE international conference on image processing (ICIP)* (pp. 3126–3130). IEEE.

Andersson, R., Larsson, L., Holmqvist, K., Stridh, M., & Nyström, M. (2017). One algorithm to rule them all? An evaluation and discussion of ten eye movement event-detection algorithms. *Behavior Research Methods*, *49*(2), 616–637.

Bahill, A. T., Brockenbrough, A., & Troost, B. T. (1981). Variability and development of a normative data base for saccadic eye movements. *Investigative Ophthalmology & Visual Science*, *21*(1), 116–125.

Bishop, C. M. (1994). Mixture density networks.

Blignaut, P., & Beelders, T. (2012). The precision of eye-trackers: A case for a new measure. In *Proceedings of the symposium on eye tracking research and applications, ETRA '12* (pp. 289–292). New York: ACM.

Boyce, P. R. (1967). Monocular fixation in human eye movement. *Proceedings of the Royal Society of London B: Biological Sciences*, *167*(1008), 293–315.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using

RNN encoder-decoder for statistical machine translation. CoRR, arXiv:1406.1078

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37–46.

Duchowski, A. T., Jörg, S., Allen, T. N., Giannopoulos, I., & Krejtz, K. (2016). Eye movement synthesis. In *Proceedings of the ninth biennial ACM symposium on eye tracking research & applications* (pp. 147–154). ACM.

Enderle, J. D., & Zhou, W. (2010). Models of horizontal eye movements, part ii: A 3rd-order linear saccade model. *Synthesis Lectures on Biomedical Engineering*, *5*(1), 1–159.

Engbert, R., & Kliegl, R. (2003). Microsaccades uncover the orientation of covert attention. *Vision Research*, *43*(9), 1035–1045.

Friedman, L., Rigas, I., Abdulin, E., & Komogortsev, O. V. (2018). A novel evaluation of two related and two independent algorithms for eye movement classification during reading. *Behavior Research Methods*, 1–24.

Grace, K., Salvatier, J., Dafoe, A., Zhang, B., & Evans, O. (2017). When will AI exceed human performance? Evidence from AI experts. CoRR, arXiv:1705.08807

Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv:1308.0850

Hein, O., & Zangemeister, W. (2017). Topology for gaze analyses - raw data segmentation. *Journal of Eye Movement Research*, *10*(1), 1–25.

Hessels, R. S., Hooge, I. T., & Kemner, C. (2016). An in-depth look at saccadic search in infancy. *Journal of Vision*, *16*(8), 10–10.

Hessels, R. S., Niehorster, D. C., Kemner, C., & Hooge, I. T. C. (2017). Noise-robust fixation detection in eye movement data: Identification by two-means clustering (i2mc). *Behavior Research Methods*, *49*(5), 1802–1823.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Holmqvist, K., & Andersson, R. (2017). *Eye tracking. A comprehensive guide to methods, paradigms and measures.* Lund Eye-Tracking Research Institute.

Holmqvist, K., Zemblys, R., & Beelders, T. (2017). Magnitude and nature of variability in eye-tracking data. In *Proceedings of the ECEM* (p. 2017). Wuppertal: ECEM.

Hooge, I., Holmqvist, K., & Nyström, M. (2016). The pupil is faster than the corneal reflection (CR): Are video based pupil-CR eye trackers suitable for studying detailed dynamics of eye movements? *Vision Research*, *128*, 6–18.

Hooge, I. T. C., Niehorster, D. C., Nyström, M., Andersson, R., & Hessels, R. S. (2017). Is human classification by experienced untrained observers a gold standard in fixation detection? *Behavior Research Methods.*

Hoppe, S., & Bulling, A. (2016). End-to-end eye movement detection using convolutional neural networks. ArXiv e-prints.

Houpt, J. W., Frame, M. E., & Blaha, L. M. (2017). Unsupervised parsing of gaze data with a beta-process vector auto-regressive hidden markov model. *Behavior Research Methods.*

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR, arXiv:1502.03167

Juhola, M. (1988). Detection of nystagmus eye movements using a recursive digital filter. *IEEE Transactions on Biomedical Engineering*, *35*(5), 389–395.

Komogortsev, O. V., Gobert, D. V., Jayarathna, S., Koh, D. H., & Gowda, S. M. (2010). Standardization of automated analyses of oculomotor fixation and Saccadic behaviors. *IEEE Transactions on Biomedical Engineering*, *57*(11), 2635–2645.

Komogortsev, O. V., & Karpov, A. (2013). Automated classification and scoring of smooth pursuit eye movements in the presence of fixations and saccades. *Behavior Research Methods*, *45*(1), 203–215.

Larsson, L., Nyström, M., Andersson, R., & Stridh, M. (2015). Detection of fixations and smooth pursuit movements in high-speed eye-tracking data. *Biomedical Signal Processing and Control*, *18*, 145–152.

Larsson, L., Nyström, M., & Stridh, M. (2013). Detection of saccades and postsaccadic oscillations in the presence of smooth pursuit. *IEEE Transactions on Biomedical Engineering*, *60*(9), 2484–2493.

Lee, S. P., Badler, J. B., & Badler, N. I. (2002). Eyes alive. In *ACM transactions on graphics (TOG)* (Vol. 21, pp. 637–644). ACM.

Ma, X., & Deng, Z. (2009). Natural eye motion synthesis by modeling gaze-head coupling. In *Virtual reality conference, 2009. VR 2009. IEEE* (pp. 143–150). IEEE.

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., ..., Bengio, Y. (2016). Samplernn: An unconditional end-to-end neural audio generation model. cite arXiv:1612.07837

Mould, M. S., Foster, D. H., Amano, K., & Oakley, J. P. (2012). A simple nonparametric method for classifying eye fixations. *Vision Research*, *57*, 18–25.

Nyström, M., & Holmqvist, K. (2010). An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavior Research Methods*, *42*(1), 188–204.

Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 symposium on eye tracking research & applications, ETRA '00* (pp. 71–78).

Sangi, M., Thompson, B., & Turuwhenua, J. (2015). An optokinetic nystagmus detection method for use with young children. *IEEE Journal of Translational Engineering in Health and Medicine*, *3*, 1–10.

Startsev, M., Agtzidis, I., & Dorr, M. (2016). Smooth pursuit. http://michaeldorr.de/smoothpursuit/

Startsev, M., & Agtzidis, I. (2017). Manual & automatic detection of smooth pursuit in dynamic natural scenes. In *Proceedings of the European conference of eye movements.*

Sutskever, I., Martens, J., & Hinton, G. (2011). Generating text with recurrent neural networks. In Getoor, L., & Scheffer, T. (Eds.) *Proceedings of the 28th international conference on machine learning (ICML-11), ICML '11* (pp. 1017–1024). New York: ACM.

Tinker, M. A. (1928). Eye movement duration, pause duration, and reading time. *Psychological Review*, *35*(5), 385.

Turuwhenua, J., Yu, T.-Y., Mazharullah, Z., & Thompson, B. (2014). A method for detecting optokinetic nystagmus based on the optic flow of the limbus. *Vision Research*, *103*, 75–82.

Van Den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *Proceedings of the 33rd international conference on international conference on machine learning - Volume 48, ICML'16* (pp. 1747–1756). JMLR.org.

Yeo, S. H., Lesmana, M., Neog, D. R., & Pai, D. K. (2012). Eyecatch: Simulating visuomotor coordination for object interception. *ACM Transactions on Graphics (TOG)*, *31*(4), 42.

Zemblys, R. (2016). Eye-movement event detection meets machine learning. In *Biomedical engineering* (pp. 98–101).

Zemblys, R., Niehorster, D. C., Komogortsev, O., & Holmqvist, K. (2018). Using machine learning to detect events in eye-tracking data. *Behavior Research Methods*, *50*(1), 160–181.