Behavior Research Methods (2023) 55:1653–1714
https://doi.org/10.3758/s13428-021-01763-7

Evaluating Eye Movement Event Detection: A Review of the State of the Art

Mikhail Startsev[1] · **Raimondas Zemblys**[2]

Accepted: 27 November 2021 / Published online: 17 June 2022

Abstract

Detecting eye movements in raw eye tracking data is a well-established research area by itself, as well as a common pre-processing step before any subsequent analysis. As in any field, however, progress and successful collaboration can only be achieved provided a shared understanding of the pursued goal. This is often formalised via defining metrics that express the quality of an approach to solving the posed problem. Both the big-picture intuition behind the evaluation strategies and seemingly small implementation details influence the resulting measures, making even studies with outwardly similar procedures essentially incomparable, impeding a common understanding. In this review, we systematically describe and analyse evaluation methods and measures employed in the eye movement event detection field to date. While recently developed evaluation strategies tend to quantify the detector's mistakes at the level of whole eye movement events rather than individual gaze samples, they typically do not separate establishing correspondences between true and predicted events from the quantification of the discovered errors. In our analysis we separate these two steps where possible, enabling their almost arbitrary combinations in an evaluation pipeline. We also present the first large-scale empirical analysis of event matching strategies in the literature, examining these various combinations both in practice and theoretically. We examine the particular benefits and downsides of the evaluation methods, providing recommendations towards more intuitive and informative assessment. We implemented the evaluation strategies on which this work focuses in a single publicly available library: https://github.com/r-zemblys/EM-event-detection-evaluation.

Keywords Eye movement event detection · Evaluation · Metrics · Event matching

Introduction

Most of the eye movement research to date relies heavily on eye movement event detection – parsing the raw gaze data into various eye movement types, including fixations, saccades, post-saccadic oscillations (PSOs), smooth pursuits (SPs), optokinetic nystagmus (OKNs), etc[1]. Ever since computers were first employed for eye movement data analysis, researchers as well as eye-tracker manufacturers developed a vast number of event detection algorithms (Nyström & Holmqvist, 2010; Komogortsev & Karpov, 2013; Larsson et al., 2013; Anantrasirichai et al., 2016; Hessels et al., 2017; Houpt et al., 2018; Zemblys et al., 2018; Bellet et al., 2019; Startsev et al., 2019a; Zemblys et al., 2019b; Dar et al., 2020; Kothari et al., 2020, to name a few). These algorithms employ different techniques for detecting events, such as thresholding the velocity, spatial gaze sample distribution, or other hand-crafted features, use various statistical methods or machine learning approaches. Naturally, a question then arises: Which algorithm performs better and which one to use? However, interpreting or comparing the reported performance figures for the algorithms can often prove challenging even for the experts in the field: Even if one disregards the differences between various datasets and only focuses on the strategies for evaluating the algorithms, the diversity is very high.

[1]For definitions of these and other events we refer the readers to dedicated literature, e.g. (Lappi, 2016; Hessels et al., 2018). Here it suffices to say that various event types correspond to characteristic patterns in the eye tracking signal, and these differences can be exploited to automatically detect events of the different types.

✉ Mikhail Startsev
mikhail.startsev@gmail.com

[1] Munich, Germany

[2] Gothenburg, Sweden

Currently, there is neither a standard "go-to" performance metric for eye movement event detectors, nor a standard way of choosing one. When considering how to approach designing or assessing an evaluation pipeline in any particular case, the *purpose* of deriving quantitative measures describing a set of eye movement event labels naturally has a bearing on the applicable evaluation strategies: For example, one would likely use different measures to report on a comparison between a dozen of eye movement detectors (a concise set of measures of the overall performance that could be easily compared e.g. in the form of a table or a plot), and to present an in-detail description of the labelling patterns in two sets of experts' labels (a set of highly descriptive statistics that would facilitate the discussion of the differences in expert annotations). Ideally, of course, a set of computed performance measures would be suitable for all applications: It would be both concise and descriptive, enabling easy comparison of competing algorithms as well as a clear understanding of the differences in the annotations.

Above all and in the context of any use case, however, the evaluation should be *fair*, i.e., insofar as possible, not over- or underestimate the performance of evaluated algorithms. To more clearly specify this concept, we provide two important examples of what fairness entails in this context:

- When comparing several algorithms to one another, it is especially important that the evaluation is not biased in favor of a subset of these, either by-design (e.g. overfitting - for instance when the parameters of the newly developed algorithm are tuned on data extremely similar to the one used for the comparative evaluation, while the other similar algorithms are untuned) or because the evaluation pipeline implicitly "prefers" certain patterns (e.g. completely failing to register certain kinds of labelling errors, for instance event fragmentation). The latter is as important when examining the performance of a single algorithm in isolation, so as not to induce a false sense of very good performance.
- Meaningful results should be produced for any sets of compared entities. This includes, for instance, not producing reassuringly good evaluation results for intentionally unreasonable predictions (e.g. randomly assigned labels (Startsev et al., 2019)). This robustness enables the researchers to trust the results of the evaluation without having to keep in mind the situations where one or the other evaluation method is known to be unreliable.

Evaluation pipeline and paper structure

The motivation outlined above forms the basis of our assessment of all evaluation approaches we review in this paper. With this in mind, we argue that certain choices made already well ahead of actually computing some metric of the algorithm's performance can affect the fairness and descriptiveness of the evaluation as a whole. Therefore, the evaluation pipeline as it is discussed here (see the diagram in Fig. 1) starts already at the stage of selecting or acquiring the eye tracking data to be used (corresponding to "Data source"). The green parallelograms in the figure denote the different choices that the researchers can make in this context. At this stage e.g. selecting a dataset that does not contain annotated eye movement events may either severely limit the scope – and the descriptiveness – of potentially

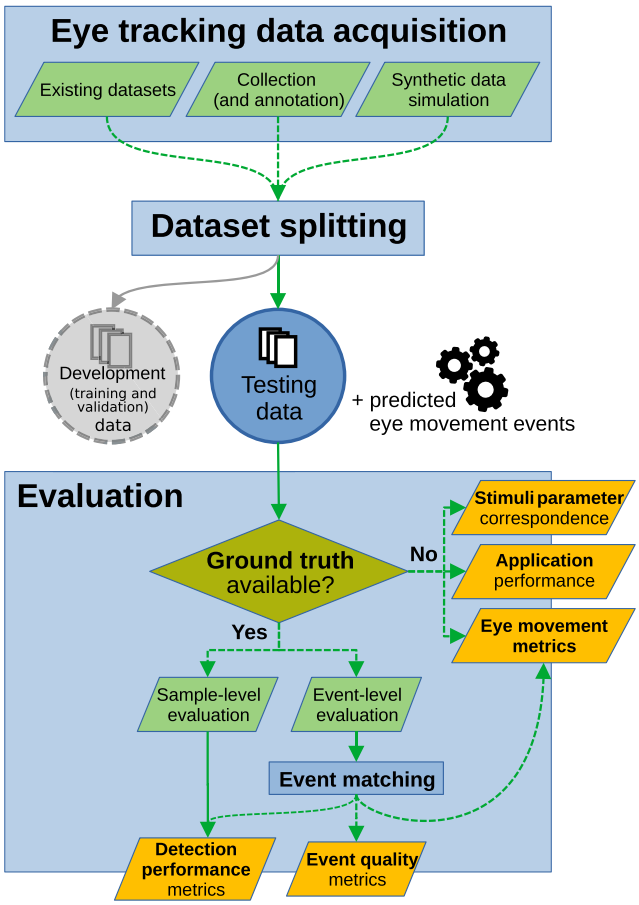

Fig. 1 Diagram of the steps (blue blocks) and respectively available options (green parallelograms) involved in the complete evaluation process, up to the quantitative outcomes of the evaluation (denoted in orange). In order to put these into context, depending on the chosen evaluation procedure (cf. "Evaluation procedures"), they can be compared to the same metrics for either other algorithms or the same algorithm under different conditions

applicable evaluation methods, or result in the necessity of laborious annotation.

Next, the collection of eye tracking data needs to be subdivided into development and testing data. While in the context of evaluation we are only interested in the testing part of the data, a careless splitting of the data may result in invalid or inherently unfair evaluation, e.g. if the development set is not sufficiently different from the testing data. "Validation procedures" deals with different validation procedures that can be employed for algorithm evaluation, with "How to split the data" focusing on the specific aspects of the development-testing split that may affect the fairness of the evaluation results.

Our main focus – and the largest block in Fig. 1 – is the evaluation itself, i.e. the computation of certain metrics or statistics reflecting the performance of the tested eye movement detector. The primary factor affecting the evaluation choices is the presence of *ground truth* for the eye movements in the eye tracking signal – typically expert annotations[2], though when comparing the outputs of two detectors to one another, the predictions of one of them can be used as ground truth.

In this paper we briefly touch on the evaluation possibilities that do not involve labeling the ground truth ("Evaluation based on eye movement metrics", "Evaluation based on stimuli parameters", "Application-based evaluation"), corresponding to the topmost group of possible outputs of the evaluation block in Fig. 1. Methods that are applicable in a particular case heavily depends on the specific set-up of an eye tracking experiment: E.g. whether the eye tracking data can be stored and analyzed offline; to which extent the known properties of the stimuli and instructions given to the participants define the gaze behavior in advance or can be correlated to some statistics of this behavior (e.g. duration of fixations, amplitude of saccades, etc.); whether there is a target application, the performance of which is the primary optimization target when improving or developing the eye movement event detection (e.g. gaze-based user interaction or biometrics); etc.

By far the largest part of the paper is dedicated to various aspects of comparing the predictions of an eye movement detector to the available ground truth (i.e. the "Yes" branch in the evaluation block of the flowchart in Fig. 1). "Detection performance evaluation" provides an overview of the necessary steps and concepts, introducing sample- and event-level evaluation, as well as eye movement event matching. There we also discuss certain decisions that are crucial for the evaluation, such as how multiclass evaluation or unlabelled samples can be handled, etc.

"Evaluation metrics" is dedicated to describing the evaluation metrics, including those that measure how well an the eye movement events are detected (according to a certain criterion of accepting a detection) and those that measure the detections' *quality* (i.e. how well they align with the corresponding "true" events) – the two outputs at the bottom of the evaluation block in the flowchart. At the end of this section we discuss the usability of the reviewed metrics as a stand-alone evaluation tool in terms of a number of properties (naturally handling multiclass evaluation, suitability for imbalanced data sets, etc.), comparing them to one another.

In "Event matching methods" we describe the specific details of various event matching approaches – a cornerstone step for almost any type of event-level evaluation of an eye movement detector. This step is at the same time extremely implementation-dependent (as the logic of most matchers in the literature is non-trivial) and lacks any standardization across different works. The fine implementation details or potential flaws in the matching logic can substantially shift the outcome of the overall evaluation. Moreover, there is not one "correct" event matching logic that would handle all the corner-cases that could come up in the data in a way that would be satisfactory for any use case. Therefore, it is important to understand the specific matching patterns that one can expect from the matchers described in the literature. We provide plentiful illustration of the different matchers' behavior in this section, discussing their benefits and shortcomings, in general and with respect to one another. At the end of this section, we summarize the matcher landscape for an easier overview.

To illustrate and quantify the influence of event matching on the evaluation, in "Interaction between the performance metrics and event matchers" we examine the difference between the values of the same metric obtained under different event matching approaches on an example of a large published dataset. This highlights the difficulty in comparing event-level metrics between papers that use subtly different evaluation methodology.

Finally, "Summary, recommendations, and discussion" summarizes the recommendations towards improving the evaluation pipeline made throughout this work. We also discuss various open questions regarding designing or assessing an evaluation strategy.

Intended audience and related work

This review of the existing practices of evaluating eye movement event detection algorithms mostly mirrors the

[2]There is plenty of debate about whether (or to which extent) expert annotations represent the gold standard of eye movement detection (Hooge et al., 2018), what are the exact definitions of fixations (Hessels et al., 2018), saccades, and other events, what information is needed to annotate the exact onsets and offsets of the events, especially when eye-head coordination is involved (Agtzidis et al., 2019), etc. These questions are outside the scope of this review, but we encourage algorithm developers to always report against what their algorithms' outputs are being evaluated.

tutorial that the authors held at the ACM Symposium on Eye Tracking Research & Applications (Startsev & Zemblys, 2019), expanding on the ideas and methods presented there, covering more ground overall, and providing a comprehensive source of information on the subject. We note that algorithm evaluation strategies that we describe in the rest of the paper are mostly relevant for the algorithm developers who want to convincingly demonstrate the advantages of their proposed methods or gain insights in order to improve them. Likewise, researchers examining and reviewing publications in the eye movement detection field could find the critical review of the existing evaluation methodology useful, as it aligns with the interests of ensuring the high quality of publications that contribute to the state of the art, as well as encouraging inter-comparability and reproducibility in the research field.

A deeper understanding of the building blocks of the evaluation process would also allow those not actively developing their own eye movement detectors to make a better informed decision when e.g. selecting an evaluation procedure to follow in order to choose an existing detector to use in their application. It has to be pointed out, however, that there is no single "right" choice of an evaluation pipeline, as e.g. some groups of those in need of creating an evaluation set-up might prioritize ease of use or implementation simplicity over robustness or descriptiveness, and vice versa.

Studies focusing on the evaluation measures exclusively are rare, as even novel evaluation approaches are typically introduced in conjunction with describing a new algorithm and, therefore, not extensively studied in a dedicated manner. The authors usually name the key differences from the other methods in the literature, as well as the addressed shortcomings of the existing evaluation strategies, but only in a limited context (Zemblys et al., 2019b; Startsev et al., 2019a; Kothari et al., 2020). In a recent work by Startsev et al. (2019), a number of eye movement event detection metrics were evaluated in an empirical way, demonstrating that they strongly differ at least in some cases. That work served as an initial motivation for a wider-scope comparison and analysis that we undertake in this review.

To the best of our knowledge, no dedicated reviews addressing evaluation strategies for eye movement event detection exist to date. Comparisons of a number of different algorithms (Andersson et al., 2017; Stuart et al., 2019; Startsev et al., 2019a; Dalveren and Cagiltay, 2019) do not specifically focus on the differences of various metrics. Eye tracking methodology books (Duchowski, 2007; Holmqvist et al., 2011) also do not discuss performance measures for event detectors. While Holmqvist et al. (2011, part III) list around 120 "eye movement measures", such as number of fixations or saccades, fixation duration, saccade amplitude, etc., these are statistics meant to quantify the properties of respective events in the recording, not measure the algorithm's detection quality. In "Evaluation using similarity to the ground truth event parameters" we describe how such measures can be "re-purposed" for algorithm evaluation as well.

Remarks on terminology

In this review we strive to use the term "detection" for parsing raw gaze data into events, as in our view it better describes the end result of the eye tracking data processing system that we want to evaluate. Since "eye movement" is used in the literature to describe a complete event, we also interchangeably use "eye movement detection" and "eye movement event detection" to refer to exactly the same process. We provide detailed reasoning for this nomenclature choice in Appendix A and encourage other researchers to use this term to avoid further confusion in the field.

When talking about sample-level labels, however, we will refer to individual samples as "classified as X" or "labeled as X" as a shorthand for "belonging to the eye movement type or class X". Similarly, when talking about confusion matrix-based evaluation metrics ("Confusion matrix-based measures"), we will refer to them as "classification" performance metrics.

When talking about the quantitative outcomes of the evaluation, we often refer to these as "scores" for brevity, as a shorthand for "result of metric computation" – i.e. a numerical value reflecting the quality of the algorithm's predictions as measured by some evaluation metric.

Evaluation protocols

In this section we go through the full pipeline of developing and testing of the eye movement event detection algorithm, preceding to the computation of evaluation metrics themselves. We start from creating, selecting, or acquiring a data pool ("Data sources"), moving to algorithm validation procedures during its development ("Validation procedures"), to a comparison against either competing approaches ("Comparison against other algorithms") or against different eye tracking data quality levels ("Robustness against varying data quality and other data- or algorithm-specific parameters"). We finally offer some remarks on cross-dataset evaluation in "Cross-dataset evaluation".

While this part of the paper does not focus on the particular evaluation strategies themselves (for that, see ensuing "Evaluation methods" and "Evaluation metrics"), we argue that meaningful evaluation does not merely concern the metrics used to test the results of an algorithm,

but touches every aspect of the analysis. For instance, using a biased validation or an overfitting-prone validation scheme, or choosing an unsuitable dataset cannot be helped by switching to a better detection quality measure.

Data sources

This section provides an overview of different sources of eye tracking data relevant for evaluating an eye movement detector. For this, we focus, first and foremost, on evaluation pipelines that rely on the availability of some form of the "ground truth" eye movement events. For the case of evaluation without the ground truth, we refer the reader to the evaluation strategies in the first parts of "Evaluation methods" and the dataset assumptions or requirements in the respective papers. We note, however, and elaborate on this in the corresponding sections, that such strategies can be seen as a sanity check, and not a direct evaluation of the algorithm's performance.

Therefore, when talking about evaluating the performance of an eye movement event detector against some form of ground truth, we essentially mean the comparison of two eye movement data streams – sequences of eye tracking samples each labeled as a certain eye movement type (e.g. fixation, saccade, etc., or *undefined* if no label is available). One of these data streams is referred to as the *ground truth*, the other – as *prediction*. That is, having an output of an algorithm (predicted event labels), we want to establish how similar these labels are compared to the ground truth.

While "ground truth" typically means eye movement labels manually produced by an expert annotator, it is worth noting that, in principle, this type of comparison can and has been applied to various pairs of eye movement data streams (as long as they describe the labels for the same underlying eye tracking signal) – the labels of two experts, two algorithms compared to one another, or e.g. the outputs of the same algorithm before and after eye tracking data quality degradation. Nevertheless, we will refer to the two data streams as the *ground truth* and *prediction* sequences for simplicity.

We assume that the labels in both data streams are provided at the level of individual gaze samples. While this is not usually the way e.g. human annotators would label the data, this is a universal representation, to which annotated continuous events can be converted. Also, it is not necessarily the case that *all* samples receive a label: Some event classes may not be detected by an algorithm, or be irrelevant to the study and thus not annotated. Without limiting the generality, we state that the gaze samples without a label have actually been labelled as an *undefined* class (some works, for example Larsson et al. (2013), do this explicitly in the annotation pipeline). In this case, we can speak of every gaze sample being labelled without imposing restrictions on the event detector or the annotation methodology.

We also note that it is not required that the same *sets* of labels are present in the two compared data streams: An expert could have annotated more event types than the algorithm detects, or vice versa. The only two requirements are that each sample in the ground truth and prediction sequences has a single label (either assigned explicitly or undefined) and that there is a strict one-to-one correspondence between the samples in both sequences (i.e. both sequences map to the same eye tracking data). The special cases where a sample can be annotated with several labels at once – e.g. as a part of a saccade and at the same time a part an optokinetic nystagmus event (Agtzidis et al., 2019) – would be handled by the currently existing evaluation pipelines via performing several evaluations: Testing is divided into such subsets of labels that the requirements above are fulfilled, i.e. the algorithm is tested as a fixation and saccade detector separately from its ability to detect optokinetic nystagmus.

Publicly available datasets

The easiest and most straightforward source of expert annotations for an eye movement event detector is relying on already published datasets. The corresponding publications typically include thorough descriptions of the annotation process and the details of the recording set-up. To facilitate the task of selecting a dataset that could be used for either evaluation or development, we provide a list of publicly available data in Table 1 together with their basic properties, including overall duration, the annotated eye movement types, and notes of how these data were collected. We also provide an online version of the table on the project repository page that can be updated when necessary. An important consequence of the availability of a number of datasets for diverse eye tracking contexts is that, in many cases, using one of the larger existing datasets for algorithm testing or development would be preferential to annotating a small number of recordings for each study (especially for machine learning-based algorithms, having more data for development and testing generally leads to a better algorithm generalization). However creating new datasets is also an extremely valuable contribution to the field and we encourage researchers to make their datasets publicly available. In this case, considerations listed in "Eye tracking data collection and annotation" should be taken into account and described in sufficient detail.

Another important advantage of using publicly available datasets is that evaluation results for other algorithms on these data are often available either online or in papers that used these data themselves. This allows for an easier state-of-the-art comparison, without searching and ensuring

Table 1 List of publicly available annotated datasets to illustrate the variety of readily available material for algorithm development and evaluation. "Duration" reflects the mount of unique eye tracking data; duration in parentheses – the total amount of available annotated data (including undefined samples and taking into account several available annotations for a single recording). Sample distributions do not list proportion of undefined samples and samples annotated as noise, blinks and similar. Note that datasets might have different definitions of fixations, saccades, and other events

Dataset	Duration	Set-up	Sampling frequency	Eye-tracker	Sample distribution
Lund2013† (Andersson et al., 2017)	14.9 min (18.6 min)	Screen-based, pictures, moving dots and video clips	500 Hz††	SMI Hi-Speed 1250	46.49% Fixation 5.88% Saccade 3.34% PSO 41.60% Pursuit
Notes: two expert annotators, fully manually annotated, partial annotation overlap. Includes data that was used in other papers. Download from https://github.com/richardandersson/EyeMovementDetectorEvaluation					
IRF (Zemblys et al., 2018)	8.1 min	Screen-based, fixate-saccade task	1000 Hz	EyeLink 1000Plus	86.77% Fixation 5.65% Saccade 3.00% PSO
Notes: one expert annotator, fully manually annotated. Six participants, data from a replication study (Zemblys et al., 2019a). Download from https://github.com/r-zemblys/irf					
MPIIEgoFixation (Steil et al., 2018)	24.2 min	Head-mounted, unscripted daily life activities	30 Hz	Pupil Pro	74.19% Fixation
Notes: frame-by-frame annotations of one annotator. Download from https://www.mpi-inf.mpg.de/MPIIEgoFixation					
humanFixationClassification (Hooge et al., 2018)	5.9 min (70.4 min)	Screen-based, pictures and search task	300 Hz	Tobii TX300	71.82% Fixation
Notes: 12 expert annotators, fully manually annotated, all annotation data overlap. 10 adult free viewing and 60 infant search task (Hessels et al., 2016) trials. Download from https://github.com/dcnieho/humanFixationClassification					
360EM (Agtzidis et al., 2019)	32.9 min	Head-mounted, naturalistic 360° videos	120 Hz	FOVE	*Primary labels* 75.15% Fixation 10.44% Saccade 9.76% Pursuit *Secondary labels* 0.81% OKN 27.64% VOR 15.84% OKN+VOR 1.47% Head pursuit
Notes: two stage annotations of one expert annotator after training and discussion session. First stage (*primary labels* and optokinetic nystagmus – OKN – or nystagmus) uses pre-labelled saccades and does not account for the head motion. Second stage (vestibulo-ocular reflex – VOR, VOR + OKN, Head pursuit) uses labels from the previous stage that are re-examined in the context of the eye-head coordination. Ca. 3.5 h of eye- and head-tracking recordings, ca. 16% annotated. Download from https://gin.g-node.org/ioannis.agtzidis/360_em_dataset					
GazeCom (Startsev et al., 2019b)	4.7 h (14.1 h)	Screen-based, naturalistic video	250 Hz†††	EyeLink II	73.96% Fixation 10.67% Saccade‡ 9.83% Pursuit
Notes: manual annotations of one expert tie-breaking and adjusting labels of two novice annotators. Novice annotators (paid undergraduate students) used pre-labeled data and went through the data twice. Labels of novice annotators are available. Download from https://gin.g-node.org/ioannis.agtzidis/gazecom_annotations					

Table 1 (continued)

Dataset	Duration	Set-up	Sampling frequency	Eye-tracker	Sample distribution
Hollywood2EM (Agtzidis et al., 2020)	2.15 h (4.3 h)	Screen-based, movie clips	500 Hz	SMI Hi-Speed 1250	59.46% Fixation 9.87% Saccade‡ 26.54% Pursuit
Notes: manual annotations of pre-labeled data, two stage annotation (paid student followed by an expert coder). Labels of student annotator are available. Download from https://gin.g-node.org/ioannis.agtzidis/hollywood2_em					
Gaze-in-wild (Kothari et al., 2020)	3.06 h (4.15 h)	Head mounted, naturalistic tasks	300 Hz‡‡	Pupil Labs + custom setup	12.50% Fixation 7.12% Saccade 2.65% Pursuit 26.72% VOR
Notes: independent annotations of five trained annotators, ca. half of the data is annotated. Naturalistic tasks: indoor navigation, ball catching, object search, tea making. Download from http://www.cis.rit.edu/rsk3900/gaze-in-wild					

†Named as such by Zemblys et al. (2019b)

††Part of the data were recorded at 200 Hz (Friedman, 2020; Zemblys et al., 2020)

†††Data for subject "SSK" were recorded at 500 Hz (Startsev et al., 2019a, p. 559)

‡PSOs annotated as a part of saccades

‡‡Upsampled from the original 120 Hz data due to synchronization with other devices. Upsampled using low-pass filtering and Piecewise Cubic Hermite Interpolating Polynomial (Kothari, R., personal communication, May 29, 2020).

reasonable operation of the latest algorithms (especially since their complexity is increasing). It should be noted, however, that simply comparing published scores to the ones obtained using a new algorithm does not ensure valid evaluation if any of the evaluation details differ. For that reason, some of the datasets either provide the outputs produced by the tested algorithms directly (Agtzidis et al., 2019; Startsev et al., 2019b; Agtzidis et al., 2020), eliminating the need to exactly match the evaluation strategy reported in the corresponding papers, or provide code to run the evaluation for the new approaches (Hooge et al., 2018; Startsev et al., 2019b; Kothari et al., 2020).

Eye tracking data collection and annotation

It can often be the case that the available datasets do not satisfy the requirements for the study, e.g. there are no recordings in a scenario of particular interest to the researchers, with a particular device, etc. In that case, a collection of the dataset may be undertaken, with an ensuing annotation effort. The same considerations we describe here for the dataset collection apply when choosing a publicly available dataset, whether already annotated or that will be labelled during the planned study.

Collection

The primary considerations for dataset conception could probably be condensed to the eye tracking experiment set-up, including the type of stimuli (static vs. moving; synthetically generated and controlled vs. naturalistic vs. the real world; task vs. no task given to the observers, etc.), the eye-tracker type (screen-based vs. wearable) and frequency (from 30 Hz for the majority of currently existing wearable systems to at least 2000 Hz in state-of-the-art screen-based trackers). For eye movement detectors, these factors essentially influence (1) their applicability to this domain (for already published approaches) and (2) the eye movements that one can expect to detect. E.g. testing a microsaccade detector on 30 Hz data will likely not yield meaningful conclusions.

The size of the dataset plays a role as well: A larger amount of labelled eye tracking recordings allows for a more reliable performance estimate, and enables a larger-scale optimization. Depending on the scale of the study, varying amounts of data can be annotated in its context. This is not, however, to say that the recordings that are not annotated are collected in vain: Firstly, annotation can also take place at a later time point, allowing for expanding the scale of the dataset by investing additional expert time only, i.e. without the need to reproduce the set-up potentially years after the initial recordings were collected. Secondly, there are more immediate benefits to be had even from unlabelled data: (1) Some approaches allow learning or inferring important patterns from data with no annotation (the unsupervised learning research area), or benefit from more unlabelled data being available. The algorithm in Startsev et al. (2019b), for instance,

demonstrated improving performance with the increasing number of recordings per stimulus video clip during its testing. (2) While undertaking quantitative evaluation on the data without the ground truth is not always a sensible option, qualitative examination of the developed algorithm's output on independent data that were definitely not used during its fine-tuning is always of value.

Annotation

Annotating eye tracking data is a difficult and time-consuming process. The time required for labelling e.g. one second of eye tracking the data can range between 10 and 60 seconds in different paradigms (Hooge et al., 2018; Startsev et al., 2019b; Agtzidis et al., 2019; Kothari et al., 2020), and months of man hours might be needed to collect a representative set of annotated recordings. Annotation is usually done by the field experts that use graphical interfaces with visualizations of various representations of raw or filtered eye tracking data, such as gaze position over time, gaze speed over time plots, 2D scanpath plots, etc. (Hooge et al., 2018, Fig. 1; Agtzidis et al., 2019, Fig. 1; Startsev et al., 2019b, Fig. 2; Kothari et al., 2020, Fig. 7). Experts then either freely, solely based on their experience, or following some preset rules label onsets and offsets of events – fixations, saccades, PSOs, smooth-pursuit, etc. – for which the annotations are desired. In case of explicit rules being used for annotation, it can be undertaken by trained non-experts as well. It has to be remembered that as the algorithm, the parameters of which are optimized to produce the outputs better resembling the events observed in the annotated data, will inherently be attempting to replicate the annotation patterns present in these data, the quality and consistency of the annotations to a large extent define the quality of the predictions.

One approach to speed up the typically lengthy annotation process (especially when multiple event types are being annotated) is to first detect (a subset of) events using an algorithm, and then ask for the experts to confirm and modify these pre-annotated events (Agtzidis et al., 2019; Startsev et al., 2019b). This approach might, however, bias expert decisions towards agreeing more with the algorithm's predictions. It is, therefore, advisable to employ a very simple algorithm (i.e. one with only few parameters that usually cannot cope with complex data) for this pre-annotation and encourage annotators to make as many edits as they feel is necessary (Agtzidis et al., 2020).

In addition to choosing a suitable graphical interface (e.g. see discussion in Agtzidis et al., 2019 for the head-mounted eye tracker case), the number of raters annotating the same eye tracking signal needs to be considered. This is especially important when it comes to evaluating the results of an algorithm, as well as putting them into context. For some applications, the discussion of inter-coder differences may not be of interest (Hoppe & Bulling, 2016; Steil et al., 2018; Zemblys et al., 2018). In other cases, the output of an algorithm may be compared to multiple codings or a consensus coding (Andersson et al., 2017; Friedman et al., 2018; Bellet et al., 2019; Zemblys et al., 2019b, etc.) and, therefore, multiple raters performing the annotation are required. Comparing the differences between the algorithm's labels and those of the human annotators to the inter-rater variability e.g. may be specifically beneficial, indicating to which extent the imperfections in the algorithm's predictions may be irreducible.

Synthetic data

In parallel to real data collection and annotation, there is a possibility of simulating eye tracking recordings. This approach could ensure virtually unlimited amounts of data for eye movement algorithm development and evaluation. Simulation enables objective ground truth labels in the sense that it allows controlling various properties of the data, such as the exact order and duration of events, level and properties of the noise, as well as sampling rate and various event-related parameters – amplitude and velocity of saccades, fixational drift, amplitude of PSOs, etc. However, the main drawback of using synthetic data is that they still differs from real recordings and in many cases are just idealised representations of a subset of all the possible cases. In addition, all properties of raw data and events that are used to simulate recordings are usually derived from models previously published in the literature or from real data using computational approaches, meaning that the same skepticism of whether "real" events exist applies to the synthetic data as much as to the manually labeled real data.

Despite the limitations of synthetic data, researchers have used it for algorithm development and validation. Otero-Millan et al. (2014, Fig. 2) simulated microsaccade sequences using a microssacade template (an average shape of manually labeled microsaccades) and used them to compare two microsaccade detection algorithms. Synthetic eye tracking signal was generated by randomly inserting microsaccades with random peak velocities and adding noise, generated by the 10^{th} order autoregressive process and a multiplicative low-frequency component of white noise. Dai et al. (2016) proposed a parametric model for saccadic waveforms and used simulated saccades and different noise levels to evaluate four common saccade detection algorithms. One step further, Fuhl et al. (2018) used artificial data to train a machine learning based event detection algorithm. Authors demonstrated that their algorithm, even without having "seen" any of the real

recordings in its training, performs on par with three other traditional algorithms when tested on three different datasets of real eye tracking recordings. Zemblys et al. (2019b) employed an end-to-end approach (i.e. achieving the goal while bypassing the intermediate steps typically present in traditional pipeline designs, more specifically, using raw gaze data to get desired output without employing filtering, feature engineering, thresholding and similar steps) to generate eye tracking data[3] with fixations, saccades, and PSOs. The authors trained a recurrent neural network – gazeGenNet – using only 44 seconds of manually labeled data, which was then able to generate signal with properties similar to the real eye movement recordings. This synthetic data was subsequently used to train a neural network – gazeNet – for detecting the eye movement events. The authors concluded that their approach was capable of generalizing to other datasets well, outperforming the state-of-the-art event detection algorithms.

Validation procedures

When evaluating an eye movement event detection algorithm, it is essential that evaluation is performed using a dataset (or part of it) that was not used when developing and fine-tuning the algorithm. This is a general rule in algorithm development and machine learning, but it becomes somewhat more specific in the context of the eye movements.

In general, the data are split into the *development* and *test* subsets. The development set represents the part of data, with the help of which any of the parameters of the algorithm are optimised. These parameters can be e.g. thresholds for a conventional event detection algorithm or weights and meta-parameters when training a machine learning-based model. In the latter case, it is very common to once again split the development set, where the typically larger part (the *training* set) is used for actual training, and the typically smaller part (often called *validation* set) – for early-stopping and meta-parameter optimization. Using the validation set should also prevent the optimization process from *overfitting* to the training set, as the difference in detection performance on the two can be monitored during the training process, guiding the developer to make corresponding decisions and adjustments if the discrepancy increases unacceptably.

In principle, this training-validation-test subdivision of the dataset should be applied anytime a non-negligible optimization of the algorithm's parameters takes place, even if it does not involve machine learning.

The test set should, ideally, be used just once for each developed method, after all training, optimization, and modifications have already taken place based on the other subsets. This prevents the developers from "cheating" and giving their algorithms an unfair advantage on the test set, compared to competing algorithms.

There is no standard way of performing the data splitting for eye tracking data, and researchers have employed various approaches: Larsson et al. (2013) divided their dataset into two parts by assigning the participants into two equally large groups – one for training and one for testing. Hoppe and Bulling (2016) randomly selected 75% of the data for training while the remaining 25% were equally split into validation and testing sets (although the authors do not specify how exactly they split the data). Similarly, Zemblys et al. (2018) used 20% of the data (one of the five subjects) for testing, while the rest were split into the training and validation sets by randomly selecting continuous part of each of the trials (corresponding to 25% of total length) to the validation set.

Various types of *cross-validation* are also quite common, as the typically small amounts of available annotated data discourage the researchers from evaluating on a single allotted subset of the data. Consequently, algorithm developers choose to iterate through different parts of the entire dataset and use those for testing, while the rest is used for training and validation. The results of all iterations can then be aggregated to reflect the algorithm performance score on the whole dataset. For example, Anantrasirichai et al. (2016) trained their model ten times by randomly assigning half of the data to the training and test sets on each run; Startsev et al. (2019a) employed 18-fold cross-validation (each fold corresponding to all recordings for a single stimulus video), while Bellet et al. (2019) used a fixed testing set and 10-fold cross-validation for algorithm tuning and optimization. We comment on different strategies to perform both train-test and cross-validation splitting of a single dataset in the section below.

A more thorough algorithm development and testing procedure would be to use *cross-dataset* validation. The training and testing datasets then might be recorded in different labs using different eye trackers, have different sampling rates, and differ in data quality. A more detailed description of cross-dataset evaluation is provided in "Cross-dataset evaluation".

How to split the data

Albeit several relatively large datasets with the ground truth eye movement annotations have become available in recent years (cf. Table 1), they are still relatively rare, and typically contain recordings for a number of observers viewing a certain collection of stimuli. While the lack of the amount of available data limits extensive testing of the algorithms' generalization, the implications of the recording scenario are indirect but no less important. In particular, this issue makes it cumbersome to ensure that the training and the

[3] Available from https://zenodo.org/record/1476449

test subsets do not overlap in terms of possibly critical higher-level information.

Consider, for instance, two higher-level properties of an eye tracking recording: *who* is watching *what*, i.e. the observer identity and the stimulus. It is very easy to ensure that the individual gaze samples are not shared between the training and testing data. However, ensuring that both the stimuli and the observers viewing these stimuli are different between the training and test sets presents a more challenging problem. Avoiding the observer and the stimuli information leak from the training to the test set would be very impractical – it would mean that a potentially large part of the dataset would remain unused, either for training or for testing (see Appendix B).

To understand why the *who* and the *what* questions are important for the purpose of dataset splitting, consider works on observer identification via eye movement-derived features (Rigas and Komogortsev, 2017, e.g.) and on the identification of the viewing context – e.g. activity of the observer (Bulling et al., 2010; Kunze et al., 2013, to name a few). If information about both the *who* and the *what* can be inferred based on eye movements, the inverse may also be true, and optimizing an eye movement-related model on the data from the same observers or for the same stimulus as in the test set could yield biased results.

Based on the reasoning above, it would seem optimal to collect datasets with a single recording from each subject, each corresponding to a unique stimulus – in that case, any split of the resulting eye tracking recordings would automatically ensure both stimulus- and observer-level independence. This is, of course, both unrealistic and inefficient, and would complicate certain related usages of the data, where statistics over a number of viewings are desired (clinical population viewing patterns' analysis (Gutiérrez et al., 2020) or saliency prediction (Judd et al., 2009), e.g. analysed jointly with eye movements (Startsev & Dorr, 2020)).

We elaborate specifically on the stimulus dependency of the eye movements on an example of smooth pursuit. This eye movement type is very much dependent on the movement patterns seen by the observer. Both Dorr et al. (2010) and Mital et al. (2011) noted that the gaze patterns of different viewers are more similar when moving objects are presented. Additionally, Meyer et al. (1985) reported that the human gaze matches the target speed rather accurately during smooth pursuit up to ca. 100 °/s, which means that pursuit gaze signals for different observers will be similar, when the same targets are followed. This similarity was quantitatively verified in Startsev et al. (2019b) as well, meaning that when an algorithm is trained on gaze data with the same smooth pursuit targets as in the test set, there is a chance it will overfit for this information, and the results on the test set will be overly optimistic.

In Startsev et al. (2019a), two modes of cross-validation were compared: One where the train and test sets never contained gaze recordings for the same observers, and one where they did not overlap in terms of the viewed stimuli (video sequences, in that case). While for fixation or saccade detection there was no large difference between the validation modes, smooth pursuit demonstrated a stronger dependency on the video signal than on the observer: The algorithm trained and tested on gaze recordings of different observers but for the same stimuli was more prone to overfitting compared to the algorithm trained and tested on the data of the same observers but non-overlapping video sets.

Time-wise splitting for eye tracking data. If this level of separation is impossible or impractical, one can also split data time-wise, i.e. based on the *part* (in time) of the viewed stimuli. On an example of video data, a random (continuous) subset of each video's duration can be used as an evaluation set (see schematic illustration in Fig. 20 in Appendix C). We validated the usefulness of this data splitting scheme for a practical eye movement detection application in a preliminary experiment (see details in Appendix C), where train-validation-test split is performed. There, the test set was always chosen in the same way, but two methods to perform the train-validation split of the remainder of the data were implemented: random sampling (of windows of gaze data) and the proposed time-wise splitting. The model with time-wise disjoint training and validation sets demonstrated consistently better test performance. We thus strongly advocate for ensuring that data set splitting be performed in such a way that all gaze signal corresponding to identical (parts of the) stimuli would be assigned to the same data set part.

Evaluation procedures

In principle, an eye movement detector *could* be meant to only be used with e.g. data recorded with one eye tracker. In this case, its evaluation with the help of a dataset recorded with that particular tracker can serve as an adequate representation of future-use conditions. Even in this case, however, it has to be noted that data quality (even obtained with the same device) can vary widely because of the subject-specific characteristics, subject population, the operator, and recording protocol (Holmqvist et al., 2011; Holmqvist et al., 2012; Nyström et al., 2013; Blignaut and Wium, 2014; Hessels et al., 2015).

Therefore, if the test set of recordings is not diverse enough (either in terms of the utilized devices or recording conditions), additional robustness evaluation should be considered (e.g. additive noise, etc.). If the algorithm represents a general approach to event detection,

evaluation should also include testing on as many datasets as reasonably possible. Researchers could also consider resampling the data to simulate a wider range of sampling frequencies, on which the algorithm should be able to operate. We elaborate on various considerations connected to these aspects of the evaluation set-up in the remainder of this section.

Comparison against other algorithms

In order to draw conclusions from the performance scores of a developed eye movement detector, examining these in isolation may not be enough, as this does not necessarily provide sufficient context for their interpretation (e.g. in order to judge whether the developed algorithm advances the state of the art). Thus, quantitative and qualitative results of the evaluation can benefit from comparison to other algorithms in the literature, ideally tested under the same conditions. While it does require additional effort, this is the most straightforward way to demonstrate the benefits of the proposed method. Given the multitude of publicly available algorithms published in recent years, it is difficult to justify completely abstaining from this type of comparison. On the other hand it has to be admitted that not all publicly available code repositories are easy to navigate or utilize.

An important consideration to remember is that while comparing newly developed eye movement detectors to such established and simple algorithms as I-VT or I-DT (Salvucci & Goldberg, 2000) is very appealing, they can no longer be deemed to represent an adequate approximation of the state of the art in eye movement detection, and thus would likely in many scenarios be a very weak standard of performance. This is especially true when the data are somewhat more complex than what these algorithms were designed to deal with – include noise (e.g. originate from infants, see Hessels & Hooge, 2019) or include movement either in the stimulus (Mital et al., 2011; Larsson et al., 2016; Startsev et al., 2019b), or of the observer themselves (Kinsman et al., 2012; Agtzidis et al., 2019; Kothari et al., 2020). While in e.g. Steil et al. (2018) I-VT and I-DT were used as baseline fixation detectors in wearable eye tracking data, it is worthwhile noting that especially in this context it is known that simple few-statistic thresholding of the gaze speed or its absolute movement is an inherently flawed approach to eye movement detection due to the more complicated gaze-head-world dynamics (Kinsman et al., 2012).

Slightly more flexible yet very easy-to-implement algorithms include I-VVT and I-VDT (Komogortsev & Karpov, 2013) that additionally enable the detection of smooth pursuit. For head-mounted eye tracking specifically, a more versatile simple algorithm (I-S^5T – relying on five speed thresholds to disentangle head and eye-in-head movement) was introduced in Agtzidis et al. (2019) as well.

An important note and warning about using any of the appealingly simple to implement thresholding-based approaches is that their performance on any given dataset is heavily dependent on the thresholds themselves, and their values optimized on one dataset are not (or at least not necessarily) suitable for another (Steil et al., 2018; Startsev et al., 2019a). Therefore, it is reasonable to systematically test a variety of threshold values or their combinations in order to achieve the best reachable performance level of the given approach. This effectively leads to overfitting, but, taking into account the incapability of very simple methods to perfectly fit the complex real-world data, it could in many cases be desirable to let the baseline method overfit when comparing a newly developed method to a simple baseline. This results in comparing the new approach with the best-case-scenario version of the baseline, making for a stronger argument of the proposed method's superiority. For example, Steil et al. (2018) iterated over a range of thresholds for I-VT and I-DT, demonstrating their strong dependency on the threshold value; in Startsev et al. (2019a), a grid-search for threshold pairs of I-VVT, I-VDT, and I-VMP (Lopez, 2009) was performed, and the best threshold pair for each method was used in the final comparison.

It is as important to point out that when a modern highly-parameterized detector is used as a baseline, but its pre-trained version cannot be used directly (e.g. the out-of-the-box performance is unreasonably poor, or due to technical reasons such as eye tracker data frequency, etc.), letting such an algorithm overfit will lead to an unrealistic estimate of its performance, and thus an unfair comparison to any other method. In this case, for maximal fairness, both the developed algorithm and the baseline approach should be optimized in a similar manner so that neither overfits the data.

As a fallback option, when comparison to already published methods is for some reason impossible or would only be meaningful for one or two other methods, another layer of validation can stem from testing against what was referred to as "baseline eye movement classifiers" in Startsev et al. (2019) – algorithms that produce eye movement labels *without* taking the gaze signal itself into account. Comparing the developed algorithm to some of those methods allows the researchers to quantify the extent to which their method improves on just leveraging aggregated eye movement statistics or patterns in the dataset that is used for the analysis. This could also point towards the limitations of the employed dataset – its implicit biases or lack of variability.

Robustness against varying data quality and other data- or algorithm-specific parameters

Here we touch on additional strategies that are used to test the robustness of eye movement detection algorithms. A generally applicable strategy consists of testing the detection algorithm against the same underlying eye tracking data, but while varying its data quality (with the assumption that the corresponding eye movement labels remain the same). For example, original data can be resampled when testing the algorithm's robustness against different sampling rates, or artificial noise can be added if the algorithm is expected to be applied to data from different eye trackers or different populations (e.g. infants or those with eye movement-related disorders) – i.e. data with varying noise levels. The desired outcome for the tested algorithm is that the detected events' statistics and other performance measures should remain similar across a range of sampling frequencies, noise levels, and other aspects of data quality.

Hessels et al. (2017), for example, tested their algorithm by adding constant and variable noise up to 5.57° RMS (also known as RMS-S2S – the root mean square of the sample-to-sample distances) and simulating different levels of data loss. Authors showed that their algorithm produced similar numbers of fixations, mean (and standard deviation) of fixation duration for noise levels up to 5° RMS and was able to deal with data loss. In Zemblys et al. (2018) the authors resampled 1000 Hz eye tracking signal to sampling frequencies between 30 and 1250 Hz, as well as added white Gaussian noise resulting in data with noise from 0.0083° to 7.076° RMS in different parts of the screen. Performance of the examined algorithm remained similar for the data frequencies of 120 Hz and higher, and with an average noise levels up to 0.26° RMS. Detected events were then compared to the expert annotations via both event parameters and Cohen's kappa metric. Similarly, Peng et al. (2019, Figure 7) used additive white Gaussian noise, demonstrating that their algorithm performs better than the competing approach for signal-to-noise ratios between 38 and 50.

Interestingly, however, Niehorster et al. (2020) recently demonstrated that adding neither white noise nor the noise simulating based on a real eye tracking recordings is sufficient for an extensive testing of an algorithm's resilience to noise. The authors argue that both – white and the signals measured from a specific eye tracker – are only points in a continuum of noise types. Therefore, a thorough noise robustness analysis (representative of the data from many different eye trackers) should include tests on the whole range of possible noise types[4].

An algorithm-specific robustness testing was undertaken in Startsev et al. (2019b): The algorithm described there relies on clustering the gaze samples from multiple observers' recordings for the same stimulus. While the full dataset contained the recordings of ca. 47 observers per stimulus, it was not clear how exactly the number of available recordings influences the method's performance. In combination with both sample- and event-level quality measures, a quantitative evaluation was performed at different observer subset sizes, thus systematically exploring the dimension of the parameter space to which the method would clearly be sensitive.

A very important note at this point is that for any type of robustness testing, one can freely adjust the parameters of the algorithm *as long as it is done with fairly obtained information* (i.e. as long as this adjustment can be performed for any independent and unseen data to which the method can be applied). For instance, the noise level can be a parameter of the algorithm as long as it is not taken directly from the used noise generator, but computed based on some gaze signal statistics. Sampling frequency can also be a parameter as it can be inferred from the gaze samples' timestamps, etc. For instance, the algorithm by Startsev et al. (2019b) provides a formula to adjust one of its main clustering parameters based on the sampling frequency and the number of observers, and the algorithm of Dar et al. (2020) is parameterized by both the sampling frequency and the noise factor (among others).

Cross-dataset evaluation

As mentioned previously, various eye trackers and participant populations can lead to a gaze signal with widely varying properties. Therefore, whenever relevant for the intended use of the algorithm, a thorough algorithm evaluation should include testing using multiple datasets. For example, Bellet et al. (2019) used four datasets: two 1000 Hz Eyelink1000 datasets with human subjects, 500 Hz Eyelink1000 recordings from a single macaque monkey and 1000 Hz data collected from three rhesus macaque monkeys implanted with scleral search coils. Hauperich et al. (2020) compared their microsaccade detection algorithm to another algorithm using two different datasets – Lund2013 (500 Hz SMI Hi-Speed 1250 recordings (Andersson et al., 2017))

[4]Niehorster et al. (2020) provide fixational noise generator code at https://github.com/dcnieho/FixationalNoise_generator

and in-house 1000 Hz Eyelink1000 recordings. Dar et al. (2020) used the same Lund2013 dataset and two 1000 Hz Eyelink1000 datasets – one from a laboratory setting and another recorded using telephoto lens during simultaneous fMRI acquisition thus yielding lower data quality.

When the developed algorithm is meant to only be used with a certain eye tracker or a certain participant population, multiple datasets in the relevant experiment set-up, but recorded on separate occasions, by different operators, in different environmental conditions, with different relevant stimuli types, etc., could be used for a thorough evaluation.

In some cases when the algorithm is designed to only work on a data with a specific sampling rate, cross-dataset evaluation poses certain challenges – data need to be resampled to algorithm's "native" sampling rate. For example, Zemblys et al. (2019b) upsampled the 250 Hz GazeCom (Startsev et al., 2019b) and the 300 Hz humanFixationClassification (Hooge et al., 2018) datasets to 500 Hz using first-order spline interpolation before evaluating their algorithm.

Downsampling the data is not a trivial matter either. First, the data need to be low-pass filtered to avoid *aliasing*. For instance, Zemblys et al. (2018) used a Butterworth filter with a window size of 20 ms and cut-off frequency of 0.8 times the Nyquist frequency of the new data rate. Voloh et al. (2020) downsampled their original 300 Hz data to 150 Hz although did not specify any details while Pekkanen and Lappi (2017) downsampled simulated 1009 Hz data simply using linear interpolation.

Downsampling the ground truth event annotations poses another challenge, as data might become incoherent. Consider e.g. a fixation followed by a very short saccade and a longer PSO. When downsampling to a very low sampling rate, this saccade might become too short to span even one sample and, therefore, be removed, resulting in an annotation where a fixation is followed by a PSO. Houpt et al. (2018) report that after downsampling 1000 Hz data to 30 Hz, 73.1% of the saccades and 3% of other events became one sample long (yet it is unclear how many events were removed altogether). A possible workaround for "disappearing" events could be ensuring that no event will be downsampled to 0 samples, but this does not guarantee consistent event labels in all cases either.

Evaluation methods

The most common evaluation method – comparison between two sources of eye movement event labels can be carried out in two principally different ways – either by comparing the labels of each individual gaze point in the two sources (i.e. *sample*-level evaluation), or by comparing entire oculomotor events – uninterrupted sequences of samples with the same label in either of the class label sources, such as complete fixations, saccades, etc. – i.e. *event*-level evaluation. Evaluation on the level of samples does not allow for many options, although a few different metrics are used in the literature: accuracy or disagreement rates (Anantrasirichai et al., 2016; Hoppe and Bulling, 2016; Andersson et al., 2017, etc.), Cohen's kappa values (Larsson et al., 2013; Hooge et al., 2018; Zemblys et al., 2018, etc.), as well as sensitivity, specificity, or F1 scores (Santini et al., 2016; Peng et al., 2019; Bellet et al., 2019; Startsev et al., 2019a, etc.).

On the other hand, a growing number of event-level evaluation strategies exist already, often principally different from one another: average statistics of the ground truth and detected events, such as duration, amplitude, main sequence, etc. (Andersson et al., 2017; Zemblys et al., 2018; Dar et al., 2020), or comparing the distributions of such event statistics (Startsev et al., 2019b); different ways of computing F1 scores (Hooge et al., 2018; Startsev et al., 2019a); variations of the Cohen's kappa (Zemblys et al., 2019b; Startsev et al., 2019); temporal offset measures of Hooge et al. (2018); average intersection-over-union ratios in (Startsev et al., 2019a; Kothari et al., 2020); Levenshtein distance between event label sequences (Zemblys et al., 2019b, can be applied on the level of gaze sample labels as well).

In addition to the availability of the ground truth and the level of the evaluation (individual samples or events), selection of the evaluation method should also be guided by considering whether the evaluation treats the eye movement detection algorithm as a tool to achieve a further goal (e.g. calculate saccade main sequence, use algorithm output in a human-computer interaction, etc.), or as the end-goal in its own right. The latter evaluation perspective usually requires evaluation to be very descriptive and often very concise, and is mostly a domain of algorithm developers. Below we list available evaluation methods and discuss the context in which these can be applied.

In this section, we first cover the evaluation methods that are not directly related to quantifying the detection performance itself, starting from event statistics-based evaluation, to stimulus parameters-driven methods, to application-based eye movement detector assessment. The last part of this section ("Detection performance evaluation") focuses on the direct evaluation of the event detection capabilities of the tested algorithm. This part is most relevant to the remainder of this review, as it reflects the need for generally applicable unified evaluation in order to facilitate algorithm comparison between publications, and thus contribute to clearer understanding and easier collaboration between different researchers. This naturally entails a variety of decisions that ideally need to be taken in a uniform way across different papers, spurring the discussion in the field.

Evaluation based on eye movement metrics

Evaluation methods that are not directly linked to quantifying the detection performance of the algorithm itself and in some cases do not even require the ground truth are mostly based on descriptive event statistics – such as average fixation duration or mean saccade amplitude – and statistical tests. In addition, several other computational methods that enable a more in-depth performance analysis exist and are described in the sections below.

If stimuli properties with predictable effects on gaze behavior, e.g. size, position, or trajectories of salient targets, etc. are unknown, the most obvious evaluation of the detected events is computing average eye movement statistics. A few examples include:

- number or frequency of fixations and saccades (e.g. detecting a dozen saccades per second on average is indicative of false detections);
- distributions of event durations, saccade amplitudes, etc. (e.g. the distribution of detected smooth pursuit durations peaking very close to zero indicates likely fragmentation of the detected events);
- main sequence (i.e. an empirically known relationship between saccade magnitude and peak velocity; if the saccades identified by the model follow the typical pattern with few outliers, these are likely realistic – nothing can be said about the amount of missed detections, however.

These and other eye movement parameters have been extensively studied and, therefore, the algorithm's ability to produce plausible event statistics is a sign of reasonable performance. It is important to point out, however, that this is merely a "sanity check" for the detections, as even the well-understood properties of eye movement sequences depend on the stimulus and instructions, even in similar-sounding paradigms: For video clip viewing, using different stimulus material can lead to different distributions of gaze on the screen – e.g. increased center bias for cinematic material, compared to naturalistic videos (Dorr et al., 2010) – and thus influence such basic properties as saccade amplitude and duration distributions (Agtzidis et al., 2020). In the set-ups where observers are instructed with regards to their viewing behavior (e.g. to look at the moving dot or to follow moving objects as much as possible) the influence is even more directly reflected in the eye movement statistics.

A possible additional use of the evaluation procedure above is to find the range of algorithm's settings that result in stable performance, i.e. finding such parameter values, where small changes in those will not lead to a drastic change in the considered statistics (Komogortsev et al., 2010). Similarly, Hessels et al. (2017) evaluated a number of algorithms against themselves to examine the robustness of the eye movement measures to additive noise and simulated data loss. It is, however, important to note that this type of evaluation still needs additional analysis employing other evaluation methods to confirm that these stable measures are "correct" and make sense. Without additional analysis it is quite possible that an otherwise useless algorithm, e.g. one that identifies all samples as fixations, will score the highest as its output would appear very stable despite perturbations in its parameters or the noise added to its inputs.

Evaluation using similarity to the ground truth event parameters

This type of evaluation is based on the assumption that a better performing algorithm should also produce event detections with properties more similar to these of the ground truth. For example, Andersson et al. (2017, Tables 2, 4, 5, 6 and Fig. 2) compared mean duration, standard deviation, and number of fixations, saccades, and PSOs obtained from two human coders and ten event detection algorithms. To summarize these three parameters with a single similarity measure, the authors calculated root-mean-squared deviations (RMSD, Andersson et al., 2017, Eq. 2) for all algorithms against the two human coders. The algorithm having the minimum RMSD was considered to be the most similar to the humans experts.

Hooge et al. (2018, Figs. 3 and 5) compared the number of fixations and saccades, average fixation durations, and saccade amplitudes across twelve expert coders. Zemblys et al. (2018) used a similar approach and compared the ground truth event properties – the number and duration of fixations, saccades, and PSOs – to the output of their random forest based algorithm (IRF). To evaluate how well the algorithm reproduces the main sequence compared to manually coded data, the authors calculated main sequence and amplitude-duration relationships using the ground truth labels (Zemblys et al., 2018, Fig. 13) and then used the coefficient of determination (R^2) to compare these relationships to the ones obtained when using the output of IRF and the Nyström and Holmqvist (2010) algorithms.

Instead of comparing summary statistics, Startsev et al. (2019b, Table 4) used Kullback–Leibler divergence and histogram intersection similarity to quantitatively evaluate the correspondence between the distributions of the ground truth and predicted smooth pursuit durations.

Two examples of using distribution-related analysis for *qualitative* evaluation are presented by Dar et al. (2020) and Otero-Millan et al. (2014). The former work used main sequence and distributions of event durations to qualitatively demonstrate that their algorithm, when tested on two different datasets, produces events with similar properties (Dar et al., 2020, Figs. 3 and 4). In Otero-Millan

et al. (2014, Fig. 3), the authors compared the distributions of microsaccade amplitudes and peak velocities, confirming that their unsupervised clustering based algorithm produces reasonable values.

Evaluation based on stimuli parameters

The underlying intuition of the methods described in Komogortsev et al. (2010) and Komogortsev and Karpov (2013) and summarized in this part of the review is that a non-negligible amount of information about the gaze behavior of the observers is known beforehand when the observer has been instructed to follow a dot (or another target) that is being programmatically moved on the screen. In this way, the eye movements of the observer are "predefined" by the stimulus: The periods of time when the dot is stationary encode fixations (with a known duration and location), when it is jumping from one point to another – a saccade is encoded (with a known amplitude), and when it is smoothly moving along a certain trajectory, a smooth pursuit is expected (with a speed matching that of the target). This paradigm presumes either a single gaze target, or instructions to the observer that sufficiently define gaze behavior in the presence of multiple targets. This is a strong limitation, but it can be suitable for some experimental set-ups.

Based on these assumptions, several scores can be computed to compare the detected eye movements to those encoded in the stimulus. The authors refer to these as *behavioral scores*. One group of these scores is referred to as "quantitative", and describes (1) the proportion of fixation samples in the vicinity of the simultaneously displayed stationary gaze targets, (2) the ratio of the sum of all detected saccade amplitudes to the sum of all target jump amplitudes in the stimulus, and (3) the ratio of all detected smooth pursuit tracks to all the smooth pursuit target tracks in the stimulus. "Qualitative" score group describes the spatial proximity of the centres of detected fixations and pursuit samples from the respective simultaneously displayed targets, as well as the speed difference in detected and stimulus-encoded pursuits. One last additional measure is defined as the ratio of smooth pursuit-labelled gaze samples that occurred when only fixation targets were being displayed. This is an approximation of a false positive rate metric for smooth pursuit detection, and can be seen as a much simplified version of a similar video-based measure later proposed by Larsson et al. (2016, see also "Video-based evaluation for smooth pursuit detection").

Overall, while behavioral scores do not require manually annotating the data in order to produce a numerical quality measure for an eye movement detector, they have very limited applicability – one has to know in advance which eye movements have to (or are at least very likely) to happen at which time. Additionally most of the measures are affected by the precision and accuracy of the eye tracking data, subject's ability and willingness to follow the instructions, and also properties of subject's oculomotor system. For example, larger saccades to distant targets are executed in multiple steps – usually one large, inaccurate saccade and one or multiple corrective saccades (Van Gompel et al., 2007, Chapter 13; Laurutis and Zemblys, 2009). A well-performing algorithm that correctly detects all of these saccades is likely to score lower compared to the algorithm that only detects one large saccade. The sum of such multi-step saccade amplitudes might be considerably larger than the sum of target jump amplitudes because of overshoots and inaccurate landing positions of large saccades.

Video-based evaluation for smooth pursuit detection

Larsson et al. (2016) introduced an approach to evaluate smooth pursuit detectors based on whether its detections correspond to the motion patterns in the video (since smooth pursuit can be defined as following a moving target with the movement of the eyes). This falls somewhere in-between evaluating without any knowledge of the stimulus and with the exact knowledge of how the gaze targets were moving – the stimulus material is required for analysis, but no coding of moving objects in it is needed.

The authors proposed first identifying up to six moving objects in a video (via feature point detection and tracking, already implemented in computer vision libraries, followed by clustering according to their speed vectors). The periods when gaze point is in motion are then matched to the objects moving with the most similar speed vector. If the matched gaze and object were close in spatial coordinates as well, this part of the signal is marked as pursuit according to what the authors call the "video-gaze model". For an evaluated smooth pursuit detector, the percentages of smooth pursuit it labels *in agreement* and *in disagreement* with the video-gaze model were computed as quality measures.

While this does provide a way to express the performance of a model in a numeric way, it effectively compares the tested model against another, independent model – the proposed video-gaze model, the predictions of which are used in place of the ground truth. Additionally, the video gaze model itself involves a substantial number of parameters and thresholds (for determining the "moving" gaze and feature points, tracking, clustering, as well as determining the alignment between the gaze and the video motion).

Application-based evaluation

Application-based evaluation is indirect, "third party" evaluation where the algorithms' performance is assessed or compared using an application that depends on eye movement event detection. For example, a better detector is likely result in a better interaction experience and faster task completion times in human-computer interfaces similar to the one presented by e.g. Schenk et al. (2017), because such interaction techniques directly dependent on the eye movement detection. Earlier/better saccade detection could also lead to a better saccade landing position estimation in foveated rendering systems like the one developed by Arabadzhiyska et al. (2017). Zemblys et al. (2018) hypothesized that it is reasonable to expect that more precise eye movement detection should result in better performance of eye movement-based biometric system, since it is based on fixation and saccade features. Authors demonstrated that their proposed event detection algorithm improves equal error rate of the Rigas et al. (2016) biometric system by between 1% and 6.2%, compared to a baseline algorithm by Nyström and Holmqvist (2010).

Application-based testing is useful when accurate event detection is not of direct interest or, for example, when data recording is disallowed by the license restrictions. It can also be an additional test for a newly developed event detector (as it was used in Zemblys et al. (2018)), although the usability of this approach on its own is very limited.

Detection performance evaluation

All of the approaches described above only provide a limited view of the performance of the event detector and usually lack any insight into how the detection performance could be improved. They can be considered as high-level performance evaluation techniques that show the influence of the eye movement detector on the final application that utilizes it or on the subsequently computed eye movement measures. Nevertheless, these approaches operate without explicitly comparing event predictions to the ground truth or can infer an approximation of the ground truth from the stimuli.

Further in the section, we survey the methods that can be utilized when lower-level, more detailed information about the performance of the detector itself is desirable. These methods can provide insights into *what kinds* of errors the detector is prone to: for example, whether it misclassifies a large portion of smooth pursuits as fixations or tends to label saccade onsets and offsets incorrectly. These methods strictly require ground truth, but enable precise evaluation and can point out *where* the errors occur.

Sample-level evaluation

Comparing labeling performance on the sample level is straightforward, as there is always a one-to-one correspondence between the predicted and the ground truth sample labels and, therefore, any classification performance evaluation metric (see "Confusion matrix-based measures") or other prediction quality measures can be directly computed.

However, since the eye tracking data are usually imbalanced – the majority of samples belong to fixations, smooth pursuits, or other slow eye movement types (cf. Table 1) – it may be challenging to achieve a reliable evaluation. As a consequence, sample-level performance estimates can be unreliable and over-optimistic. Take, for example, two different predicted sequences illustrated in Fig. 2. On the sample level, the situations depicted on the left and on the right are identical: Both predicted sequences have 80% of the samples labeled correctly, and the remaining 20% of the ground truth fixation samples are mislabeled as saccades. Both qualitatively, and when considering quantitative evaluation on the level of complete events, however, the predictions are considerably different: The left prediction sequence has one seemingly correctly detected fixation (GT1→P1) and one falsely detected saccade (P2). The predicted sequence on the right instead consists of three separate fixations (P3, P5 and P7) and three falsely detected saccades (P4, P6 and P8). It is intuitively obvious that detections illustrated on the right of Fig. 2 are worse (or at least differently bad, compared to the ones on the left), yet sample-level evaluation does not in any way quantify this, and any sample-level score would indicate that the two predicted label sequences are identically similar to the ground truth.

Event-level evaluation

Compared to the evaluation on the level of individual gaze samples, event-level evaluation is considerably more involved and ambiguous. With *events* defined as uninterrupted sequences of samples with the same label, there is no one obvious way how to establish correspondences between the events in the ground truth and in the predictions. Subsequent evaluation is not necessarily straightforward either, with many possibilities of quantifying the agreement. We call the procedure of obtaining event correspondences *event matching*, and provide a high-level overview of the aspects of this process below. "Event matching methods" contains detailed descriptions of the different approaches for event matching developed in the eye tracking literature to date. It should be noted that, on the whole, event-level evaluation is slower and more computationally intensive than its sample-level counterpart, especially when event matching

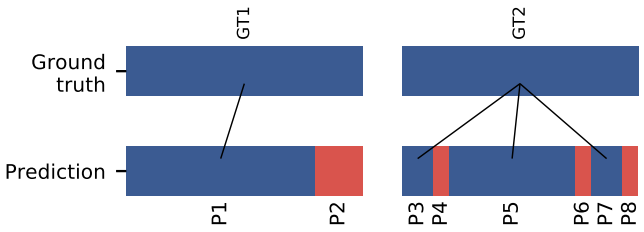

Fig. 2 Example of limitations of sample-level performance evaluation. Blue are fixations, red – saccades. Black lines between the ground truth and predicted events illustrate possible event matching

is involved. Researchers have also developed a number of event-level evaluation methods that do not require explicit matching of ground truth and predicted events, though these typically rely on distributions and other statistics of event properties (see "Evaluation based on eye movement metrics").

Event matching

Event matching is a procedure that establishes a mapping between the eye movement events in the ground truth and the predicted event sequence. The procedure for producing an intuitive and useful matching (in the sense of enabling sound and descriptive evaluation) can be rather complicated and potentially computationally expensive, as the events in the two sequences rarely align perfectly; might be fragmented or merged; there might be multiple viable match candidates for a single ground truth event; or a corresponding event might not even exist. It is, however, a worthwhile endeavour, as matching enables identifying what kind of errors the algorithm makes, and where exactly in the data they occur.

In our definition, producing a matching is equivalent establishing a mapping $\mathcal{M}$ between two event sets (GT and P – ground truth and predictions), which can be represented as a set of correspondences between event sequence subsets, i.e.

$$\mathcal{M} = \{\langle gt_i \subseteq GT, p_i \subseteq P\rangle, i = 1 \dots |\mathcal{M}|\}, \quad (1)$$

where $|\mathcal{M}|$ is the number of correspondences in $\mathcal{M}$ and gt and p – corresponding ground truth and predicted events respectively. Note that both gt and p are event *sets* – subsets of the respective complete sets GT an P. This is a very loose definition that does not restrict all the possible matches, thus enabling entirely arbitrary mapping, including matching to an empty set $\emptyset$, matching non-overlapping events, one-to-one, one-to-many, and many-to-many matching.

Matching methods (see "Event matching methods") either implicitly or explicitly impose certain restrictions on the matches they can possibly produce, with the purpose of making them more intuitive, thus enabling sensible event detector performance evaluation. Example restrictions include such rules as: matched events need to either overlap or be in very close proximity; matched events need to be of the same class; only one-to-one matches are allowed, etc. Naturally, the restrictions internally imposed by a matching algorithm have a bearing on which events are matched to which, if to any at all, thus in turn affecting the applicability of evaluation metrics, as well as the resulting scores of the evaluation pipeline as a whole (cf. "Interaction between the performance metrics and event matchers"). For instance, a one-to-one matching will not enable quantifying event fragmentation directly, while enforcing same-class-only event matches will in most cases yield a different evaluation score, compared to allowing matches between events of any classes.

An important concept of event matching is *unmatched events*, or, in our formalization – events matched to an empty set $\emptyset$ rather than a corresponding event or events. Usually it is the evaluation procedure itself that defines how to account for unmatched events. For example, evaluation that consists of comparing the properties of the corresponding ground truth and predicted events (see "Evaluation based on eye movement metrics") only considers same-class matches and ignores the rest, together with the unmatched events. Classification performance metrics (see "Evaluation metrics"), on the other hand, can account for various combinations of classes in each match, including the unmatched events: When calculating the multiclass performance scores, the events that remain unmatched, if any, can be treated as either "missed" (an unmatched event in the ground truth) or "false alarm" (an unmatched event in the predicted labels), and thus contribute to the score. Therefore, it is important to remember that not all event matchers report unmatched events, effectively making even the same evaluation metrics fundamentally incomparable when obtained using different matching techniques (see "Interaction between the performance metrics and event matchers").

Matching symmetry Typically, direct event-level evaluation relies on a fixed ground truth (e.g. expert annotations) to which the predicted labels of all the tested algorithms are compared, i.e. event matching direction is $GT \rightarrow P$. However, in the case of comparing two expert coders or two algorithms between themselves, event matching symmetry becomes important both for evaluation consistency and comparability. Different matching techniques may yield different matches and, consequently, different evaluation scores, depending on what is compared to what. Therefore, it is very important to report the direction of the applied event matching. One alternative would be to run the matching (and the subsequent evaluation) in both directions and average the resulting scores or measures, thus eliminating potential matching direction bias (e.g.

Kothari et al., 2020). While guaranteeing the evaluation symmetry, this approach is naturally slower and somewhat more cumbersome. Therefore, all other aspects being equal, it would be preferable to achieve the symmetry of evaluation via the guaranteed matching symmetry directly.

One-to-many and many-to-one matching When talking about matching in general, the "*one-to-many*" characterization refers to the ability of registering matches between single entities and entity groups (in our case – eye movement events and their groups). This is in contrast both to *one-to-one* matching, where only matches between single events or single events and $\emptyset$ are allowed, and to *many-to-many*, where matching between arbitrary event groups is possible. To be able to better describe the approaches in the eye movement event detection literature, we need to additionally differentiate between matchers capable of establishing *"one-to-many"* and *"many-to-one"* correspondences, the left side of the adjective referring to the ground truth events, and the right side – to the predicted events. This means that if an event matcher is characterized as "one-to-many", it can match a single ground truth event to multiple detected events (i.e. it can potentially correctly handle fragmentation cases). Similarly, a "many-to-one" characterization is applied to a matcher that can potentially handle cases where multiple ground truth events are merged into a single detected event by matching these to one another. A matcher that can do both we will then describe as "one-to-many and many-to-one". Naturally, if an event matcher is symmetric, it has to belong to this latter variety.

Multiclass vs. binary evaluation

Many of the annotated eye movement datasets are inherently multiclass, as more than two event types are annotated (cf. Table 1 and "Cross-dataset evaluation"). Many existing algorithms also detect more than one eye movement class. In this situation, the performance of an algorithm can be evaluated using an *overall* score, i.e. a score that reflects its performance at correctly detecting *all* event types that this algorithm was designed to detect (or all event types in the ground truth). We refer to such evaluation scores as the *multiclass* performance scores. Contrasted with these are *binary* performance scores that evaluate the algorithm's ability to correctly detect a single event class separately. Binary scores can be computed for every annotated event type, thus providing a more descriptive and easily interpretable, though less concise evaluation of algorithm's performance. To obtain a concise statistic, binary scores can then be averaged and represented as overall performance as a single score.

Concise multiclass sample- or event-level scores (further discussed in "Confusion matrix-based measures", especially in the context of Table 3) are useful for brevity, e.g. when comparing different algorithms that all detect the same set of event types, or different versions of the same algorithm (for example when training a machine learning-based algorithm, or adjusting thresholds for a traditional event detector). Nonetheless, it is not uncommon to also report classification performance scores for each of the event types separately (Agtzidis et al., 2019, Table 3; Startsev et al., 2019a, Table 2; Zemblys et al., 2019b, Tables 7 and 8). Event-level detection quality metrics (expanded on in "Event quality metrics"), on the other hand, usually only make sense in a binary setting, as they are calculated for each of the event types separately (e.g. the alignment between the ground truth fixation and corresponding falsely detected event is usually not of interest).

A noticeable advantage of binary evaluation is that it enables relatively fair comparison of algorithms that do not necessarily detect the same set of events, allowing to focus on the detection performance for the classes of interest, or those that the compared algorithms all have in common. It is important to note that multiclass evaluation using multiclass classification performance measures (accuracy, κ, MCC) will result in different scores compared to averaging the same scores of per-class binary evaluation.

There are multiple ways to perform binary evaluation in the presence of several event classes. Having denoted the "event class of interest" as E, the simplest way forward would be to ignore all non-E events altogether. This would correspond to e.g. directly applying the evaluation procedure of Hauperich et al. (2020) in the presence of multiple-class events, although it was originally developed for a single-class setting. This, however, does not distinguish between missing and misclassifying the event of type E, and thus does not lend itself to a descriptive evaluation pipeline on the whole.

If the non-E events are not to be ignored completely, they could be simply registered as belonging to the "negative class", while events of type E can be considered as positive-class entities. This process is called *binary remapping*, and its application allows for e.g. any of the

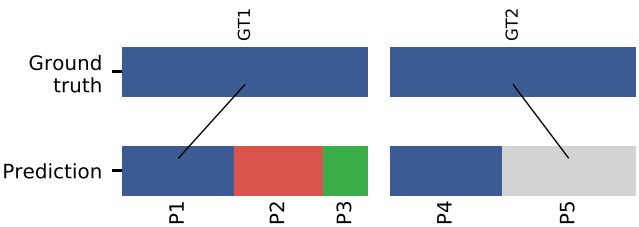

Fig. 3 Example of multiclass (on the left) and binary evaluation using sample-level binary remapping (on the right). Blue are fixations, red – saccades, green – PSO, gray – negative class in binary evaluation. Black lines connect the ground truth with the predicted events matched to it according to the largest overlap criterion

detection performance evaluation methods (see "Confusion matrix-based measures") to be applied and per-class scores to be calculated. The remapping can be performed either on the level of samples or events: With *sample-level* binary remapping (i.e. performed *before* the uninterrupted same-class sequences of samples are grouped into events of the respective class), a sequence of two separate non-E events becomes a single longer negative-class event. With *event-level* binary remapping (i.e. *after* the samples are grouped into events already), the two subsequent non-E events would stay as two separate but now explicitly negative-class events (e.g. for fixation as class E, a sequence of a saccade and a PSO would be mapped to a sequence of a two negative class events, i.e. a non-fixation and a non-fixation). Both types of binary re-mapping have been used in the literature already, e.g. Zemblys et al. (2019b) used the sample-level remapping procedure, while Startsev et al. (2019a) and Startsev et al. (2019) opted for its event-level alternative.

Interaction with event matching When performed *prior* to event matching in the event-level evaluation, binary remapping can and usually will affect the matching outcome. Figure 3 illustrates a simple example of sample-level binary remapping affecting a hypothetical event matching approach (based on largest overlap between the events, similar to Maximum overlap in "Maximum overlap matching"). In the multiclass case (in the figure on the left), the ground truth fixation GT1 is matched to a predicted fixation P1 since their overlap is the largest. However, after sample-level binary remapping (in the figure on the right), event P5 (the result of grouping two sets of negative-class samples – corresponding to P2 and P3) has the largest overlap with the fixation GT2. Therefore, the fixation GT2 can no longer be considered as correctly detected, and a false negative error will be registered instead.

Descriptiveness A downside of a binary evaluation strategy in terms of producing insightful statistics is that it is not always obvious how to produce descriptive performance summaries of the detection pattern, e.g. that a fixation is not just "misclassified", but confused with a smooth pursuit, or, in a more complicated example – with a sequence of several events, like smooth pursuit, a small saccade, and a PSO. The latter situation would also present a challenge for multiclass evaluation, as it requires one-to-many event matching, which is rarely seen in the literature, but it is at least theoretically possible to produce observations that would be directly useful and interpretable for the algorithm developer in this case as well. Maintaining this information throughout the binary evaluation (e.g. that the predicted negative-class event P5 in Fig. 3 consisted of respective events P2 and P3 prior to the remapping) would alleviate this problem, but would require a careful overhaul of the existing pipelines. As it stands right now, multiclass event matching and evaluation seems to provide the most descriptive option when considering one algorithm, while binary evaluation (with sample-level binary remapping before event matching) provides the fairest way of comparing multiple detectors, especially when they do not detect the same set of eye movement events.

Negative-class events Furthermore, event-level binary evaluation requires additional consideration of the unmatched negative events. While unmatched positives are treated as "missed" or "false alarms", unmatched negatives can be considered as errors, as correct predictions or ignored. Recent works (Zemblys et al., 2020; Startsev et al., 2019) argue that penalizing the algorithm score for incorrectly detecting events other than the one under evaluation does not provide a desirable view of the algorithm's performance. On the other hand, Friedman (2020) argues that approach originally used in Zemblys et al. (2019b) where the authors counted unmatched negative events as true negatives unfairly inflates the performance score. Zemblys et al. (2020, Table 2) analyzed all three approaches, demonstrating that disregarding unmatched negative-class events(i.e. belonging to the class other than the one being evaluated) results in a Cohen's kappa score that best reflected the algorithms' performance in terms of detecting only the event type of interest.

Handling undefined events

As mentioned before in "Data sources", it is not required that all the samples in the data have an event label. For example, parts of the data might be too noisy to reliably annotate events, or the study requires only one type of event to be annotated (Hooge et al., 2018; Steil et al., 2018). In this case, all the unannotated samples form *undefined* events. These, especially those in the ground truth, are typically not of interest when assessing the performance of event detection algorithms. Therefore, special care needs to be taken to avoid the artifacts that undefined events can introduce into the evaluation process and results.

It is also very common to evaluate algorithms using data with a different set of event types annotated compared to what the algorithm is designed to detect (Andersson et al., 2017; Zemblys et al., 2019b). Hence, events of types that are not *both* annotated in the ground truth *and* detected by the algorithm need to be either ignored or explicitly converted to undefined before performing the evaluation. For example, to develop their algorithm Zemblys et al. (2019b) used only part of the Lund2013 dataset (Andersson et al., 2017) where two independent coders identified samples being

either fixation, saccade, or PSO. When comparing different algorithms, authors then relabelled output samples that do not belong to the three evaluated classes to *undefined*. Depending on the differences in eye movement definitions, conversion between classes may be necessary as well: E.g. to evaluate an algorithm that detects both saccades and PSOs on a dataset where the definition of saccades includes PSO into this event type (e.g. GazeCom dataset from Startsev et al., 2019b), re-mapping predicted PSO samples to "saccade" rather than treating them as undefined events would align the saccade definitions in the annotations and the algorithm's output.

While some event matchers described below in "Event matching methods" ignore the sequences of undefined class labels altogether, others treat undefined events just like any other event class: That is, they allow for the undefined events to be matched with any other event(s), including undefined-to-undefined matches. The later are however uninformative and simply increase the performance score without describing the algorithm's performance any better. Take, for instance, a fixation detector tested on a dataset of annotated fixations (e.g. Hooge et al., 2018). Any matches found in the unlabelled part of the data are irrelevant (though correlating in this particular example) with regards to the detection quality that one might want to measure.

When considering the event-level evaluation pipeline as a combination of two parts – "event matching" and "metric computation" – that can be chosen freely, it is a responsibility of the researchers to ensure a meaningful combination of those steps, especially concerning the undefined events. For instance, a matcher that treats undefined events as any other class should not be combined with either a multiclass evaluation metric (that also equates all classes) or a binary metric that rewards true negatives: Both of these might be easily inflated by an abundance of undefined-to-undefined matches.

Evaluation metrics

Evaluation metrics listed in this section require ground truth annotations and the correspondence between the ground truth and the prediction sequences established. On the sample level this correspondence is inherently present, while event-level evaluation requires event matching ("Event matching methods"). Once that is done, the detection power of the prediction algorithm can be quantified using various metrics. For example, Hooge et al. (2018, Fig. 9) and Kothari et al. (2020, Table 7) provide event timing evaluation, Hooge et al. (2018, Table 4) and Startsev et al. (2019a, Tables 1 and 2) report event-level F1-scores, Zemblys et al. (2019b, Tables 7 and 8) and Startsev et al. (2019, Table 2) use event-level Cohen's kappa, etc. All evaluation methods below are discussed in the context of the properties that a reliable and intuitive event-level evaluation method should possess: It should be able to identify all the errors (together with their types) that the algorithm makes and evaluate the algorithm's output in a consistent and easily interpretable way.

Event quality metrics

Event matching enables the evaluation of the *quality* of the predicted events, i.e. the assessment of how similar, in terms of their properties, the ground truth and the predicted events are. We note that usually only the matched events of the same class are analysed for event quality – mistakes related to mislabelling the events are handled via other metrics (see "Levenshtein distance" and "Confusion matrix-based measures").

Event timing

Event timing is the most often evaluated event quality metric: Even if the algorithm is capable of producing event sequence similar to that in the ground truth, onset, offset, and duration of the test events might differ. Hooge et al. (2018) proposed using the *Relative Timing Offset* (RTO) and the *Relative Timing Deviation* (RTD) to capture these subtle timing differences. RTO (equivalent to bias in Bland-Altman plot, see below) is the difference between the on-/offset of the ground truth and the on-/offset of the predicted events, respectively, while RTD is the variance of RTO. Authors argue that these measures are "the missing links between agreement measures such as the F1 score and the eye movement parameters" (Hooge et al., 2018, p. 1879). A similar metric was used by Kothari et al. (2020), but instead of providing differences in onsets and offsets separately, authors calculated timing offset as l_2 *distance* between the on- and offset timestamp pairs (Eq. 2):

$$l_2 = \sqrt{(GT_{\text{onset}} - P_{\text{onset}})^2 + (GT_{\text{offset}} - P_{\text{offset}})^2}, \tag{2}$$

where GT stands for an event in the ground truth, and P – for the correspondingly matched predicted event.

The absolute timing offsets have different implications for the events of different durations: E.g. the misalignment of a few samples is probably not very important in fixation detection, but might mean that half of a saccade is misaligned. To evaluate how well the ground truth and predicted events align in time, Startsev et al. (2019a) and Kothari et al. (2020) used event *Intersection-over-Union ratio (IoU)* – a measure that takes the duration of the two matched events into account and indicates the proportion of their overlap with respect to the duration of

the union of the two events. IoU has a useful property of ranging between 0 and 1 (where 1 means perfect event timing agreement and 0 – no overlap), contributing to its general-purpose ease of interpretation. On the other hand, in the situation of close familiarity with the data set, the algorithm developer might prefer absolute timing differences since those are expressed in the same units as event durations.

It is worth noting that all of the timing measures are easiest to formalize under the assumption of exclusively one-to-one matches being allowed, and indeed we have formulated them in this setting here. Nevertheless, all of the measures above can be applied with a one-to-many matcher: E.g. if a ground truth event if fragmented in the algorithm predictions, and the matcher has accounted for it, one could use the first and last events of the fragmented sequence to determine the on- and offset, respectively, of the "predicted event" e.g. in RTO of l_2 definitions. Indeed, the matcher used by Kothari et al. (2020) allows for one-to-many correspondences, and the authors have computed all the event quality measures in this setting.

Bland-Altman analysis

Analysis of timing offsets as well as any other event property (duration, amplitude, velocity, etc.) can be performed with Bland-Altman plot (Bland & Altman, 1986) – a method for analyzing the agreement between two quantitative measurements of the same variable. It was used by e.g. Swan et al. (2020, Figs. 9 and 10) when comparing the output of their algorithm detecting certain patterns in the gaze signal ("gaze scans") to the ground truth. Bland-Altman plot – the difference of the two paired measurements plotted against the mean of these (referred to as "magnitude" by the authors) – enables visualization and statistical analysis of the differences between the two measurement systems. In addition such representation allows to investigate the relationship between the measurement error and the magnitude of the measurement, e.g. if larger timing errors are associated with longer fixations, etc. Note that even though the ground truth measurement value is available, the magnitude is estimated as the average of the two measurements. Plotting the difference against the ground truth will always show a relation when there is none (Bland & Altman, 1995). Average difference is called *bias* and shows how much, on average, the two systems (e.g. the ground truth and predictions) differ. The *limits of agreement* show an agreement interval, within which 95% of the differences between the two measurement methods fall.

For an in-depth review of Bland-Altman analysis refer to (Giavarina, 2015). Here we only note that such analysis is very flexible, not being limited to a fixed set of compared measures. Such plots can be produced for any conceivable eye movement-related measure that is of direct interest for the undertaken study (average estimated pursuit speed, fixation stability, etc.), as long as it can be computed based on the results of event matching.

Unlike computing the average of these measures across all events in the ground truth or predictions, or even comparing the distributions of these values (cf. "Evaluation using similarity to the ground truth event parameters"), Bland-Altman plots allow directly assessing the differences between the *corresponding* events, not the overall distributions. Special care has to be taken with respect to undetected or falsely detected events, however, as those are not represented in the plot, as there are no two "measurements" that can be compared.

Levenshtein distance

Levenshtein distance is a metric originally used for measuring the differences between two sequences of characters. It is also referred to as "edit distance" in the literature, as it corresponds to the minimal number of "edits" (single character insertions, substitutions, or deletions) required to match the two compared sequences. In application to eye movements, the labels of the ground truth samples (or events) and the corresponding-level output of the detector can be easily re-interpreted as sequences of "characters" denoting event type, enabling the ready application of this metric.

Length-normalized Levenshtein distance is often used to evaluate speech recognition systems (Amodei et al., 2016; Dernoncourt et al., 2018; Chiu et al., 2018). It is called Character Error Rate (CER) or Word Error Rate (WER) depending whether the evaluation is performed on the level of individual characters or complete words:

$$CER = \frac{(i_c + s_c + d_c)}{n_c}; WER = \frac{(i_w + s_w + d_w)}{n_w}, \quad (3)$$

where i is number of insertions, s – number of substitutions, d – number of deletions, and n – number of characters or words in the ground truth sequence. Subscript indices c and w refer to characters and words, respectively.

Sample Error Rate In the eye movement literature, Zemblys et al. (2019b, Table 6) used a length-normalized sample-level Levenshtein distance (denoted *Sample Error Rate* by the authors) as one of the evaluation metrics for their recurrent neural network-based event detection algorithm. It was also tested in Startsev et al. (2019) among a number of other detector quality metrics and demonstrated a fair ability to separate the "baselines" from dedicated eye movement

detection algorithms, even though the different in their scores was not large.

This metric is relatively easy to interpret and, if normalized for sequence length, it ranges from 0 to 1, where 0 is the best achievable score. In some cases, however, Levenshtein distance can be a very poor quantification for eye movement detection performance: For example, imagine the case of alternating saccades and fixations, where a detector misclassifies the first sample of every ground truth saccade as a fixation and extends every saccade by one sample. The actual number of errors made by the algorithm would be twice the number of saccades in the dataset. By contrast, the number of *edits* required to match such ground truth and predicted event sequences will only be 2 – one deletion at the beginning and one insertion at the end of the test sequence.

This metric is also relatively difficult to compute (compared to other sample-level measures), requiring the number of operations proportional to the square of the evaluated sample sequence length. In contrast, all of the confusion matrix-based metrics are computed in linear time. This becomes especially noticeable with prolonged eye tracking recordings and high sampling frequencies. For a thorough description on calculating Levenshtein distance, see Manning et al. (2010, Chapter 3.3.3, p. 58).

Event Error Rate is the event-level Levenshtein distance normalized by the length of the ground truth event sequence that provides information on how well the event detector preserves the ground truth event sequence[5] (Zemblys et al., 2019b, p. 846). The benefit of computing Event Error Rate lies in its convenience: While being an event-level metric, it does not require explicit event matching, and can be computed directly for the pairs of event sequences, without any preliminary steps. The matching is implicitly computed during the computation of the distance, however, though there are no restrictions on this matching (e.g. matched events do not need to overlap or even be close to one another in time).

As highlighted in (Startsev et al., 2019), event-level Levenshtein distance is susceptible to potentially "confusing" e.g. randomly vs. meaningfully labelled eye tracking sequences: For instance, a plausible yet random sequence of eye movement events produced a *lower* Event Error Rate than six out of seven tested algorithms from recent literature. Concerns that Levenshtein distance does not evaluate similarity well enough are also expressed in language processing community (e.g. Greenhill, 2011).

In addition, Event Error Rate does not necessarily range between 0 and 1. In case the predicted event sequence is longer than the ground truth event sequence, Event Error Rate could exceed 1. Therefore, if evaluation requires measures to have a certain range, for example when combining multiple different measures into one *meta*-measure (e.g. $RMSD$ in Andersson et al. (2017, Eq. 2) or S in Zemblys et al. (2019b, Eq. 6)), one would need to normalize the Levenshtein distance by the length of the longer of the two compared sequences (Startsev et al., 2019), or clip the resulting values at 1.

[5]It is widely used in speech recognition systems under the name of *Word Error Rate*

Table 2 Example of a confusion matrix C for N classes. The C_{ii} elements on the diagonal quantify correctly predicted labels, the rest – $C_{ij,i\neq j}$ refer to samples mislabelled in a particular way (label j assigned to an example of class i). Notations in brackets denote terms used in binary classification: TP for True Positives, TN – True Negatives, FP – False Positives, and FN – False Negatives

		Predicted			
		Class 1 (Positive)	Class 2 (Negative)	...	Class N
Ground truth	Class 1 (Positive)	C_{11} (TP)	C_{12} (FN)	...	C_{1N}
	Class 2 (Negative)	C_{21} (FP)	C_{22} (TN)	...	C_{2N}
	...	...	...	...	...
	Class N	C_{N1}	C_{N2}	...	C_{NN}

Confusion matrix-based measures

The most basic, yet often the most informative method to describe the performance of an algorithm is a table containing the numbers of correctly and incorrectly classified samples or events – a *confusion matrix* (Table 2).

In the case of binary evaluation, one of the classes (Class 1 in Table 2) is referred to as a positive class, and the other other one (Class 2) – as negative class. The values on the diagonal of a binary confusion matrix are the amount of *True Positives (TP)* and *True Negatives (TN)*, corresponding to correctly labelled samples or events of the positive and negative class, respectively. Entities incorrectly labelled as the positive class (C_{21} in Table 2) are counted under the name of *False Positives (FP)*, and those incorrectly labelled as the negative class – under the name of *False negatives (FN)* (C_{22} in Table 2). For example, when the event detection algorithm is designed to only detect fixations (Class 1, positive) while the rest of the samples are labeled as saccades – Class 2 or negative – false negatives are fixations in the ground truth that received the labels of saccades from an algorithm, and vice versa for the false positives.

The numbers of correctly and incorrectly classified entities are usually normalized for better interpretability. Most often a confusion matrix is normalized along its rows, i.e. each row is divided by its sum to express the proportions of correctly and incorrectly classified ground truth entities of each of the eye movement classes (e.g. Hoppe & Bulling, 2016, Fig. 7; Houpt et al., 2018, Tables 2 and 4; Zemblys et al., 2019b, Fig. 8). Such normalization allows for quick assessment of both how accurate the algorithm is at correctly detecting every events class (sensitivity, see below; values on the diagonal), and what what misclassification errors it makes.

While the metrics that can be derived from the confusion matrix attempt to summarize the matrix by one or several statistics, reporting the full table enables precise attribution of the misclassified examples: Zemblys et al. (2019b, Figure 8) report that compared to two expert coders, their algorithm misclassifies around 18% and 16% of PSO samples as fixations; Hoppe and Bulling (2016, Figure 7) show that their algorithm has a tendency to label ground truth saccade and smooth pursuit samples as fixations; Zemblys et al. (2019b, Fig. 9) also provide an event-level confusion matrix that enabled them a more in-depth analysis of potential causes of the detection errors. Given such information, algorithm developers can focus more on the algorithms' bias towards certain directions of mislabeling (e.g. other events as fixations, etc.) and thus possibly improve the performance in a targeted way.

Despite of confusion matrix providing a lot of information about the labelling behavior of the event detection algorithm, it is not a performance metric by itself, containing N^2 values. Therefore, it is often desirable to express the algorithm's performance more concisely. The performance measures that we discuss in this section are derived from the confusion matrix and are used by researchers to evaluate and compare eye movement event detection algorithms.

Accuracy

Accuracy is the ratio between the number of correctly classified samples or events and the total number of entities. It can be calculated and reported separately for each class (Eq. 4 below), with all events of "other" classes treated as negatives, and only the events of the considered class – as positives:

$$Accuracy_{\text{binary}} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4}$$

In the case of binary classification it is, of course, only computed once, since it would be the same for either class. For the multiclass case, the binary accuracy for one class effectively ignores misclassification errors between all "other" event types.

Accuracy can also be directly computed for all classes together, thus reporting the overall performance of the classification algorithm:

$$Accuracy_{\text{multiclass}} = \frac{\sum_{i=1}^{N} C_{ii}}{\sum_{i,j=1}^{N} C_{ij}}, \tag{5}$$

where C is the confusion matrix for N classes (see Table 2).

Accuracy is an easy to understand and calculate metric and, therefore, widely used when reporting performance of eye movement detectores (see Anantrasirichai et al., 2016, Tables 1 and 2; Hoppe and Bulling, 2016, Table 2; Santini et al., 2016, Table 3; Peng et al., 2019, Table 3 for a few examples). A closely related metric – misclassification rate (or error rate; equivalent to $1 - Accuracy$) – was used by Andersson et al. (2017, Table 8, referred to as "ratio"), Startsev et al. (2019a, Table 7), Dar et al. (2020, Table 2), to name a few.

However, accuracy in these definitions represents a poor metric for evaluating the performance in unbalanced datasets (where certain classes are prevalent or underrepresented) because of the *accuracy paradox*[6]: A predictive model with high accuracy might in fact have a low predictive power, i.e. although having high absolute performance scores, the output of the model would not be much different from a simple baseline calculated from event base rates. The majority of gaze samples in eye tracking data are typically attributed to fixations or other slow eye movements like smooth pursuit (see Table 1). Therefore, if e.g. an algorithm predicts all the samples as belonging to the majority class, its sample-level accuracy will be high, but such an algorithm would be useless in practice. On the event level the issue is not so prominent, yet can be problematic when evaluating data containing relatively rare events such as blinks or PSOs. A solution to account for data imbalance is to calculate a variant of the accuracy measure – *weighted accuracy*:

$$Accuracy_{\text{weighted}} = \sum_{i=1}^{N} w_i \frac{C_{ii}}{\sum_{j=1}^{N} C_{ij}}, \tag{6}$$

where w_i is the weight for the i^{th} event class and C is the confusion matrix for N classes.

In order to preserve the range and with it the intuitive interpretation for good – close to 1.0 – and bad – closer to zero – scores, weights with the following properties are used: $w_i \geq 0, \sum_{i=1}^{N} w_i = 1$. The intuition behind the weighting is to strengthen the influence of the less frequent

[6] See https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b for an in-depth explanation.

classes, which would have only a limited impact on the non-weighted accuracy: For instance, a classifier labelling everything blindly as fixations would have a 90% accuracy in a dataset of 90% fixation and 10% saccade samples, even though one class is completely ignored (see "All majority" corner-case test in Table 6, Appendix D).

A special case when all weights are equal to $\frac{1}{N}$ is called *balanced accuracy* and was used by e.g. Larsson et al. (2016, Table 8) when evaluating and comparing their head movement-aware event detector to other algorithms. Balanced accuracy ensures that if the classifier takes advantage of the majority class, its score will be as low as $\frac{1}{N}$. In a binary case, balanced accuracy is equal to the arithmetic mean of sensitivity and specificity. When employed to evaluate event-level performance, however, balanced accuracy can result in either over- or under-estimated score, thus potentially misleading the algorithm user. See Appendix F and Fig. 22 for a more detailed explanation.

Other options to calculate weighted accuracy include using weights either directly or inversely proportional to the *support* (also known as *class probability* p, i.e. the proportion of samples or events in each of the event classes). Proportional weighting however makes the correct prediction of the majority class even more important, which might be desirable in some cases. To counteract this, one can weigh proportionately $1 - p$ instead, thus making the minority class more important. In the example above, the weights for fixations and saccades would be set to 0.1 and 0.9, respectively. If one of the event types has very few occurrences, this minority class will be heavily "upweighted" in the score calculation, and the algorithm predicting every sample as belonging to the minority class will score unreasonably high.

Another approach to deal with the accuracy paradox was used by Goltz et al. (2019), where the authors used this metric in combination with *bootstrapping*: They computed the accuracy of the algorithm's predictions on 20 different random subsets of the data that were sampled in such a way as to contain an equal number of each event labels. Similar to balanced accuracy, bootstrapping ensures that a poor performing algorithm, such as majority or minority class predictor, does not score better than $\frac{1}{N}$.

On the whole, accuracy should be avoided when evaluating eye tracking data. Weighted accuracy is also not a perfect solution, as it heavily depends on the selected weights, making it difficult to compare across studies, in addition to not fully solving the accuracy paradox. Bootstrapping, on the other hand, ensures that the testing data themselves are balanced, though it can only be applied on the sample level: There is no universally sensible way of sampling only certain events for evaluation without potentially heavily affecting the outcome of event matching, thus skewing the results in each subsampled set. Moreover, bootstrapping makes accuracy calculation both parameter-dependent (number of samples, sampling with or without replacement) *and* randomness-dependent. The latter means e.g. that the researchers would need access to the exact code and random state (or the exact random subsets that resulted from these) that produced the evaluation results in order to exactly reproduce these, which may be difficult to conveniently achieve across platforms and programming languages.

Binary metrics

We will now describe a set of evaluation metrics that are based on the confusion matrix analysis and can be only computed in a binary setting: Precision, sensitivity (or recall), specificity, and F1-score. To compute these, a certain "event type of interest" – for example fixation – needs to be considered as the "positive" class, while the rest of the event types (e.g. saccade, PSO, etc.) are considered as belonging to the "negative" class.

In case an event detection algorithm is designed to detect only one event type, it is enough to report one precision, sensitivity, specificity or F1-score score. For example in Anantrasirichai et al. (2016, Tables 1 and 2) authors report precision and sensitivity (recall) only for fixations when evaluating their fixation detection algorithm. However, if the algorithm detects more than one event type, such binary-setting scores need to be calculated for each of the events separately (cf. also "Multiclass vs. binary evaluation"), and, optionally, averaged to report the overall performance (see for example Hoppe and Bulling, 2016, Table 2 or Larsson et al., 2016, Table 8).

Precision reveals the proportion of predicted entities (samples or events) of the positive class that belong to the positive class in the ground truth as well:

$$Precision = \frac{TP}{TP + FP}. \tag{7}$$

For instance, when evaluating fixation detection, sample-level precision would be computed as the share of samples with the predicted fixation label that are attributed to the fixation class in the ground truth. It can be understood as the metric of the reliability of a predicted positive-class label.

Sensitivity (also known as *recall* or *True Positive Rate*) measures the proportion of the ground truth entities of the positive class that are correctly predicted as that class. In other words, sensitivity describes how successful the algorithm is at not missing the entities of a particular type

of eye movement type – the positive class:

$$Sensitivity = \frac{TP}{TP + FN}. \quad (8)$$

Specificity (also known as *True Negative Rate*) is a measure complementary to sensitivity, and can be interpreted as the sensitivity for the negative class. It is computed as the proportion of the negative-class ground truth entities that are correctly identified as such by the algorithm:

$$Specificity = \frac{TN}{TN + FP}. \quad (9)$$

F1-score is defined as a harmonic mean (mean with the emphasis on the lowest value) between precision and sensitivity, and is often used as a way to quantitatively assess the algorithm's performance with one metric when no inherent preference for either precision or sensitivity can be inferred. In fact, F1 is just one of a family of metrics (denoted as F_β, with β setting the balance between the importance assigned to precision and sensitivity)[7]. The most commonly used variant (with $\beta = 1$) is formalised as follows:

$$F1 = 2\frac{precision \times sensitivity}{precision + sensitivity} = \frac{2TP}{2TP + FP + FN}. \quad (10)$$

Jaccard index (JI, also known as Jaccard similarity coefficient or intersection-over-union[8]) measures the ratio of entities labeled as the positive class in *both* the ground truth and predictions at the same time vs. the entities labelled as the positive class in at least one of the sequences:

$$JI = \frac{TP}{TP + FP + FN}. \quad (11)$$

JI can also be calculated for any event type pair separately and presented similarly to a confusion matrix (Dar et al., 2020, Fig. 2): E.g. one can compute the ratio of gaze samples labelled as fixation in the ground truth and as saccades by the algorithm *at the same time* vs. the gaze samples labelled *either* as fixation in the ground truth *or* as saccade by the algorithm.

Receiver operating characteristic curve

Receiver operating characteristic (ROC) curve is a graphical representation of the algorithm's performance, where, at various algorithm's settings, True Positive Rate (sensitivity, TPR) is plotted against the False Positive Rate (1−specificity, FPR). For an example of a ROC curve used to compare event detection algorithms see Otero-Millan et al. (2014, Figs. 6B and 9) and Hoppe and Bulling (2016, Fig. 6). Receiver operating characteristic is based on a confusion matrix, however in fact requiring multiple confusion matrices to be calculated, each using different settings for the algorithm (corresponding neatly to different "operating points" on the ROC – i.e. different settings leading to one single confusion matrix that results in one point on the curve).

These settings can be e.g.:

- velocity or dispersion thresholds in traditional event detection algorithms like I-VT or I-DT (Salvucci & Goldberg, 2000);
- a set of multiple thresholds if the algorithm uses more than one;
- a threshold on probability of sample belonging to positive/negative class in machine learning based algorithms, or any other algorithm that provides some kind of confidence value associated with the predicted sample label.

While showing the trade-off between specificity and sensitivity, the ROC curve analysis can, however, be misleading when applied to unbalanced data. Saito and Rehmsmeier (2015) analysed a number graphical analysis tools, including ROC curves, and advocated for using precision-recall curves (PRC) to account for class imbalance. Authors show that PRC is more informative and conclude that it better expresses the classifier's susceptibility to imbalanced datasets.

As it is often desirable to express the performance in a concise way, preferably as one number, any curve-based analysis would need to be in some way summarized. The *area under the curve (AUC)* metric, applied to an ROC or a similar curve (e.g. PRC), describes how well the algorithm performs across all possible settings or thresholds. In practice however it is often impossible to evaluate the algorithm for every possible setting, therefore only a subset is used, and an approximation of the AUC is calculated and reported. If the curve is contained within the $[0; 1] \times [0; 1]$ square (which holds for ROC and PRC, for instance), its AUC ranges from 0 to 1.

For ROC and PRC, the "ideal" curve would have the y coordinate at 1.0 on the whole considered segment (i.e. maintain perfect sensitivity at any specificity, or perfect precision at any recall level). Thus, the AUC of 1 would be achieved by the perfect detector. For ROC, an AUC of 0.5 corresponds to chance-level detector, and values below 0.5 signal that the algorithm is doing the opposite of what it

[7] An intuitive understanding of F_β scores can be gained e.g. from the visualization in https://www.mikulskibartosz.name/f1-score-explained

[8] We reserve the term intersection-over-union, IoU for the measure of similarity between two eye movement *events*, cf. "Event quality metrics", to avoid confusion, and use JI to refer to the classification quality metric.

was designed to do, for example labels most of the fixation samples as saccades and vice versa.

Single operating point ROC In many practical situations, the evaluation is performed for one set of detected eye movement events without a possibility to produce detections at any other operating point of the detector. While many data science frameworks will technically allow computing area-under-the-curve metrics in this case, we strongly discourage this practice, as (1) in the theoretical definition such ROC AUC can be directly expressed as $(sensitivity + specificity)/2$, but (2) in practice it will substantially underestimate[9] the AUC, as well as (3) may be strongly affected by the specific implementation details due to the ambiguity of sorting elements with precisely equal values.

Different settings and comparability The most straightforward case for producing an ROC curve is having a confidence score assigned to each of the entities (samples or events) detected by the algorithm under evaluation. In this case, for any threshold value, one can treat only the entities with confidence *above* this threshold as positive-class predictions. By varying the threshold enough, a full range of performance from "all entities are marked as negative class" (threshold higher than maximal confidence score) to "all entities are marked as positive class" (threshold lower than minimal confidence score) can be covered in a systematic way. This is, however, a relatively rare scenario for eye movement detectors, where one will more likely be able to change certain parameters of the algorithm in order to obtain several performance points. Such process is less straightforward than varying a single confidence threshold, since one typically needs to sample from the multi-dimensional parameter space of the algorithm (which is not a well-defined process by itself), and the resulting operating points will likely not form a curve (see e.g. Fig. 8 in Startsev et al., 2019b). To perform any sort of curve-based analysis, the highest-sensitivity points at any specificity value would need to be selected to form the "encompassing" curve. Additionally, it is not guaranteed that by varying algorithm settings one would obtain sufficient coverage of the full x axis of the ROC plot – from zero to perfect specificity, forcing the partial AUC computation, further complicating inter-publication comparison, since different x axis ranges have might been examined. We also refer the reader to DeLong et al. (1988) for a statistical viewpoint on comparing ROC AUC scores for different detectors on the same dataset.

[9]Hanley and McNeil (1982) discuss the underestimation of the "true" AUC when an empirical ROC with a handful of rating categories are constructed, and the effect is stronger for just one operating point – equivalent to using two rating categories only. Katostaras and Katostara (2013) introduce a special formula that adjusts for this underestimation.

Cohen's Kappa

Cohen's Kappa (Cohen, 1960) is the measure of agreement between two sources of labels that is also based on a confusion matrix and can be calculated in both, binary and multiclass setting. Compared to accuracy, precision, sensitivity, and other similar measures, Cohen's kappa (denoted κ) directly accounts for class imbalance by effectively quantifying the *relative improvement* the algorithm makes compared to what a random mix-up of the class labels it assigns would achieve:

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \tag{12}$$

where p_o is the *observed* agreement, equivalent to accuracy, and p_e is the the agreement *expected* by chance, calculated as[10]:

$$p_e = \sum_{i=1}^{N} \frac{C_{i\cdot} \times C_{\cdot i}}{s^2}, \tag{13}$$

where C is confusion matrix for N classes; $C_{i\cdot} = \sum_{j=1}^{N} C_{ij}$ (the sum of i^{th} row – total number of entities of class i in the ground truth); $C_{\cdot i} = \sum_{k=1}^{N} C_{ki}$ (the sum of i^{th} column – total number of entrites labelled as class i by the algorithm); $s = \sum_{i,j=1}^{N} C_{ij}$ – total number of entities in C. Thus, p_e quantifies how often the ground truth and the algorithm would on average agree on the label for a single entity, given the respective frequencies of each label (for class i: $C_{i\cdot}/s$ in the ground truth and $C_{\cdot i}/s$ in the algorithm's labels).

Cohen's kappa is, by far, the most frequently reported metric when evaluating the performance of eye movement detection algorithms (Larsson et al., 2013, Tables 6 and 7; Larsson et al., 2015, Tables 4, 5 and 6; Larsson et al., 2016, Table 9; Santini et al., 2016, Fig. 6; Andersson et al., 2017, Table 7; Pekkanen & Lappi, 2017, Table 1; Hooge et al., 2018, Table 3; Zemblys et al., 2018, Table 3, Figs. 7 and 10; Startsev et al., 2019a, Table 7; Startsev et al., 2019, Tables 2 and 3; Zemblys et al., 2019b, Tables 6 and 8; Bellet et al., 2019, Tables 2, 3 and 5; Wadehn et al., 2019, Table 2; Dar et al., 2020, Table 3, and many more).

Despite of its popularity, there are a number of issues causing Cohen's kappa to produce unreliable scores. Some criticism with regards to not taking bias and prevalence in the dataset into account was voiced by Byrt et al. (1993). However, the proposed correction effectively turn the metric into accuracy (the formula for both prevalence- and bias-corrected κ can be rewritten with the use of a

[10]Equation adapted from Delgado and Tibau (2019)

constant and accuracy). It is, nevertheless, suggested that Cohen's kappa should not be reported without additional statistics (prevalence and bias). Furthermore, Delgado and Tibau (2019), among others, advocate against using Cohen's kappa and show that in some cases when marginal probabilities are very small, the κ score can be affected to the extent that worse classification results can obtain higher scores.

Adjusted event-level Cohen's kappa In the formulation above, Cohen's kappa quantifies the improvement an algorithm makes by ordering its predicted labels (for eye movement events or samples, depending on the level of the evaluation) compared to a random shuffling of the same entities (i.e. "agreement by chance"). For event-level evaluation, this random agreement is equivalent to shuffling the order of the *labels* of the events, without changing the events' *durations*. This means that this condition includes assigning e.g. saccade labels to one second-long fixation events, etc. As this behavior may not represent a reasonable chance-level baseline for performance evaluation, and, therefore, may be suboptimal in the role of a normalization statistic for the kappa formula, Startsev et al. (2019) re-defined the procedure for computing chance-level agreement for event-level evaluation. Instead of shuffling labels alone, that work proposed shuffling the temporal order of whole *events*. Not to shift the distribution of event statistics, the same-class events that end up one after the other as the result of this shuffling are not merged together. This shuffling can be repeated multiple times to stabilize the statistic (should not be required for large-scale datasets). After every shuffling, F1-score for the evaluated class is computed (though not named as such in the paper; contrasted to all-class accuracy in traditional Cohen's kappa) so as not to penalize poor temporal alignment of e.g. saccades to the ones in the ground truth when evaluating fixation detection. The average agreement value is used as p_e in Eq. 12. The value of p_o is also computed via the positive-class F1 score. For multiclass evaluation, all-class accuracy is used – same as for traditional Cohen's kappa analysis.

Correlation

Correlation quantifies how two variables change with the respect of one another. Depending on the nature of the two compared variables, correlation can be measured by calculating, for example, Pearson's, Spearman's or Kendall's correlation coefficients. Although not describing which correlation measure was used, Munn et al. (2008, Figures 3 and 6) employed a correlation coefficient to compare fixation detection performance between the three coders and the I-VT (Salvucci & Goldberg, 2000) algorithm. In the case of binary classification, the correlation measures listed above would yield the same result (see Table 6 in Appendix D). However, neither of them is applicable in the multi-class case, because event classes are neither continuous nor discrete ordinal variables.

A correlation measure suitable for evaluating classification results is called *Matthews Correlation Coefficient – MCC* (Matthews, 1975). In a binary case, as originally proposed by Matthews (1975), MCC is equivalent to *phi coefficient* (Cramir, 1946) and is defined as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{14}$$

A natural extension to multiclass case was proposed by Gorodkin (2004) and is calculated as[11]:

$$MCC_{\text{multiclass}} = \frac{c \times s - \sum_n^N p_n \times t_n}{\sqrt{\left(s^2 - \sum_n^N p_n^2\right) \times \left(s^2 - \sum_n^N t_n^2\right)}} \tag{15}$$

where, for a confusion matrix C for N classes,

- $c = \sum_{n=1}^{N} C_{nn}$ – the total number of correctly predicted entities in C;
- $s = \sum_{i,j=1}^{N} C_{ij}$ – the total number of entities in C;
- $t_n = \sum_{i=1}^{N} C_{ni}$ – the number of occurrences of class n in the ground truth; and
- $p_n = \sum_{i=1}^{N} C_{in}$ – the number of occurrences of class n in the algorithm's predictions.

Chicco and Jurman (2020) compared Matthews Correlation Coefficient to accuracy and F1-score in a binary classification case and concluded that MCC, contrary to the other two metrics, is robust when evaluating imbalanced data. In order to achieve a high MCC score, the "classifier has to make correct predictions both on the majority of the negative cases, and on the majority of the positive cases, independently of their ratios in the overall dataset". Despite of its obvious advantages, to the best of our knowledge, Matthews Correlation Coefficient was never used when evaluating the performance of event detection algorithms.

Summary

Table 3 summarises metrics for evaluating the sample or event level performance of the eye movement detection algorithms. All discussed metrics can be used in a binary setting, i.e. separately calculating the performance for each eye movement event type that the algorithm can detect. These per-class scores can then be averaged (or weighted in some other way) to provide an overall performance score

[11] Equation adapted from https://scikit-learn.org/stable/modules/model_evaluation.html#matthews-corrcoef

if needed. Accuracy, Matthews Correlation Coefficient, Cohen's kappa, and Levenshtein distance can also be directly calculated while considering all event classes at once. Weighted accuracy is the only parameterized metric. Although its parameters (the per-class weights) can be inferred from the ground truth annotations, it has to be noted that there are several ways of accounting for class imbalance via weights: e.g. the class with a share of samples or events of p could get a weight of p, $(1 - p)$, $\frac{1}{N}$, where N is the number of classes, or similar.

Another property listed in Table 3 is robustness against the corner-cases, i.e. the metric's ability to provide low scores for very simple detectors – randomly assigning event labels (with equal probability), labelling all samples with the majority- or minority-class label, randomly shuffling the ground truth labels, and assigning predictions of the "wrong" class to all samples. We computed all the metrics we list in the table for these scenarios and provide respective scores in Table 6 in Appendix D. These tests are similar to what was performed in Startsev et al. (2019), though the analysis in the latter work focused on event-level evaluation, and was dataset-dependent (i.e. performed on real data). Here, we aimed for a simplified analysis, and therefore opted for a fixed example of ground truth labels: a sequence of alternating fixations and saccades, all fixations and saccades comprising 90 and 10 samples, respectively, to simulate a realistic class imbalance. We did not include other classes for the reason of simplification as well.

Note that this corner-case robustness score can be seen as an empirical assessment of the ability of each metric to describe the performance of the event detection algorithm *on its own*. – i.e. without any additional measures or without comparing the score to that of any other algorithm or baseline. This is a rather uncommon scenario, since studies overwhelmingly report several performance measures for at least two algorithms that are being compared. However, ranking the metrics according to their usability in a simple context provides an additional way to characterise them.

We assigned three "usability" levels – low, medium, and high, – and include these in Table 3. Rank was assigned to all the metrics based on the number of properties listed in Table 3 that they possess (e.g. whether the score can be directly defined for a multiclass case, does it account for imbalanced data, etc.). The most important role in this scoring was attributed to the corner-case robustness, since without additional analysis of e.g. label distribution in the ground truth or algorithm predictions we cannot exclude that we are dealing with a corner-case. Believing in the reasonable performance of an algorithm (supported by a high performance score) while it is in reality no better than chance is a poor outcome of an evaluation, and a dangerous conclusion to reach.

We note that the corner-case robustness tests in Appendix D were performed in conjunction with *sample-level* evaluation. This was done to maintain the simple and easily reproducible set of corner-case tests. In principle, the

Table 3 Summary of the metrics for eye movement detection performance evaluation. In the *Range* column the target value (best performance) is in bold. For Cohen's kappa and MCC, values below 0 indicate performance below chance level; for ROC AUC – values below 0.5 have the same meaning. For other evaluation metrics, the chance level performance is dependent on the class balance, and such a threshold cannot be provided

	Binary	Multiclass	Accounts for imbalanced data	Parameter-free	Corner-case robustness	Range	Usability on its own
Accuracy	✓	✓		✓		[0; **1**]	Low
Balanced accuracy	✓	✓	✓		✓	[0; **1**]	Medium
Precision	✓			✓		[0; **1**]	Low
Sensitivity	✓			✓		[0; **1**]	Low
Specificity	✓			✓		[0; **1**]	Low
F1-score	✓			✓		[0; **1**]	Low
Jacard index	✓			✓		[0; **1**]	Low
Cohen's Kappa, κ	✓	✓	✓	✓	✓	[-1; **1**]	High
MCC	✓	✓	✓	✓	✓	[-1; **1**]	High
ROC AUC	✓			†	✓	[0; **1**]	Low
Length normalized Levenshtein distance	✓	✓		✓		[**0**; 1]‡	Low

†In this context, ROC AUC cannot be called parameter-free, since in most practical cases of eye movement event detection evaluation it is unclear what should be iterated in order to obtain the ROC curve, and this can be perceived as a relatively complex "parameter" when computing this metric.

‡Can be larger than 1 if the predictions sequence is longer than the ground truth (e.g. for event-level evaluation, see "Levenshtein distance")

conclusions about the metrics' usability can be extended to the event-level evaluation as well, as the main properties of the metrics remain unchanged, and the corner-case robustness in terms of class imbalance is also similar. However, *event-level* corner cases would bring their own set of problems (excessive event fragmentation or merging, temporal offsets, etc.), which would affect the event matching outcome, which in turn needs to be adequately handled by the metrics. This is difficult to cover with representative examples of ground truth and prediction pairs while testing an appropriate set of event matchers and metric pairs. Startsev et al. (2019) compared different event-level metrics with real data – manual annotations and real algorithm's predictions, – however only used slight variations of the same matcher. In "Interaction between the performance metrics and event matchers" we test different matcher-metric combinations on real data as well, although with the focus on their differences on average, and the interaction between the matchers and metrics.

Event matching methods

Event-level evaluation of the eye movement event detection algorithms (beyond average event statistics) is a quite recent development in the field, yet several methods for event matching were proposed. Below we describe in detail and discuss the pros and cons of event matching techniques applied to eye tracking data in the literature to date. This part of the evaluation pipeline seems to be the least standardized and the most diversely developing, so its review and improvement forms an important part of this paper.

Majority voting

Majority voting evaluation method, proposed by Hoppe and Bulling (2016) does not require explicit event matching, and the events as such (i.e. as uninterrupted sequences of sample labels) are never considered in one of the two compared event label sequences. Hoppe and Bulling (2016) evaluate their event detection algorithm by checking how the majority (over 50%) of the predicted samples corresponding to each ground truth event are labeled. A ground truth event can remain not classified if no class in the corresponding predictions obtained the necessary majority.

Although this approach is conceptually simple and easy to implement, it is prone to inflating evaluation scores and not accounting for the detection of false events. Majority voting is effectively a sample-level evaluation approach, the only difference being that the agreement is calculated for each of the ground truth events instead of each samples. See for example Fig. 2: When using Majority voting evaluation, same as for sample-level evaluation, the left and the right

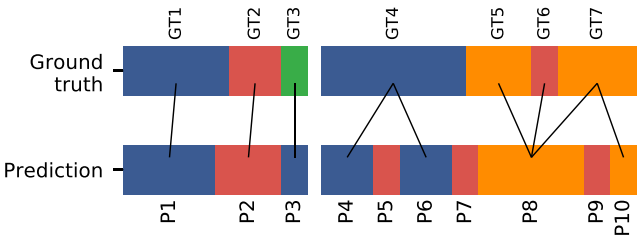

Fig. 4 Example of the Majority voting evaluation (Hoppe & Bulling, 2016). Blue are fixations, red – saccades, green – PSO, orange – smooth pursuit. Black lines connect the ground truth with predicted events that correspond to the sample-wise majority class during a particular ground truth event

predicted event sequences would score the same – one correct prediction of ground truth fixation will be registered, even though the two prediction sequences are considerably different qualitatively. Nevertheless, we consider Majority voting to be the event-level evaluation method since it accounts for events at least to some degree.

Figure 4 further illustrates the limitations of the majority voting evaluation approach. Events GT1 and GT2 are counted as correct predictions because the majority of the samples during these events are predicted to belong the same class as the corresponding ground truth events. GT3 and GT6 are counted as PSO-Fixation and Saccade-Pursuit errors (or false negatives in the case of per-event evaluation). Events GT4, GT5 and GT7 are further counted as correctly detected, however false saccades P5, P7 and P9 during these events are not accounted for, thus inflating any confusion matrix based evaluation score. Furthermore, note how event P8 contributes to making both GT5 and GT7 correctly detected events. However, as the predictions are only considered as samples, this evaluation method cannot in any way account for the corresponding event merging or splitting. Finally, the majority voting technique is not symmetric, meaning that if one swaps the ground truth and predicted sequences, the resulting matches and calculated score will differ.

Although Majority voting evaluation approach was not presented as an event matcher, it could be seen as such. All predicted events that belong to the majority class during each ground truth event can be considered as matched (see black lines in Fig. 4). This would enable one-to-one, but also one-to-many and many-to-one matches (i.e. detecting fragmented and merged events). However, similarly to Event-driven error characterization (see "Event-driven error characterization" below), a situation when events appear to be both merged and fragmented could arise (e.g. P8→[GT5;GT7] and GT7→[P8;P10] in Fig. 4) and would require additional decisions to handle.

Manual error coding

Friedman et al. (2018) compared four event detection algorithms using the taxonomy of 32 error types that were designed to cover all the decisions the algorithms need to make when detecting noise, fixations, saccades, and PSOs. Half of these error types described missing the event or mislabeling the event class, for example "PSO not detected", "fixation misclassified as noise", etc (Friedman et al., 2018, Table 1), which is essentially a reduced version of confusion matrix that excludes correctly detected events. The other half of the error types described timing errors such as "fixation starts too late" or "saccade ends too early". When evaluating algorithms, expert coders were presented with one second of the algorithm-labelled sequence in an interface that closely resembled those used for hand-labelling the eye movement events (Andersson et al., 2017; Hooge et al., 2018; Startsev et al., 2019b) and were asked to classify and count the errors the algorithm made in its predictions for the depicted part of the signal. Since it was possible that the exact same samples would be assigned different error types by different coders – for example, inaccurate detection of saccade offset can be interpreted as "saccade ending too early/late" or "PSO starting too early/late" error – authors used error type hierarchy, based on the order in which the Nyström and Holmqvist (2010) algorithm detects events. In addition, authors used several heuristic rules for coding the errors: saccade timing errors were only counted if timing was off by at least 3 samples (3 ms at 1000 Hz) from the human expert judgement (note that the exact expert judgement of where a saccade event border *should* have been is not explicitly recorded anywhere and exists only in the head of the expert); the first event in the recording and unclassified fixation periods shorter that 40 ms were not evaluated; saccades were defined according to the shape in the gaze position plot, while PSOs had to have a certain velocity profile and start immediately after the saccade. Such heuristic rules would need to be updated in case of e.g. a different sampling frequency of the underlying gaze data, and these modifications are not always obvious.

While the approach by Friedman et al. (2018) does not require pre-annotated ground truth labels, an implicit assumption remains the same as for obtaining the ground truth labels, i.e. that there exists a perfect set of labels for a given sequence of gaze points, and an expert annotator could potentially produce it.

The major difference is rather procedural: The annotation of error types does not require precise adjustment of the border between the detected events, meaning that this could potentially be less time-consuming than manually coding all the events. However, as the authors themselves point out, manual error coding is only faster when comparing two or three algorithms (Friedman et al., 2018, p. 1375).

The major limitation of manual error coding is reproducibility. When comparing multiple algorithms, the coder is presented with the same data but labeled by different algorithms, therefore the same errors can potentially be coded differently on two different occasions. To be able to use this approach, coders first need to be trained not only to recognise fixations, saccades and other events in raw eye tracking data, but also to follow quite complex heuristic rules and hierarchy of error types. Friedman et al. (2018), therefore, held a training session where three raters first practiced by scoring a small set of data, while having open discussion followed by detailed comparisons of the results and discussions to reach consensus. However even such training does not guarantee that implicitly-set internal thresholds (Hooge et al., 2018) for registering a certain type of error would not shift over time, leading to the impossibility of fairly evaluating a new algorithm e.g. a year later or by a different coder. Much like humans are not a gold standard of fixation annotation when one wants to consistently assign labels to all recordings with the same set of rules (Hooge et al., 2018), the reliability of the results obtained using the evaluation approach by Friedman et al. (2018) can be very problematic, potentially making two scores obtained in the same procedure incomparable. Furthermore, since this method does not account for the correctly detected events, it is impossible to compare the performance of the algorithm across different datasets. The absolute number of errors is only meaningful when comparing different scoring on the same data, but if the algorithm was to be evaluated using different datasets, no conclusions about the performance could be drawn. Since datasets can differ in size and, therefore, in the total number of events, finding the same number of errors in two datasets does not mean that the algorithm performs equally well on both. In this case, a meaningful performance metric would be the proportion of errors to the number of correctly classified events, or the number of errors per one second of an eye tracking recording, assuming a comparable frequency of events in the different datasets, which is not always an acceptable assumption.

Yet another limitation is presented by the error labeling hierarchy that is based on the order in which one particular algorithm (Nyström & Holmqvist, 2010) detects eye movement events. Different algorithms might process events in a different order, while in the case of machine learning-based algorithms such event processing order does not exist at all – each sample is rather assigned a probability of belonging to a certain event type , without a discernible order of steps. As a result, a fixed error type hierarchy would be rather a disadvantage of the evaluation scheme, instead of helping obtain tailored insights into the particular parts of the detector.

Earliest overlap matching

Hooge et al. (2018) proposed using a simple method for explicit single-type event matching – they match each reference (ground truth) fixation with the first overlapping fixation in the test sequence. All matches are one-to-one, i.e. when two fixations are matched they are withdrawn from the pool of possible matches for further analysis. For example, in Figure 5 fixation GT1 in the ground truth overlaps with two fixations in the prediction sequence (P2 and P4). Fixations GT1 and P2 are matched since they overlap earliest in time, and this match can then be counted as a true positive. Meanwhile, the predicted fixation P4 remains unmatched as it does not overlap with any still available ground truth fixation, and can be subsequently counted as an error (a false positive). Similarly, the ground truth fixation GT3 is matched with the fixation P6, while the ground truth fixation GT5 remains unmatched and can be counted as an undetected event, i.e. a false negative error. And finally, the GT7→P8 match can be registered as another correct (true positive) detection. Based on this event matching, Hooge et al. (2018) computed both confusion matrix-based scores (F1-score) and event-quality metrics (see RTO and RTD in "Event quality metrics") in order to compare the annotation patterns for each pair of the twelve expert coders in their data.

Despite being a one-to-one matching technique, the earliest overlap method can be seen as being able to deal with event fragmentation and merging, in a certain context. While in terms of event *detection* evaluation it will only be able to match one event in the ground truth to one predicted event, the situation is different for the event *timing* evaluation pipeline proposed by Hooge et al. (2018): In their work, the timing of event on- and offsets are evaluated separately, with the earliest overlap matcher applied in two directions. For event *onsets*, the matcher is applied as described above, from the start to the end of the recording. For the *offset* timing evaluation, however, the order is reversed, and the matcher effectively prefers the events overlapping *latest* in time. An example demonstrating the potential usefulness of this inversion can be seen in Fig. 5, where for event offset evaluation, the offset of fixation GT1 would be compared to that of P4, and the offset of P6 – to GT6 (see dashed black lines in the figure). For timing evaluation on the whole then, GT1 can be seen as fragmented into P2 and P4, while GT3 and GT5 are seen as if merged into a detection P4. Without additional modifications to the evaluation pipeline, however, these implied merges and fragmentations are not reflected in the evaluation of the event detection performance.

Earliest overlap event matching was designed in the context of a binary evaluation setting only (i.e. evaluating the performance of detecting a single eye movement type;

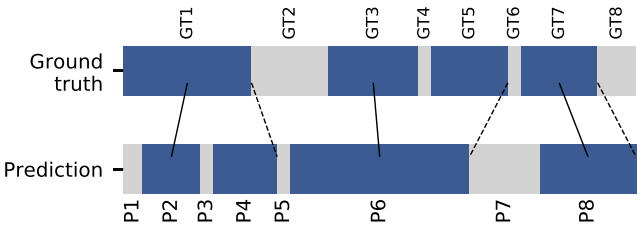

Fig. 5 Example of the Earliest overlap event matching method (replicates Fig. 10 from Hooge et al., 2018). Blue are fixations, gray – unlabeled events. Solid black lines between the ground truth and predicted events indicate event matches, dashed black lines show event matches when earliest overlap matching is performed in reverse order (for fixation offset timing evaluation; hence, the endpoints of events are connected with dashed lines)

fixations vs. non-fixation). As noted by Zemblys et al. (2019b, p. 845), using it for multiclass event analysis will cause *unintuitive*[12] event matching. Therefore, when used in combination with most of the event-quality metrics discussed in "Event quality metrics", it can result in both over- or under-estimation of the algorithms' performance. Yet, in some scenarios, unintuitive matching can also occur in binary setting. For example, even a slight overlap with an earlier positive event is preferred over a much larger overlap with a later negative event (see GT2→P1 match in Fig. 6). Fixations GT2 and P1 in ground truth and prediction sequences respectively are matched despite of majority of the samples in the predicted sequence during the fixation GT2 being labeled as an event P2 of another type. Such scenario is very likely – e.g. event P2 can be a smooth pursuit detected by the algorithm. Mislabeling fixations as pursuits and vice versa is a very common behavior in event detection algorithms (Holmqvist et al., 2011, Section 5.8; Hoppe and Bulling, 2016). Evaluation using the Earliest overlap event matching method would unfairly benefit the algorithm's scores, even if its detection is poor: This particular GT2→P1 match (Fig. 6) would be counted as a correct detection of a fixation, albeit with potentially poor event-quality scores.

Multiclass extension of the earliest overlap matching

The issue of poorly handling sequences of events of multiple classes was not obvious in the context in which the earliest overlap event matching method was employed in Hooge et al. (2018), as only one class – fixation – was considered. In the case when all gaze samples are assigned a label, the question of how to apply this matching strategy does not have a good answer. There are two naturally arising

[12]We use the word "unintuitive" instead of "incorrect", because the matching algorithm is not actually making an error and doing exactly what it was designed to do. However the resulting matching may not be what one would intuitively expect.

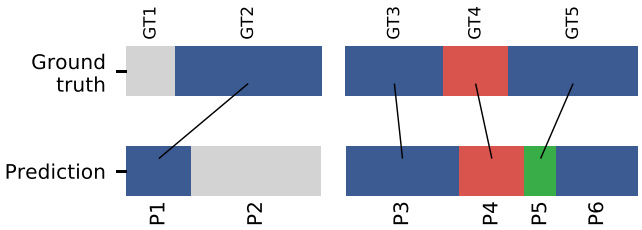

Fig. 6 Example of unintuitive event matching in binary and multiclass setting using the Earliest overlap event matching method (Hooge et al., 2018). Blue are fixations, red – saccades, green – PSOs, and gray – unlabeled events. Black lines between the ground truth and predicted events indicate event matches

possibilities (Zemblys et al., 2019b, p. 845): (i) Ignore the samples of the classes that are not evaluated at the moment (i.e. first match saccades, then fixation detection, etc.), and (ii) keep the samples of all the classes and performing the matching as-is[13]. The alternative (i) is effectively erasing the multiclass nature of the problem, and converts matching into a binary set-up as originally proposed by Hooge et al. (2018) with all the limitations described earlier. It is also identical to our multiclass extension of Hauperich et al. (2020) approach (see "Overlap matching").

For the alternative (ii), the situation of unintuitive matching as exemplified in Fig. 6 can lead to the underestimation of the algorithm's performance. The Earliest overlap method matches fixation GT6 in the ground truth with the predicted PSO P5 despite only slight overlap between these events, while the majority of the samples constituting GT5 are correctly predicted as a fixation. This is by far not an unrealistic example of this shortcoming of the matching procedure. Consider, for instance, a perfect event detector, the predictions of which are all shifted by one sample (with the first sample of the recording being labelled as e.g. noise). Practically speaking, this is a very accurate algorithm with a minor offset in its detections. The Earliest overlap strategy applied in multiclass setting will, however, mean that all of the detections would be registered as incorrect.

Overlap matching

Hauperich et al. (2020, Fig. 4B) used a simple binary matching technique, where a match is registered if a positive event in the predicted sequence overlaps with positive ground truth events by at least one sample. The main evaluation in that work was performed by considering only one match from overlapping events and ignoring the rest of the events that are part of a merge or fragmentation. That is, in the event sequence in Fig. 5, Hauperich et al. (2020) approach ignores events P4 and GT5, thus not penalizing the performance score if the algorithm splits or merges the ground truth events. The authors argue that such an approach is reasonable because the number of fragmentations and merges depend on tunable parameters that the user of the algorithm has access to. In addition, the authors repeated the calculations by penalising the occurrence of unmatched events in fragmentation and merges (Hauperich et al., 2020, Fig. 5B). In the binary setting, this approach is identical to the Earliest overlap method (Hooge et al., 2018): The earliest of the overlapping positive events is chosen for a match since one-to-many or many-to-one matching is not allowed.

Multiclass extension of the overlap matching

In order to systematically compare different matcher and metric combination in "Interaction between the performance metrics and event matchers", we also extended the overlap event matching strategy to the multiclass setting. It is somewhat similar to the extension of the earliest overlap method, but instead of matching the ground truth event with the first overlapping event from the predicted sequence, multiclass overlap matcher looks for the first event of the same class, i.e. only allows matching fixations with fixations, saccades with saccades, etc.

However, Zemblys et al. (2019b, p. 845) argued that "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' ". Figure 7 illustrates a case of unintuitive event matching using Hauperich et al. (2020) approach in the multiclass setting. Despite the algorithm effectively confusing fixations and smooth pursuits, the overlap matcher identifies one correct prediction (GT2→P1), while the rest of the events remain unmatched (and during the subsequent evaluation can be counted as missed). Intuitive and useful (in a sense of helping to identify what errors the algorithm makes)

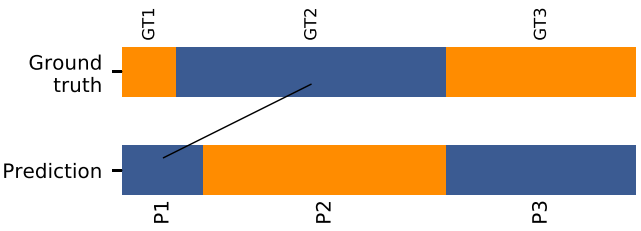

Fig. 7 Example of the Overlap event matching method (Hauperich et al., 2020) in a multiclass setting. Blue are fixations and orange – smooth pursuit events. Black lines between the ground truth and predicted events indicate event matches

[13]Note that our multiclass extension of the Hooge et al. (2018) matching method ignores *undefined* events to best reflect what the original approach is doing: matching "positive" and ignoring "negative" events, where "negative" event in multiclass setting is the *undefined* class.

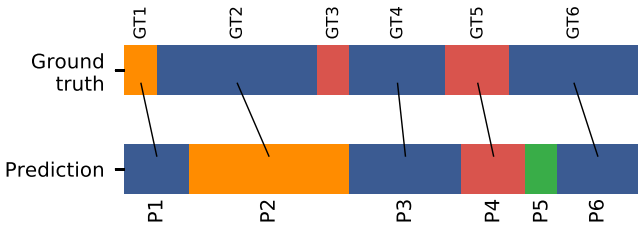

Fig. 8 Example of the Maximum overlap event matching method (Zemblys et al., 2019b). Blue are fixations, red – saccades, green – PSOs, and orange – smooth pursuit events. Black lines between the ground truth and predicted events indicate event matches

matching in this example would be to report pursuit-fixation and fixation-pursuit errors (GT1→P1, GT2→P2, and GT3→P3), which can be achieved with e.g. the Maximum overlap matching technique described below (also see Fig. 8).

Maximum overlap matching

To establish a more intuitive one-to-one correspondence between the two sets of events, Zemblys et al. (2019b) proposed matching events based on the *amount of overlap* (measured in the number of gaze samples or in time units) between the ground truth and predicted events. For every event in the ground truth, all overlapping events in the predicted sequence are examined and the one with the largest overlap is selected as a match. Maximum overlap method thus presents a combination of event-level evaluation procedures by Hoppe and Bulling (2016) and Hooge et al. (2018) that solves some of the issues characteristic to these methods. First, unlike Majority voting, it explicitly performs event matching, thereby enabling the use of any evaluation metrics described in "Evaluation metrics". Second, it ensures a more intuitive matching, where there is some guarantee on the quality of the matches aside from the matched events just intersecting. In addition, maximum overlap method can be directly applied for both binary and multiclass event-level evaluation.

Figure 8 shows an example of applying the maximum overlap event matching method. Note how fixation GT2 is now matched with event P2 (contrary to GT2→P1 match in Fig. 6 for Earliest overlap), because of the much larger overlap for the GT2–P2 pair. Similarly, fixation GT6 is now matched with fixation P6 instead of PSO P5 when earliest overlap method was used (see Fig. 6). The result of the event detection evaluation in Fig. 8, therefore, would consist of three correct detections, two misclasification errors, one missed event (GT3), and one false event detection (P5).

In some scenarios the Maximum overlap method is subject to misplacing the cause of the mistake made by an algorithm. The classes of the eye movements are inherently unbalanced in terms of their durations – for example, saccades and PSOs are by their nature short, in order of tens of milliseconds, while fixations or smooth pursuits can span several seconds. This leads to a disproportionate number of samples assigned to certain classes. Therefore, e.g. for a 20 ms saccade, a 10 ms overlap with a fixation would mean a different (intuitive) degree of overlap compared to the 5 ms overlap with another saccade. This case is illustrated in Fig. 9. Saccade P9 overlaps with the fixation GT5 and the saccade GT6. Because the overlap between GT5 and P9 is larger, the match between this event pair is preferred when using the Maximum overlap method, which results in reporting one misclassification (fixation GT5 → saccade P9) and one false negative error (the "missed" GT6). However, a more intuitive outcome would be to match GT6→P9, since the algorithm in Fig. 9 actually detected the ground truth saccade GT6, even if this detection was not perfectly timed.

While Maximum overlap improves on the Earliest overlap when it comes to event *detection* evaluation, it loses some ground in terms of evaluating the event *timing* (for example when comparing how two coders differ in labeling on- and offsets of fixations (Hooge et al., 2018)). The Maximum overlap approach does not address the issue of event fragmentation and merging and, therefore, the subsequent analysis of timing differences can result in inaccurate estimation of timing errors. In Fig. 9, the ground truth fixation GT1 was split into three separate fixations (events P1, P3, P6) by falsely detecting two saccades P2, P4 and a PSO (event P5). Since GT1 and P3 get matched because of the highest overlap, comparing on- and offset differences between these two fixations would signal large disagreement. In reality, however, the algorithm that produced the predictions illustrated in the figure has accurately detected both, the onset and the offset of the fixation GT1 by accurately labeling the onset of the fixation P1 and the offset of the fixation P6. Similarly, the predictor in Fig. 9 missed the saccade GT4 and, therefore, merged two fixations GT3 and GT5 into one long fixation P8. The Maximum overlap matches GT3 to P8, and as P8 is thereby taken out of the possible matches for GT5, the latter

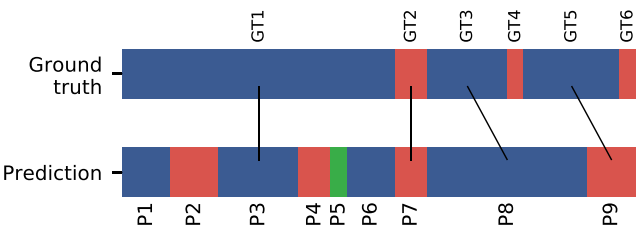

Fig. 9 Example of the limitations and unintuitive event matching of the Maximum overlap method (Zemblys et al., 2019b). Blue are fixations, red – saccades, and green – PSOs. Black lines between the ground truth and predicted events indicate event matches

is matched to a saccade P9. If earliest overlap matching method were to be used to analyse these two sequences, fixation *onset* error would be estimated to be zero (since GT1 would get matched with P1 instead of P3, and GT3 remains matched to P8). The *offset* timing error would only take into account the small difference between offsets of events GT5 and P8 (earliest overlap matching is performed in reverse order for offset timing difference estimation, see "Earliest overlap matching"), instead of the large difference between offsets of GT3 and P8 in the case of Maximum overlap matching. The shortcoming related to event timing as exemplified above can be generalized to event quality measures ("Event quality metrics") in general. It also is not specific to the Maximum overlap matching, but applies to any one-to-one matching technique, unless special steps are taken as for the Earliest overlap matcher – e.g. the Maximum intersection-over-union matcher described in "Maximum intersection-over-union matching" suffers from this problem just as much.

Maximum intersection-over-union matching

First utilized for eye movement events in Startsev et al. (2019a), this strategy is very similar to the Maximum overlap matcher of Zemblys et al. (2019b) that was introduced at the same time. The difference is that when choosing a match for a given ground truth episode, the decision is based on maximising not the overlap itself, but the ratio of the episode pair's overlap to their joint length (referred to as intersection-over-union ratio in the literature; IoU) instead. This helps the matching technique avoid the unintuitive matching of the maximum overlap as exemplified in the Fig. 9 (GT5→P9 match). When using Maximum IoU matching, the predicted saccade P9 is now matched with the ground truth saccade GT6 (Fig. 10), because IoU for the GT6→P9 pair is 0.33, while for GT5→P9 it is lower – 0.29. In general, this matching scheme makes it less likely that a very long event could be matched with a very short one, allowing for a more intuitive error attribution.

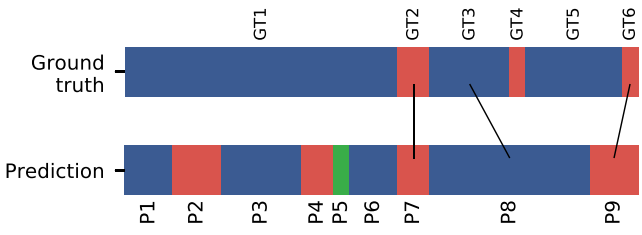

Fig. 10 Example of Maximum intersection-over-union (IoU) event matching method (Startsev et al., 2019a) with IoU threshold of 0.3. Blue are fixations, red – saccades, and green – PSOs. Black lines between the ground truth and predicted events indicate event matches

An important feature of IoU matching method is that the used criterion involves a measure that falls within the same [0; 1] range for any event pair, of any of the eye movement classes. This means that the matching criterion can be strengthened by using a class-independent IoU threshold, thus forcing the matched event pairs to adhere to a higher standard of what a "match" is, i.e. only allowing the matches where the IoU is above a certain threshold (Startsev et al., 2019a; Startsev et al., 2019; Voloh et al., 2020). In Fig. 10 a threshold of IoU>0.3 is used, and the ground truth fixation GT1 is not matched to any event from the predicted sequence as the highest possible IoU for a potential match with this event is only 0.29 (GT1→P3). Setting a threshold at IoU>0.5 or higher means that there can be no more than one candidate match for any given ground truth event, i.e. while selection is still based on the maximum IoU, for any given pair of overlapping events in the ground truth and predicted sequences, if one pair satisfies the IoU>0.5 inequality, there can be no other pair that satisfies it as well. Such a threshold would consequently simplify the implementation of the matching procedure, as there is no need to compare the IoU values for different candidate matches. This or similar match-strengthening criteria can be in principle combined with any event matcher, either potentially simplifying its algorithm or adding flexibility via a variable matching strictness level.

Note on implementation differences

At this point we also address one non-obvious detail of implementing the ideas of "matching events with maximum overlap" or "with maximum IoU" that will influence the properties of the matcher itself and the evaluation results, namely – the order in which the potential matches are considered.

In the implementation of the maximum-overlap matcher in Zemblys et al. (2019b), all non-zero overlap between event pairs in the ground truth and in the predictions of the algorithm are recorded. These overlapping pairs are then sorted by their overlap and iterated through from maximum overlap downwards (highest-to-lowest iteration order). If neither of the events in the candidate pair is matched to another event yet, the candidate pair is recorded as a match; otherwise, the candidate pair is skipped.

In the implementation of the maximum-IoU matcher in Startsev et al. (2019a), the ground truth events are considered in their temporal order (first-to-last iteration order). For each event in the ground truth, all predicted events overlapping with it are considered. The one with the highest IoU is selected and the corresponding event pair is recorded as a match. The already matched predicted event cannot be a match to any further ground truth event.

Both iteration orders are valid, however would yield different matching results. For example, in sequences depicted in Fig. 10, the maximum-IoU matcher without an IoU threshold but with first-to-last iteration order would produce a match between fixation GT5 and detected saccade P9, even though IoU(GT5, P9) is lower than IoU(GT6, P9). This happens simply because GT5 would be considered before GT6, and P9 would at that point be still unmatched. In the highest-to-lowest iteration order, the GT6→P9 potential match would be considered before GT5→P9 due to the higher IoU, and thus have priority. With a match IoU threshold of over 0.5, however, the iteration order would not make any difference.

In addition, the highest-to-lowest iteration order ensures the symmetry of the matcher, since the sequences play equal roles in the process (unlike in the first-to-last iteration, where the ground truth events guide the iteration process). As a consequence of these differences, we opted for highest-to-lowest iteration order for the maximum-IoU approach in our codebase and analyses. The only drawback of this approach is the increased computational complexity, as potential matches have to be sorted, resulting in linearithmic time (w.r.t. the number of potential events pairs), while the first-to-last iteration order can be implemented with linear computational complexity.

Window-based matching

Kothari et al. (2020) proposed Event-Level Cross-Category metric – ELC – that they argue solves the issues with the reliability of timing offset evaluation when using either earliest – Hooge et al. (2018) – or maximum overlap – Zemblys et al. (2019b) – matching methods. ELC also works in multiclass setting and is able to handle event fragmentation (but not merge) errors. To enable this, Kothari et al. (2020) introduce what they call *Window-based matching* – an approach to ensure event match quality by adding timing constraints on when events are considered to be matched. More specifically, the ground truth event is matched and counted as a correct prediction only when its onset and offset roughly align with onsets and offsets of test events of the same class. Ground truth events that are completely contained within a predicted event of the same type are called *detached* events. The authors suggest that these can be safely counted as correct predictions (and, correspondingly, as successful matches) depending on the strictness of user's requirements (Kothari et al., 2020, p. 11).

Figure 11 shows an example of the proposed Window-based event matching procedure. The ground truth fixation GT1 is matched with the predicted fixation P1, since their onsets align perfectly (green dashed line from GT1 to P1 in Figure 11) and offsets are within a defined search window (red dashed line from GT1 to P1 in Fig. 11). Similarly, the ground truth fixation GT6 is matched with predicted fixations P6 and P8, since the onset and the offset of GT6 align perfectly with the onset of fixation P6 and the offset of fixation P8 (green and red dashed lines from GT1 to P6 and P8, respectively, in Fig. 11). The ground truth and predicted smooth pursuit episodes GT4 and P3 are slightly misaligned, however the misalignment is smaller than a defined search window and, therefore, the two events get matched. Blink episode GT2 and saccade GT5 in this example, even if they overlap, respectively, with blink P2 and saccade P4 are not matched because while the onsets of both corresponding events pairs are, under the selected search window criterion, aligned, the offsets are not. The blink event GT2 in Fig. 11) is fully contained inside the predicted blink P2, and thus presents an example of a *detached* event, in the nomenclature of Kothari et al. (2020).

After identifying all correct predictions via Window-based event matching, ELC proceeds with calculating detection errors. While normally this would not concern the descriptions of the event matching, the procedure proposed in conjunction with computing ELC makes slight changes in the event timing and establishes further "matches" in the two label sequences that need to be considered if one attempts to use the event matching from this work as a standalone and generically applicable event matcher. The proposed ELC pipeline first corrects the timing errors in the ground truth and predicted sequences, and then calculates event *mismatches* in the aligned sequences. Timing correction is applied on all matched transition points (green and red dashed lines in Fig. 11), regardless

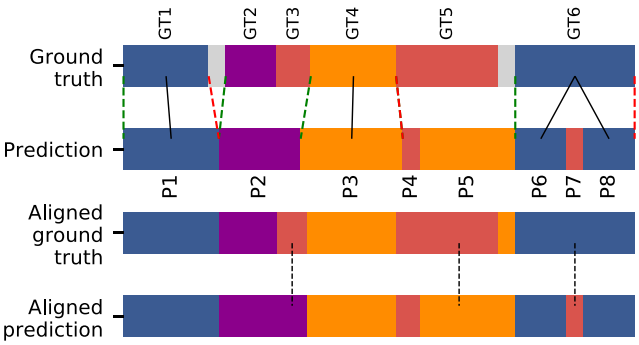

Fig. 11 Example of Window-based event matching, adapted from Kothari et al. (2020, Fig. 8). Blue, magenta, red, orange, and gray are fixations, blinks, saccades, smooth pursuits, and undefined events respectively. Black lines between the ground truth and predicted events indicate correct event matches. Green and red dashed lines indicate event onset and offset transition point matches. Black dashed lines in the two bottom scarf plots indicate error type mapping after event transition point alignment

of the match status of the corresponding events. For each matched transition point timestamps of the ground truth and predicted events are averaged to create a single transition point. Before the alignment, ELC replaces unlabeled samples with corresponding samples from the other sequence. ELC alignment procedure enables fast and easy calculation of event classification errors based on direct sample mismatches between the two aligned sequences (two bottom scarf plots in Fig. 11). For example, for the sequences in Fig. 11 ELC registers saccade-blink, saccade-pursuit and fixation-saccade errors (see black dashed lines between the aligned event sequences in Fig. 11). All correct and incorrect matches can then be analysed using most of the metrics described in "Evaluation metrics". It is then obvious that in order to obtain not just the same-class "positive" matches from the Window-based approach, this alignment needs to be considered as part of the matcher (even though it alters the original event timing).

The advantage of the window-based matching is that it in many cases can correctly handle event fragmentation, allowing for more accurate timing and detection evaluation (compared to purely one-to-one matchers). By only allowing the same-class events to be matched when certain on- and offset timing conditions are met, it also accounts for the quality of the correct event matches (similarly to setting an IoU threshold for the Maximum intersection-over-union matcher). In addition, the matching technique of Kothari et al. (2020) removes potential error type assignment ambiguity by resolving the differences in the timing of event transitions between the ground truth and predicted events. See for example the saccade GT3 in Fig. 11: It overlaps with the blink P2 and the pursuit P3; however, after the alignment (two bottom scarf plots in Fig. 11), this ambiguity is removed and the missed saccade GT3 can be considered as a saccade-blink error in subsequent evaluation.

One of the drawbacks of Window-based event matching is, however, its asymmetry, meaning that it gives different result depending on which event sequence is considered to be the ground truth and which – the prediction. The authors, therefore, propose performing the evaluation twice by interchanging the prediction and ground truth sequences, and averaging the resulting measures of choice. This event matcher also heavily depends on a choice of the threshold (the search window size for the transition point match) that might cause inter-study or inter-dataset result comparison issues, especially if is used with data sampled at different frequencies. It might be difficult for an inexperienced user to select an appropriate threshold and, moreover, thresholds might need to be different for different event types and sampling rates. In their paper Kothari et al. (2020) used $\pm$25 ms for saccades and $\pm$35 ms for other events. However, if used with low sampling rate data, e.g. 30 Hz, these values correspond to only $\approx$1 sample and therefore might be too strict for defining a correct match. The authors propose that intuitive value for a threshold is the duration of the shortest event in the annotations[14].

Window-based matching is also prone to unintuitive event matching and, in some cases, can potentially incorrectly evaluate the algorithm's performance by ignoring prediction errors. While being able to handle fragmentation of ground truth events, event merges do not get matched and, therefore, cannot be counted either as correct or as incorrect predictions by a subsequently computed metric – see fixations GT1 and GT3 that are merged into one predicted fixation P1 in Fig. 12. Fixation GT1 is considered to be a detached event and, as the authors note, can be safely counted as a match. However, the fixation GT3 would be ignored in the subsequent evaluation since it is not fully covered by the fixation P1. This inability to correctly handle simple cases of merging errors is especially important in the light of the authors' suggestion to perform the matching in both directions (ground truth $\rightarrow$ prediction and vice versa). When this route is taken, even the correctly handled fragmentation errors can potentially turn into unsupported merge errors (unless all of the merged events are fully contained in the "overarching" event).

Similarly to the Earliest overlap (Hooge et al., 2018) and Maximum overlap (Zemblys et al., 2019b) methods, the Window-based approach is prone to unintuitive event matching that misplaces the cause or labeling error. In Fig. 12, smooth pursuit episode GT5 is matched with short detected pursuit events P2 and P4 because they satisfy all event matching criteria. However, most of the samples during the pursuit GT5 were actually labeled as the fixation P3. While the matcher would still in this case register a GT5$\rightarrow$P3 match after the temporal alignment (which can be later counted as a pursuit-to-fixation misclassification error), it would also treat P2 and P4 as correct detections of GT5, thus effectively marking GT5 as a correctly detected event. Moreover, because of how sequence alignment in Window-based matching works, the ground truth saccade GT4, that in fact was mislabeled as a pursuit P2 (see top two scarf plots in Fig. 12), after the alignment is counted as saccade-fixation error.

In some cases the sequence alignment procedure can remove incorrectly detected events altogether. For example, consider a situation where the offset of the ground truth event is matched with a certain predicted event (see GT1$\rightarrow$P6 match in Fig. 13) that succeeds another predicted events that itself is a match for a *different* ground truth event (GT2$\rightarrow$P3 match in Fig. 13). In-between the two predicted events in question – P3 and P6 – a saccade P4 and a blink P5 are also detected. After the transition point alignment procedure, however, the predicted blink *P5 is*

[14] Yang, Z., personal communication, September 17, 2020

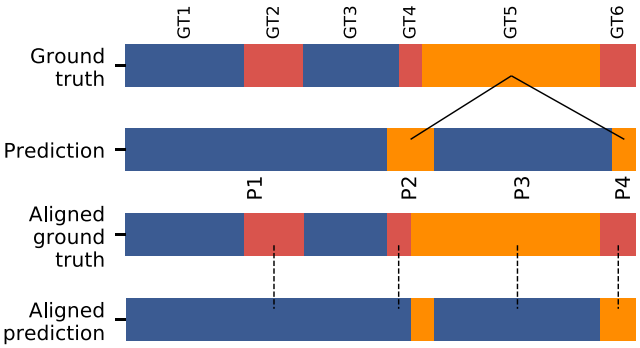

Fig. 12 Example of limitations and ambiguous cases for Window-based event matching (Kothari et al., 2020). Blue, red, and orange are fixations, saccades, and smooth pursuits respectively. Black lines between the ground truth and predicted events indicate correct event matches. Black dashed lines in the two bottom scarf plots indicate error type mapping after event transition point alignment

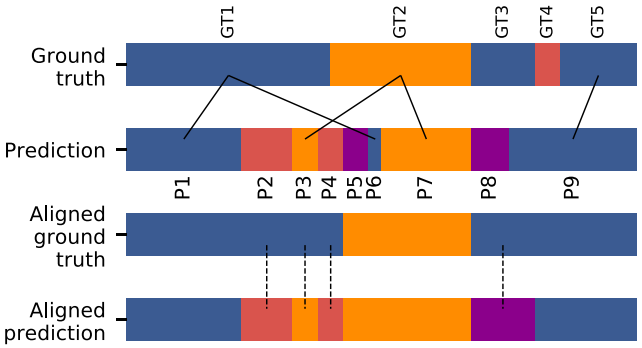

Fig. 13 Example of Window-based event matcher erasing events from the evaluation. Blue, magenta, red and orange are fixations, blinks, saccades and smooth pursuits respectively. Black lines between the ground truth and predicted events indicate correct event matches. Black dashed lines in the two bottom scarf plots indicate error type mapping after event transition point alignment

removed from the evaluation (see bottom scarf plot in Fig. 13). Moreover, fixation P6, which intuitively should be counted as a pursuit-fixation error (because pursuit GT2 fully contains the fixation P6), is also removed from the evaluation, while a similar case of fixation-pursuit misclassification (samples corresponding to GT1 and P3) remains. Indeed, it is ambiguous how to include the saccade P4 and the blink P5 in the evaluation: While P5 is completely contained within GT2 and could be counted as pursuit-blink error, saccade P4 overlaps with both fixation GT1 and pursuit GT2. Unfortunately, Window-based matching approach of Kothari et al. (2020) is not able to handle such cases. A solution to this problem could be adding other matching criteria, that require the ground truth and predicted events to overlap instead of just checking the timing of their on- and offsets. However, even such addition would not solve the problem of event detection errors that are not accounted for in the matching process. See for example the match between fixations GT5 and P9 in Fig. 13. The saccade GT4 that is missed by the detector is short enough that the sequence alignment procedure (that stretches GT5 so that its onset matches the average of that of GT5 and P9) effectively erases GT4 from existence and, consequently, from any subsequent evaluation.

Event-driven error characterization

Most of the automated matching strategies discussed in this section – Earliest overlap (Hooge et al., 2018), Maximum overlap (Zemblys et al., 2019b), and Maximum intersection-over-union (Startsev et al., 2019a) – do not address the issue of event fragmentation or merging: They only allow for events to be matched one-to-one, excluding the possibility of quantifying the extent to which the event detection algorithm suffers from splitting ground truth events into several smaller ones, or merging several several events into one. Window-based matching (Kothari et al., 2020) is only able to handle fragmentation, but not merge errors. Both of these error types can, in principle, be seen as inserting or missing the "in-between" events, respectively, that the aforementioned matching schemes can register, but a direct quantification of event fragmentation or merging is, nevertheless, not enabled.

Ward et al. (2006) proposed Event-driven error characterization (*EDEC*) – a method for scoring event errors based on the idea of *segments* (sequences of samples where *neither* ground truth *nor* prediction change) – that is able to quantify four error types, including fragmentation and merge. *EDEC* was later used by Steil et al. (2018, Figures 4 and 6) when evaluating their fixation detection algorithm for head-mounted eye-trackers. Ward et al. (2006, Figure 2, left) provide an algorithm for assigning one of four error types – insertion, merge, deletion, and fragmentation – to the events that were incorrectly detected or missed. The rest of the events where both prediction and the corresponding ground truth are not assigned to any of the four error categories are labeled as *correct*. In addition, correct, merged or fragmented events can be assigned an *underfill* or *overfill* error label if the ground truth event is not completely covered by a corresponding test event or test event "spills" over its boundary (even if by one gaze sample).

It is important to understand that EDEC does not include an explicit event matching procedure as such. It effectively computes certain statistics that describe the event detection quality. The process for computing these statistics, however, implicitly defines the correspondence between the events in the two label sources (the ground truth and predictions), which can be interpreted as a matching scheme (e.g. if event A is said to be fragmented into events B and C, then A is matched to both B and C at the same time, etc.). We

discuss a potential approach for explicit event matching based on the outputs of EDEC in Appendix E (however, fully developing, implementing such an event matching method is outside the scope of this review).

An example of all possible scoring labels assigned by EDEC is provided in Fig. 14. In this example, fixations GT1 and P1 overlap perfectly and are, therefore, labeled as a *correct* detection (cf. "C" in both GT1 and P1 boxes in the plot). Fixations GT11 and P8 also give an example of a correct detection, however P8 starts and ends earlier than GT11; therefore, GT11 is also assigned *underfill*, while P8 is marked with an *overfill* error label (cf. the subscript indices U and O of the correct detection mark "C" in the plot). Further, the ground truth fixation GT2 is misclassified as a saccade P2, while saccade GT3 is misclassified as a fixation P3. Therefore, both GT2 and GT3 are *deletion* errors, while P2 and P3 are *insertions*. Finally, the last two error types demonstrate how *EDEC* handles one-to-many and many-to-one matches. The ground truth fixation GT4 was split into two fixations P4 and P6 by falsely detecting saccade P5 in the test sequence, thus this data fragment is assigned *fragmentation* and *insertion* errors. And in the opposite case, the data fragment is labeled as a *merge* and two *deletion* errors where three separate smooth pursuit events GT5, GT7 and GT9 were classified as one long pursuit P7 because the algorithm missed the saccade GT6 and the fixation GT8.

Ward et al. (2006) conclude that their proposed event error scoring is a non-ambiguous and objective characterization of event-level errors that explicitly evaluate different sources of mistakes – timing, fragmentation, and merges – that are usually ignored by other event evaluation methods. However, in some scenarios *EDEC* can provide unintuitive or even ambiguous results. For example, in Fig. 15 the ground truth smooth pursuit event GT2 is split into two separate pursuit episodes P1 and P3 (fragmentation error). In addition, this data fragment also contains event timing errors – P1 starts earlier than GT2 while P3 ends later than the offset of GT2. However, *EDEC* does not register these timing errors[15]. Similarly, the algorithm merged the ground truth fixations GT3 and GT5 into one fixation P4 that does not entirely cover either of the two aforementioned ground truth events. The corresponding underfill errors do not get registered on the event level by *EDEC* thus potentially affecting subsequent evaluation of the algorithm performance.

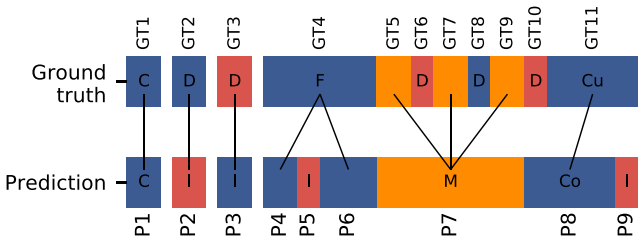

Fig. 14 Example of Event-driven error characterization (Ward et al., 2006). It partly replicates an example provided by Steil et al. (2018, Figure 4). Blue, red, and orange are fixations, saccades, and smooth pursuits, respectively. Black lines between the ground truth and predicted events indicate event relationships based on which error labels are assigned. C denotes correct detections, while *D* represent deletions, *I* – insertions, *F* – fragmentation, and *M* – merge errors. Lowercase letters *u* and *o* indicate event timing errors – overfill and underfill of the otherwise correctly identified events

An ambiguous situation for *EDEC* evaluation arises e.g. when the algorithm misses the ground truth saccade (event GT7 in Fig. 15) and detects a false one (P7) later on. In such case, the Ward et al. (2006) method reports a merge error because two ground truth fixations GT6 and GT8 appear to be detected as one fixation P6. *EDEC* also reports a fragmentation error, because saccade P7 appears to split the ground truth fixation GT8 into fixations P6 and P8. Note how the same minimally overlapping event pair, P6 and GT8, drives the reporting of both errors. In this particular example, the two ground truth and the two predicted fixations appear to be involved at the same time in both a fragmentation and a merge error. Steil et al. (2018, Fig. 4) grouped such cases into an additional error type *FM*. However, the described scenario can in fact happen because of the incorrect timing of the predicted saccade P7. Suppose the event detection algorithm uses noise filtering without delay compensation and, therefore, the onsets of all events are delayed just enough so that all short saccades do not overlap with their ground truth counterparts. *EDEC* evaluation would score all events to be fragmentation, merge, insertion and deletion errors. Modification of *EDEC* proposed by Steil et al. (2018) would report one fragmentation-merge error for the entire test trial plus insertion and deletion errors. Although such scenarios might be rare, users of *EDEC* must be cautious if many fragmentation and merge errors are reported and provide an interpretation of the reason behind these errors.

Algorithm evaluation using EDEC As mentioned already, event-driven error characterization was designed to only count detection errors and analyse event or segment level error tables (Ward et al., 2006, Tables 3 and 4). Because the method does not directly provide event matching, evaluation using confusion matrix and common classification performance measures (see "Confusion matrix-based measures") or direct analysis of event timing offsets (e.g. RTO and RTD, "Event quality metrics") is not enabled "out-of-the-box" (cf. Appendix E for a discussion of establishing an event matching approach based on EDEC).

[15] Overfill errors are only registered in *segment* evaluation using the algorithm provided in Ward et al. (2006, Fig. 2, right), but not on the *event* level (see also the definition of a segment in the beginning of "Event-driven error characterization").

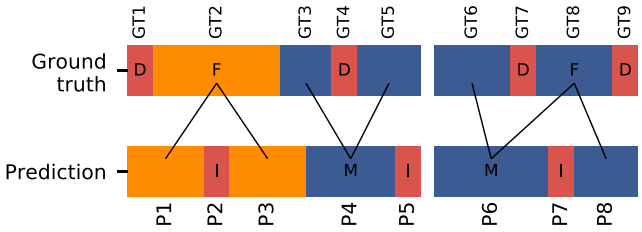

Fig. 15 Example of Event-driven error characterization (Ward et al., 2006) missing timing errors. Blue, red, and orange are fixations, saccades, and smooth pursuits, respectively. Black lines between the ground truth and predicted events indicate event relationships based on which error labels are assigned. *D* are deletions, *I* – insertions, *F* – fragmentations, and *M* – merge errors

Aside from the tables of either event or segment level errors, the output of EDEC can be used to calculate a unified metric CDI (Eq. 16 below; Steil et al., 2018):

$$CDI = C - D - I, \tag{16}$$

where C is the number of correctly detected events, D and I – numbers of deletions and insertions, respectively. As any other event detection quality metric (cf. "Confusion matrix-based measures"), this unified metric can be used for summarizing the algorithm's performance as well as to fine-tune its parameters or explore the performance across a range of settings (Steil et al., 2018, Fig. 5).

For a more detailed analysis, the output of the EDEC evaluation procedure can be visualized as an *Event Analysis Diagram* (Bulling et al., 2012, Fig. 8; Steil et al., 2018, Fig. 6). Event Analysis Diagram shows proportions of correctly detected events and errors with the respect to the total number of events in the ground truth and the test sequences.

Ensemble of matching schemes

Since different event matching approaches provide a different view of the algorithm's performance, the same metrics can be computed for several matching schemes, with averaging or in another way aggregating the scores. A similar approach was used in Startsev et al. (2019a) with a family of maximum IoU matching schemes with different IoU thresholds. The scores (F1 in that case) for different detectors were plotted against the IoU threshold values ranging from 0 to 1 (Startsev et al., 2019a, Fig. 4), allowing for a comparison of the performance over a range of the matching strictness criteria. These curves can, in principle, be integrated (with the IoU threshold as the variable of integration), resulting in an area-under-the-curve measure (similar to summarising an ROC with a single value as discussed in "Receiver operating characteristic curve").

Summary

In Table 4 we provide a summary of event scoring and matching approaches used in the literature when evaluating the performance of event detection algorithms. Like for classification performance metrics, we evaluate the *usability* (low, medium, or high) of each approach based on the theoretical analysis in this section. The properties we consider in this analysis correspond to the columns of the table: applicability to multiclass problems; directly enabling either confusion matrix or event timing analyses; whether the matching algorithm takes into account the "quality" of the match candidates; whether the algorithm is capable of producing only one-to-one matches, or whether one-to-many or many-to-one matching is enabled as well. We also consider the symmetry of the matching and whether any matching parameters need to be set.

Among the methods listed in the table, *Manual event scoring* (Friedman et al., 2018) and *Majority voting* (Hoppe & Bulling, 2016) are not technically event matchers and provide only a very limited view of where the algorithm fails to reproduce the ground truth event sequence. Overall, because of the complexity of the *Manual event scoring*, its non-deterministic nature, dependency on error thresholds, and very limited benefits (faster evaluation in some cases), we do not recommend using this method except to yourself acquaint with the dataset and the typical errors of the considered algorithm. The exception may be the case when one is certain that the same data and the same algorithm will never be tested again in the future. *Majority voting* approach, on the other hand, is very simple and easy to use. It is deterministic and enables a full confusion matrix-based score evaluation. However, it is prone to inflating scores as it does not, in any way, account for event fragmentation and is asymmetric event matching method. In addition, both *Majority voting* and *Manual event scoring* do not enable the detailed quantitative analysis of event timing, making their usability very limited.

The *Earliest overlap* (Hooge et al., 2018) and the *Overlap* (Hauperich et al., 2020) methods are very similar and, in fact, equivalent in a binary case. The only difference lies in the multiclass matching extension[16]: The former allows for matches of events of any class, while the later only matches events of the same class to one another. As such, the multiclass Earliest overlap matcher can potentially underestimate the algorithm performance, while multiclass Overlap matcher is subject to inflating the scores. Yet if one is interested in the subsequent event *timing* evaluation,

[16]Neither is part of the originally presented methods, and are based on loose interpretation of the wording their authors used to describe the binary-case procedure.

Table 4 Summary of event matching techniques used for eye movement event detection analysis. Manual event scoring (Friedman et al., 2018), Majority voting (Hoppe & Bulling, 2016), and EDEC (Ward et al., 2006) do not perform explicit event matching. However, they still establish a certain correspondence between events in the ground truth and the predictions, which is the "matching" we assess in this table. Note that Manual event scoring does not count correct detections, hence its results cannot be represented as a conventional confusion matrix. *Sample-level* is included in the table for reference and describes the properties of effectively treating each gaze sample independently, thus naturally obtaining a "matching" between sample-level labels in the ground truth and the predictions

	Multiclass	Enables confusion matrix analyses	Enables event timing analyses	Accounts for match quality	Allows one-to-many and many-to-one matches	Symmetric	Parameter-free	Usability
Sample-level	✓	✓				✓	✓	Low
Majority voting	✓	✓			✓†		✓	Low
Manual event scoring	✓					N/A		Low
Earliest overlap	✓‡	✓	✓			✓	✓	Medium
Overlap	✓‡	✓	✓			✓	✓	Medium
Maximum overlap	✓	✓	✓	✓		✓††	✓	High
Maximum IoU	✓	✓	✓	✓		✓††	✓	High
Window-based	✓	✓	✓		✓†			Low
EDEC	✓				✓	✓‡‡	✓	Medium

†Allows only one-to-many matches, i.e. one event in the reference sequence – typically ground truth – may be matched to multiple predicted events, but not the other way around

‡Original approach was only used in binary setting, however the method can be easily extended to multiclass cases

††The symmetry of these methods depends on their implementation, see "Note on implementation differences"

‡‡Ground truth and predicted event are given different labels to indicate conceptually the same detection error, otherwise evaluation is symmetric. N/A for the symmetry of manual event scoring refers to the fact that this method is inherently not reproducible, therefore it is difficult to discuss its symmetry.

both of these methods are to be preferred to the rest, albeit with a trick: Applying these matching techniques first in the *direct* temporal order (i.e. from start to the end of the sequence) – to assess the timing of event *onsets*, and then in the *reverse* temporal order (i.e. from the end to the start of the event sequence) – to assess the timing of event *offsets*. This repeated matching accounts for event fragmentation and merging and enables a more accurate estimation of event timing errors, compared to e.g. *Maximum overlap* or *Maximum intersection-over-union* matchers.

The next two methods – *Maximum overlap* (Zemblys et al., 2019b) and *Maximum intersection-over-union* (Startsev et al., 2019a) both account for the match quality (i.e. prefer higher-quality, in their respective definitions, matches to the lower-quality ones, regardless of the temporal order or match candidates). In fact, the procedures are identical apart from the definition of the metric used to quantify the quality of a certain match candidate. They were developed to work in both binary and multiclass settings, and enable a relatively accurate and intuitive estimation of labeling errors. Moreover these methods are symmetric (depending on the implementations, see "Note on implementation differences"), thus enabling non-ambiguous comparison of e.g. two expert coders without either assigning a special role of the "reference coder" to one of them or having to perform the comparison twice. While the simple principles of these event matches make them very easy to interpret from the prospective of their user, they also lead to a drawback of both strategies: When the goal of the performance analysis is the quantification not of the detection performance itself, but rather of event *timing*, one should keep in mind that neither *Maximum overlap* nor *Maximum intersection-over-union* account for event fragmentation and merge errors (only one of the ground truth events will be matched to one of the predicted events – based on the respective criterion of the matcher) and thus are prone to overestimating the timing errors (see e.g. Fig. 9).

Window-based event matching approach (Kothari et al., 2020) was developed to account for event fragmentation and event timing quality. However, it exhibits some unintended behavior and is, in its current form, susceptible to misinterpreting error types, or even hiding some errors from the analysis during its label alignment stage. Because of these complications, we do not implement this promising method in our code package and exclude it from empirical analyses in the next section.

Event-driven error characterization (EDEC) (Ward et al., 2006) is the only method that directly accounts for event

fragmentation and merge errors and enables a very detailed analysis of the detection errors. The method is parameter-free and symmetric. Unfortunately, it does not include an *event matcher* as such, and using the ideas of EDEC in combination with traditional confusion matrix-based or timing error analyses requires a non-trivial extension of this method (cf. remarks in Appendix E) in order to transform it into a general-purpose event matcher (such as the others reviewed in this section). This, naturally, limits its usability in algorithm development process. In addition, EDEC marks the detected events with and "underfill" or an "overfill" error label if the event borders are not perfectly aligned with those in the ground truth (i.e. misalignment by a single sample is sufficient). For high-frequency eye tracking recordings, this will likely mean that in practice virtually all correctly detected events will receive a marking signifying a timing error, reducing the level of insight gained from such analysis.

Interaction between the performance metrics and event matchers

In "Evaluation metrics" and "Event matching methods" we list a number of metrics and methods used for sample- and event-level detection performance evaluation. Some of these methods (e.g. comparison of fixation and saccade durations, behavioral scores, etc.) are in effect sanity checks for the algorithm outputs (and not measures of its performance), and quite a few of these do not need ground truth annotations. Other evaluation strategies, in particular those directly evaluating the quality of the algorithm's detections, strictly require ground truth and can enable descriptive and concise evaluation that is relevant for for both the user and the developer of the algorithm.

The multitude of the tools available for designing an evaluation pipeline is especially prominent in case of the evaluation on the level of whole eye movement events, which can in principle combine any event *matcher* with any *metric* to quantify the event-level performance of an algorithm. In this section, we aim to empirically demonstrate the effect of combining different metrics and matchers in order to observe systematic trends and differences. Here, we focus on event-level evaluation, and on *detection* quality metrics specifically ("Levenshtein distance" and "Confusion matrix-based measures"). We include sample-level evaluation with the same metrics for reference, to demonstrate the practical differences between sample- and event-level evaluation.

We excluded two event matching strategies from this analysis – Window-based (Kothari et al., 2020) and EDEC (Ward et al., 2006) – since the analyses we undertake mainly focus on combining the event matchers with various *detection* quality measures, and those typically summarize the confusion matrix. The two excluded matchers, however, account for fragmentation and/or merge errors, and there are multiple ways to incorporate such one-to-many and many-to-one matches into the confusion matrix, and this examination is outside the scope of this review. Moreover, Window-based matching would be additionally more challenging to analyze or re-implement due to the likely unintended behavior of the matcher (specifically in terms of label alignment, cf. "Window-based matching". EDEC is, nevertheless, implemented in our codebase and can be tested separately.

All the experiments in this section were carried out using the codebase provided with this article, thus illustrating the versatility and multitude of the available evaluation approaches and scores to be used for the future researchers and algorithm developers.

Method

The data we use for this analysis is the Hollywood2EM dataset (Agtzidis et al., 2020) with the ground truth fixations, saccades, smooth-pursuits, and noise annotated. Annotations were performed by pre-segmenting monocular 500 Hz gaze data with the I-VVT algorithm (Komogortsev & Karpov, 2013), followed by first a paid student, then an expert annotator adjusting incorrectly detected events. Around 59% of the samples were labeled as fixations, 10% as saccades, almost 27% as pursuits (see Table 1); the remainder we relabeled to undefined samples. In this analysis we use the "test" subset of the Hollywood2EM dataset – i.e. ca. 92% of all data.

In addition to the ground truth labels, the dataset includes events as detected using 15 publicly available event detection algorithms (see Agtzidis et al., 2020, Table 1 for a full list). Similarly to the ground truth data, we relabeled the classes other than fixations, saccades, and pursuits as "undefined" in order to provide a uniform evaluation set-up for the ensuing analysis.

For each eye tracking recording we calculate the length-normalized sample- and event-level Levenshtein distances, sample-level confusion matrix-based performance scores, as well as perform Majority voting evaluation. To enable direct event-level performance score calculation, we also perform several variants of event matching: Earliest overlap, Overlap, Maximum overlap, and Maximum IoU (with and without an extra match criterion). We then calculate a subset of the detection quality metrics listed in Table 3 for both sample- and event-level evaluation, in both the multiclass and binary (using binary remapping) settings. We excluded the ROC AUC metric from this analysis since in the absence of some confidence score for each of the predictions it does not provide insights into model performance beyond other

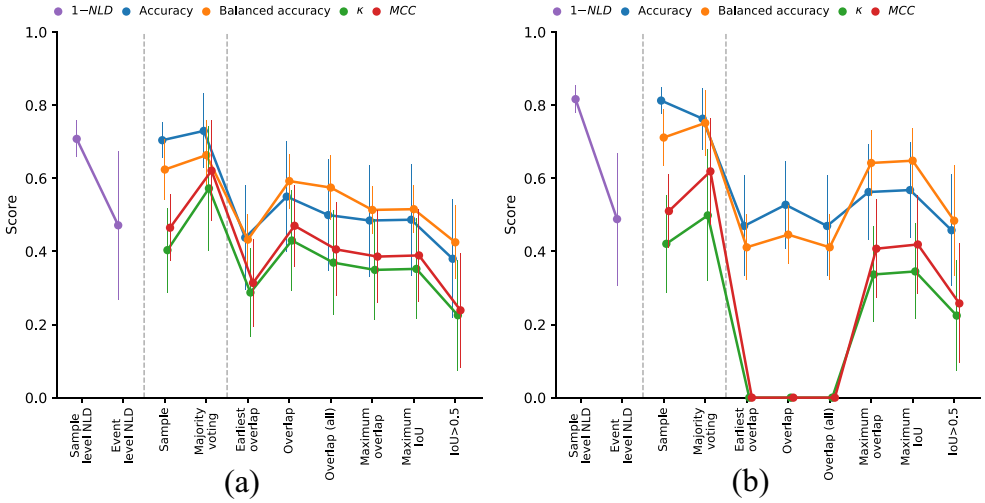

Fig. 16 Average **multiclass** (16a) and **binary-setting** (16b) sample- and event-level Levenshtein distances (NLD; presented as their complements, $1 - NLD$), accuracy, balanced accuracy, Cohens's Kappa (κ), and Matthews correlation coefficient (MCC) scores of 15 event detection algorithms. Event-level scores were calculated using the Majority voting, Earliest overlap, Overlap (two modes), Maximum overlap, and Maximum IoU event matching methods. Error bars correspond to ± 1 standard deviation. Binary-setting scores are averaged over binary fixation, saccade, and smooth pursuit detection. Figure best viewed in color

metrics (see reasoning in "Receiver operating characteristic curve"). For the other metrics, we present our result in two groups: First, for the metrics that can be computed in *both* multiclass and binary setting (Levenshtein distance, accuracy, balanced accuracy, Cohen's kappa, MCC), then – for those metrics that can only be applied in a binary setting (precision, sensitivity, specificity, F1-score, JI). In the binary setting, we average the scores of each of the metrics for the three classes of interest - fixations, saccades, and smooth pursuits – to represent the overall performance with one value. To ensure a fixed range of the computed metrics, all the scores were clipped to the [0, 1] interval (meaning that event-level Levenshtein distances cannot exceed 1, and negative Cohen's kappa values were treated as 0), while the metrics that could not be calculated are ignored.

Typically, in a research paper that e.g. proposes a new detection algorithm, the scores for the different algorithms would be analysed separately in order to understand which of the compared eye movement detectors performs best. In this work we are, in contrast, interested in the systematic analysis of the interaction between the evaluation approaches and the metrics computed. Therefore, we aggregate the scores over the full set of recordings and all of the 15 tested algorithms. Since the majority of the algorithms compared in (Agtzidis et al., 2020) are relatively old, threshold-based approaches, and, moreover, five of them do not detect pursuit at all, we do not expect their average performance to be particularly good, especially in pursuit detection. We thus expect the metrics and evaluation approaches that adequately quantify the algorithms' performance to also reflect this. Results below can, therefore, be interpreted as the performance of a hypothetical "average", and in this case by-design mediocre, event detection algorithm tested with in a large set of evaluation settings. This interpretation enable us to compare the different *evaluation settings* to one another, instead of comparing the various *algorithms*.

In the binary setting, we choose the evaluation mode where we ignore unmatched negatives (see "Multiclass vs. binary evaluation"). The two other possible choices are to consider unmatched negative events as errors or correct predictions (true negatives). However following analysis of Zemblys et al. (2020) we deem ignoring unmatched negatives as the approach that best reflects the algorithm performance in the most reliable and intuitive way.

Results

Figure 16a shows the aggregated results of sample- and event-level normalized Levenshtein distances (NLD), accuracy, balanced accuracy, Cohens's Kappa (κ), and Matthews correlation coefficient (MCC) scores. Figure 16b presents the same statistics for binary-setting evaluation.

Note that for illustration purposes and easier interpretation, Levenshtein distances are represented by their complement in the plot (i.e. one minus NLD). This ensured the uniform interpretation of the value 1.0 (i.e. as "perfect prediction") for all the metrics in the figure.

For all metrics but the Levenshtein distances, the evaluation is performed in combination with different

evaluation approaches: either on the level of single gaze samples, or using one of the seven event matching strategies we test here: Majority voting (Hoppe & Bulling, 2016), Earliest overlap (Hooge et al., 2018), two modes of Overlap (Hauperich et al., 2020), Maximum overlap (Zemblys et al., 2019b), Maximum intersection-over-union (Startsev et al., 2019a), and the variation of the latter with a match criterion of IoU$>$0.5.

The following parts of this section touch on different observations we made based on the obtained statistics, and further explores several aspects of the interaction between various evaluation strategy components, such as the conspicuous zero scores for Cohen's kappa and MCC in Fig. 16b observed for binary event-level evaluation with Earliest overlap or Overlap event matcher ("Event-level performance evaluation") or substantially higher, compared to other methods, Majority voting evaluation scores ("Sample-level and Majority voting evaluation").

Accuracy

Independent of the evaluation scheme (sample- or event-level using different event matching methods) the accuracy and the weighted accuracy metrics yielded the highest scores in our experiments, followed by the MCC and κ. In terms of relative difference, the average accuracy score is between 50 and 100% above the other metrics in the overwhelming majority of the evaluation settings, likely illustrating the accuracy paradox, and thus inflating the scores of the algorithms that take advantage of the majority event class. Over half the samples in the dataset are fixations and, in addition, 5 out of 15 algorithms in our evaluation detect only fixations and saccades but not smooth pursuit. Consequently, the dataset-wise probability of an arbitrary sample being labeled as a fixation is considerably higher compared to that of the other event types. Indeed, the dataset-wise sample-level confusion matrix (which accounts for the labels predicted by all 15 algorithms) in Fig. 17 shows that 89% of the ground truth fixation samples were labeled as fixations; moreover, 24% and 69% of the saccade and pursuit samples, respectively, were also mislabeled as fixations. Simply put, in our evaluation the accuracy score mainly depends on the performance of detecting fixations – the majority class, while incorrectly detecting other events has minimal impact. This demonstrates that a relatively high score (cf. accuracy versus MCC and κ in Fig. 16a) does not mean a good performance of the algorithm.

The inflated accuracy scores are even more prominent in binary (per-event) evaluation (visualized in an aggregated form in Fig. 16b). For example, average binary sample-level accuracy scores for fixation and pursuit detection – 0.73 and 0.76 respectively – are similar to the average multiclass accuracy score of 0.70, while binary sample-level saccade detection accuracy in this dataset is 0.95. Averaging these three per-class binary scores yields an overall binary-setting sample-level accuracy to 0.81 – considerably higher than in the multiclass setting. This further illustrates that a seemingly high accuracy score is not a sufficient indicator of good performance, and could give a false sense of the algorithm performing well.

Balanced accuracy is designed with the intention to avoid such inflated estimates on imbalanced data, yet is still yields fairly high scores on our dataset. Moreover, on average, the multiclass event-level balanced accuracy is actually higher compared to non-weighted accuracy in our evaluation. Similar to the binary evaluation, balanced multiclass accuracy depends on the interplay between the balance of event types and the performance of correctly detecting the events of each type separately. However, when used in combination with event matchers that are present in the literature, it is prone to overestimating (or in some cases underestimating) the performance, mostly because of how unmatched events are handled in event-level evaluation (see in-depth explanation in Appendix F).

Sample-level and Majority voting evaluation

All sample-level multiclass metric scores, just like the event-level ones obtained using the Majority voting, indicate high agreement between the ground truth and the predictions, while the event matchers specifically designed to find two-way correspondences in the ground truth and

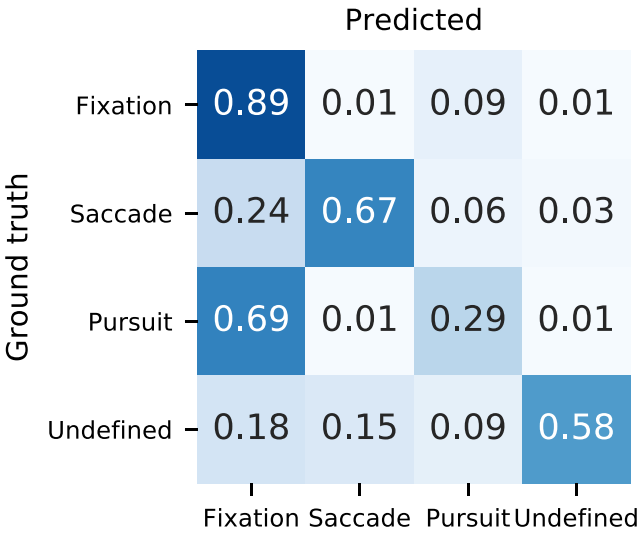

Fig. 17 Overall sample-level confusion matrix for the complete dataset, i.e. summarizing the predictions of all tested algorithms. Normalized along the rows to represent proportions of correctly and incorrectly labeled samples

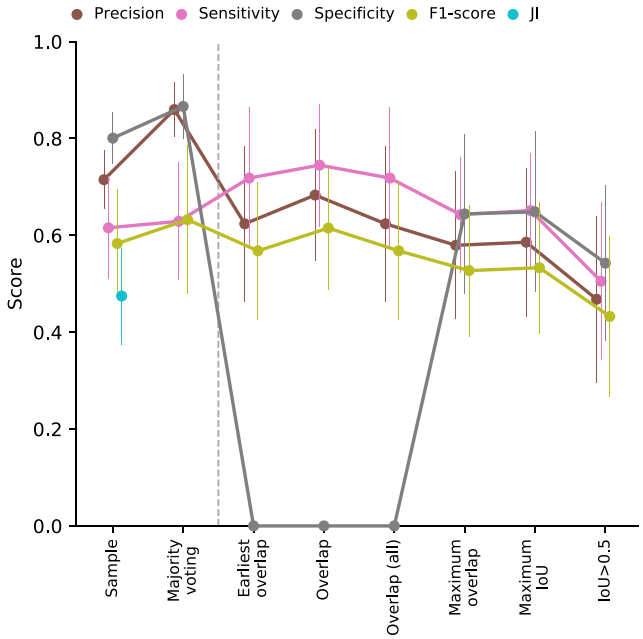

Fig. 18 Averages of binary metrics of 15 event detection algorithms, evaluated in a binary setting. All scores represent average performance of binary fixation, saccade, and smooth pursuit detection. Event-level scores were calculated using the Majority voting, Earliest overlap, Overlap (two modes), Maximum overlap, and Maximum IoU event matching methods. Error bars span ±1 standard deviation

predicted events result in substantially lower scores across the board (see Figs. 16a and 16b). These high scores are, however, overly optimistic performance estimates and are, to a large extent, affected by the class imbalance in the data.

Both discussed evaluation strategies also fail to account for merged and fragmented events: As long as the majority of the sample labels in the two sequences agree, the evaluation scores will be high and give a false sense of good performance (see "Sample-level evaluation" and "Majority voting"). This effectively means that two event detectors that deliver qualitatively widely different predictions can easily obtain the same scores with both sample-level and Majority voting-based evaluation strategies: E.g. an algorithm that detects all events with a one-sample offset can score the same as (or very close to) another detector that fragments every single event in the ground truth with one wrong-class sample in the middle. A similar example was given in Fig. 2 to motivate event-level evaluation.

Moreover, Majority voting effectively ignores some of the incorrect predictions if the majority of the samples in the prediction sequence agree with a certain ground truth event: Note how most of the metrics, including the binary ones in Fig. 18, are higher when computed via Majority voting compared to the sample-level scores.

Event-level performance evaluation

Event-level performance scores in Fig. 16a and b present the empirically observed differences between various event matchers, and support some of the conclusions about their pros and cons that we discussed in the respective sections. The magnitudes of the differences in the average statistics in these plots effectively express, in this dataset and for the compared algorithms, the frequency of the situations where the *theoretical* differences that we observed in the corresponding parts of "Event matching methods" occur *in practice*: For instance, while Maximum overlap and Maximum IoU have a distinct type of a scenario when whey would produce differing outcomes (see Fig. 9), the *practical* differences, at least on average, are small. Meanwhile, applying an IoU threshold of 0.5 for a Maximum IoU matcher leads to a drastic change in the subsequently computed metrics, meaning that a lot of the matches registered by the Maximum IoU matcher had an IoU below this threshold.

In the *multiclass* setting (Fig. 16a) the Earliest overlap method produces the lowest performance estimates (except for the thresholded Maximal IoU matcher – "IoU>0.5") for each of the metrics. This result illustrates and supports the observation that, as described in "Earliest overlap matching", Earliest overlap is likely to fail at intuitively matching corresponding events (especially in its multiclass extension), and an evaluation pipeline based on this matching technique is, therefore, prone to underestimating the algorithm performance.

The multiclass-setting extension of the Overlap matcher is similar to the Earliest overlap, except it only allows matching events of the same class. Moreover, as it was originally used by Hauperich et al. (2020), only one match from the overlapping events is taken into account for metric computation, and the rest of the events that are part of a merge or fragmentation are ignored (see "Overlap matching"). Inevitably, compared to the other methods that include all unmatched events in the subsequent agreement quantification, these choices lead to substantial performance overestimation. We also ran the Overlap matcher in a mode where all events are considered for further evaluation pipeline steps (i.e. unmatched events in a fragmentation are not suppressed). Needless to say, the resulting scores (see "Overlap (all)" in Fig. 16) are lower compared to those obtained via the Overlap matching method, and much closer to those of the Maximum overlap and Maximum IoU methods. The theoretical drawback of preferring a one-sample overlap of two same-class events over a much larger overlap with an event of a different class still remains even for the "Overlap (all)" matching strategy. In practice this would mean registering such potential unintuitive matches as correct detections, thus inflating the evaluation scores.

This aligns with the observation that the scores obtained with this matcher are higher compared to e.g. Maximum overlap.

In the *binary* setting (Fig. 16b), the Earliest overlap and the Overlap method (importantly, in the mode when all unmatched positive events are considered – i.e. the "Overlap (all)" variant) are equivalent – cf. identical respective scores in Fig. 16b, also Fig. 18 below. Note that both approaches exclude negative events from the matching procedure, i.e. *all* negative events remain unmatched. Since we ignore unmatched negatives in our evaluation, and neither the Overlap nor the Earliest overlap matcher produces any *matched* negatives, the resulting binary accuracy and balanced accuracy scores are lower compared to those obtained via Maximum overlap or Maximum IoU matchers (since the latter two matchers can produce matched negative events that count towards increasing the accuracy score). In addition, without any correct negative matches the κ and MCC scores become less or equal to 0, effectively making evaluation using the Earliest overlap or Overlap matchers in combination with the κ and MCC scores impossible. If the aforementioned combination is nevertheless desirable, unmatched negatives can be converted to true negatives (see Fig. 23 in Appendix G, also Zemblys et al., 2020).

The Maximum overlap and IoU matchers are very similar in theory, and result in nearly identical scores regardless of the examined metric. For some of the tested algorithms using the Maximum IoU matcher yields higher scores, likely illustrating that it can deal with some cases of unintuitive matching characteristic for Maximum overlap. In addition, the Maximum IoU matcher enables setting a threshold for match quality (we tested its version with an IoU of over 0.5 required for a pair of events to be considered as a match). Naturally, this degrades all the performance scores (cf. "Maximum IoU" vs. "IoU>0.5" in both parts of Fig. 16), reflecting the more stringent evaluation set-up.

Normalized Levenshtein distance

Normalized Levenshtein distance (NLD) is different from the other metrics and evaluation approaches provided in this section in a sense that it can take advantage of matching sub-sequences in the two compared sample or event data streams, even if these sub-sequences are not aligned in time, or are seconds apart. Since typical event-level transitions between eye movement types are relatively repetitive, and consist in a large part of alternating fixations and saccades, this may present a practical problem as well as a theoretical one (Startsev et al., 2019). As a result, in some corner cases NLD can considerably overestimate the algorithms' performance.

As can be observed in Fig. 16, sample-level NLD seems to indicate a much higher prediction quality level compared to other metrics (even for sample-level evaluation), and additionally has a much lower standard deviation. The latter fact is at odds with the algorithms we compare having a wide range of suitability for our data (several algorithms ignoring one of the eye movement types of interest altogether), and likely indicates that choosing between several eye movement detectors based on sample-level NLD would be challenging. Event-level NLD, however, seems to provide scores comparable to other metrics on average, albeit with noticeably higher standard deviation of the metric.

To assess how similar NLD performance estimates are compared to other methods we used linear modeling – $\mathrm{lm}(NLD \sim method)$, where $method$ is a combination of an entity matching approach (samples or events using various event matchers) and an evaluation metric. A fit with a highest coefficient of determination (R^2) is considered to identify the most similar performance estimates. In both multiclass and binary setting the sample-level NLD is nearly identical to the sample level accuracy – linear models in both cases are highly significant with $r^2 \approx 1$. This is consistent with the fact that the "editing" of all the incorrectly predicted sample-level labels would directly correspond to a normalized edit distance of $1 - accuracy$. Although the computation of NLD can result in a more elaborate set of editing operations, for a non-corner-case situation it is unlikely to be very different. Multiclass event-level NLD is most similar to the Earliest overlap accuracy score ($r^2 = 0.84$, $p = 9.3\text{e}{-5}$), while in the binary setting it is most similar to the accuracy score calculated using the Maximum IoU matcher ($r^2 = 0.9$, $p = 5.9\text{e}{-6}$). These strong empirically observed similarities indicate that both sample- and event-level NLD suffer from the same practical problems as the accuracy measure (without class balancing) – cf. theoretical observations regarding e.g. class imbalance in "Confusion matrix-based measures"), as well as the practical implications in "Accuracy" above.

Binary metrics

Figure 18 presents the evaluation results for the metrics that can only be computed in the binary setting: precision, sensitivity, specificity, F1-score, and Jaccard index (JI). Only sample-level JI is calculated since event-level JI would refer to the event quality rather than event detection performance metric (see "Event quality metrics"), which evaluates an entirely different aspect of prediction quality, not comparable to the other metrics here.

Sample-level precision (0.71) and specificity (0.8) indicate a fairly good average performance of the evaluated algorithms. Scores are almost identical to the binary accuracy and balanced accuracy scores (0.81 and 0.71

respectively, see Fig. 16b), which we deemed to be overly-optimistic performance estimates. The high sample-level precision score of 0.71 indicates that most of the samples classified as a certain event type indeed belong to that type in the ground truth. This is relatively easy to achieve in an unbalanced dataset – a random sample classifier has the precision corresponding to the proportion of a certain event type (see Table 6 in Appendix D), i.e. anything better than random will score even higher. Moreover, since saccade events are relatively easy to detect, as long as the algorithm does not predict a lot of false saccades, its saccade detection precision will be high (on average 0.83 in our evaluation), thus inflating the average precision score.

The average sample-level specificity is also inflated because the algorithms tested here tend to, on average, predict more of the majority (fixation) class samples: Specificity evaluates the performance of detecting negative (i.e. other than the currently evaluated) class, and therefore it is relatively low for fixations – 0.51 – as the negative-class samples (i.e. non-fixations) are often still predicted as fixations. However, the sample-level specificity for saccade and pursuit detection evaluation is 0.98 and 0.91 respectively, resulting in a high sample-level specificity score of 0.8 on average. In contrast to specificity, the sensitivity metric evaluates the performance of detecting positive class and is relatively high for fixations (0.88) compared the minority classes – saccades (0.68) and pursuits (0.28) – thus resulting in a comparatively low class-average sample-level sensitivity score of 0.62.

F1-score is the harmonic mean of precision and sensitivity: For sample-level scores, for instance, even though precision is relatively high (0.71), the F1-score of 0.58 is closer to sensitivity (0.62), i.e. the lower of the two. Note that in any individual case the F1-score lies between the between the precision and sensitivity, however with all the averaging that we do in this analysis it looses this property (since harmonic mean is closer to the lower of the two scores, and precision and sensitivity are not always ordered in the same way). Finally, the sample-level JI score indicates rather poor performance of 0.47, which is on par with the sample-level binary κ and MCC scores (0.42 and 0.51 respectively, see Fig. 16b) – the scores that account well for class imbalance.

When considering event-level evaluation with binary metrics in Fig. 18, it is again valuable to examine the systematic differences between different event matchers. Similarly to the case with multiclass metrics, there is little difference in the average scores obtained via Maximum overlap and Maximum IoU matchers, but introducing a match threshold in the latter matcher ("IoU>0.5" in Fig. 18) considerably lowers all performance scores. Interestingly, however, precision, sensitivity, and F1-score are *higher* for the Earliest overlap and Overlap event matching approaches compared to the Maximum overlap and Maximum IoU. This is the opposite of what we observed in Fig. 16b for all of the tested multiclass metrics – accuracy, Cohen's kappa, MCC – applied in a binary-setting evaluation procedure: There, the scores for Maximum overlap and Maximum IoU were substantially higher. This reversal is due to the fact that the evaluation measures in question here (i.e. precision, sensitivity, and F1-score) only account for two types of detection errors: missed positive-class detections (false negatives) and falsely detected positive events (false positives). By their design, Earliest overlap and Overlap matchers only allow *positive* events to be matched, thereby reducing the number of false negatives (by preferring even poorly aligned same-class matches to any different-class ones, however well aligned) and increasing the true positive count. Maximum Overlap and Maximum IoU, on the other hand, allow for any events to be matched, for example the ground truth positive event can be matched with the predicted negative event and vice versa, thus in many cases increasing the number of false positives and false negatives while decreasing the true positive count. This leads to lowered scores for the metrics we consider here (see respective formulas for precision – Eq. 7, sensitivity – Eq. 8, and F1-score – Eq. 10).

Similarly to the results of multiclass metrics in the binary setting (Fig. 16b), specificity – a metric that evaluates the performance of detecting negative class events (see Eq. 9) – cannot be calculated for the Earliest overlap and Overlap matchers since they do not produce any matches between negative-class events, therefore the number of true negatives cannot be assessed. Calculation of event-level specificity scores hence is only enabled when using the Maximum overlap and Maximum IoU matchers as they allow any events to be matched, including negative-to-negative.

Summary, recommendations, and discussion

In this section, we first summarize the main recommendations throughout this review, focusing on the evaluation that is directly quantifying the performance of an eye movement detector (thus excluding e.g. application-driven evaluation), and doing so in relation to ground truth labels, for example expert annotations. Evaluation procedures not requiring ground truth are presented in the paper as well ("Evaluation based on eye movement metrics", "Evaluation based on stimuli parameters" and "Application-based evaluation"), but the review did not delve as deeply into these.

We also bring the reader's attention to several decisions that can heavily influence the results of the evaluation, and do not have a trivial answer. These possible solutions have their own benefits and drawbacks, and cannot be chosen

without the context in which the evaluation is supposed to take place.

Ensuring fair evaluation

The cornerstone of a sound evaluation of any eye movement event detection algorithm is always separating the set of the eye tracking data that is used in the development and optimization process from the set where it is tested. Otherwise one risks *overfitting* the algorithm to a particular dataset, which will lead to reduced generalization and over-optimistic performance estimates.

Ideally, the training and the test sets would have neither overlapping observers nor stimuli. When creating a dataset from scratch, consider recording two entirely disjoint sets from the beginning – with two groups of participants and two groups of stimuli, – thus ensuring both the efficiency of the data collection process and a very clean separation of the data subsets. When working with an existing dataset, based on the findings in (Startsev et al., 2019a) we recommend focusing on keeping the *stimuli* sets disjoint between the two sets. This is especially applicable when dealing with heavily stimulus-dependent properties of eye movements: smooth pursuit or other eye movements with properties heavily influenced by the stimulus parameters – e.g. saccades amplitudes pre-determined by the synthetically generated stimuli.

If data separation on the recording level is not feasible, consider splitting the dataset in a time-wise manner, i.e. uninterrupted parts of the recording assigned to different sets, again ensuring that stimuli do not overlap in the two sets (see more in Appendix C).

Choosing the evaluation protocol

An evaluation protocol in this context consists of selecting the dataset(s) used in the evaluation and comparing the algorithm(s) against adequate baselines.

Datasets

In Table 1 we provided a list of properties and short descriptions of publicly available datasets with expert-annotated ground truth that we encourage to use for developing and testing new algorithms. The diversity and volume of readily available data enable extensive evaluation of the algorithm generalization capabilities, making cross-dataset evaluation readily accessible to the algorithm developers, in addition to being highly desirable. Furthermore, using an existing dataset instead of investing effort and time into recording and annotating a new one facilitates an easy way of comparing the proposed approach to other algorithms, since annotated datasets were likely already used to develop and test event detectors. The performance scores for these are typically available in the corresponding papers. It should be noted, however, that unless the evaluation procedure is followed exactly, the published score and the one obtained using a new approach are not comparable (see "Interaction between the performance metrics and event matchers" where we show that different evaluation approaches lead to large deviations in the obtained performance scores). Thus, prefer datasets that come equipped with either the evaluation tools or the labels produced by a number of other event detection methods so that they can be evaluated in the same pipeline as the newly developed approach.

On the other hand, annotated eye tracking datasets still by far do not offer a complete coverage of all realistic use cases for eye movement detection systems, therefore recording and annotating a new dataset is a valuable contribution. Moreover, it can sometimes present the only option: E.g. if annotated data of a certain eye tracker or at a certain frequency are not available. Annotating a new dataset is extremely time consuming (though there are options to significantly speed up the annotation, albeit with some caveats – more details in "Eye tracking data collection and annotation") and yet does not ensure *perfect*, "gold standard" labels (Hooge et al., 2018). Some studies successfully used synthetic data for both developing and testing an event detection algorithm (Otero-Millan et al., 2014; Fuhl et al., 2018; Zemblys et al., 2019b), though we still strongly recommend to focus at least part of the *evaluation on real eye tracking data*, this being the target application domain of the developed algorithms.

Synthetic data could be used to simulate the properties of a certain eye tracker in the annotated signal of another. For instance, without having the annotations for an eye tracker A, the annotated data of eye tracker B could be modified to resemble the properties (level of noise, etc.) of the device A, for example in a manner similar to neural network-based style transfer for images (cf. Gatys et al. 2016 and subsequent works in this domain). A much more straightforward way to achieve a similar goal is directly adapting the signal by the means of resampling, mixing in a certain level or patterns of noise, or filtering. These can be derived from the "target" set-up either empirically or via theoretical estimates. In principle, any approach to simulate a different eye tracking signal distribution can serve as means of testing the algorithm's generalization ability, and can be employed just as any other independent real datasets.

Comparison

In order to put the results of the developed eye movement event detector in appropriate context, we recommend

looking for publicly available detectors that were developed in a setting that is as close as possible to the one in the tested condition, as algorithms often do not work well in conditions for which they were not designed (Andersson et al., 2017): For instance if the eye tracking data involve looking at moving objects, using a traditional fixation detection method originally developed for reading studies is very unlikely to yield satisfactory results. Moreover, in general using a one-threshold method to solve an inherently multimodal problem (Steil et al., 2018) will not provide a good baseline.

When the algorithms to which the developed method is being compared are considerably simpler computationally (e.g. consist of several thresholding rules, similar to I-VT, I-VVT, etc.), it could be reasonable to estimate their best-case performance, effectively letting their parameters overfit to the test data. In addition to making a stronger case for the superiority of the developed method, this also simplifies the optimization for these simple algorithms, eliminating the need for cross-validation, etc.

Another useful approach is evaluating the robustness of the algorithm's behavior to e.g. different noise levels in the eye tracking data, or other parameters, in order to understand the algorithm's sensitivity to various properties of its inputs. Regarding noise-driven evaluation, we refer the reader to Niehorster et al. (2020), where the authors demonstrated that the simplest approaches to simulating noise may not be sufficiently representative.

Choosing an evaluation method

Available and applicable algorithm evaluation methods depend on a number of factors where the most important variable is the availability of annotated eye movement events that may be used as a ground truth. We argue that without it the evaluation of the eye movement detector's performance is essentially a "sanity check" for the detected events, but hardly a precise measure of the performance of the algorithm. For example, an algorithm that fails to detect every second fixation would still yield a fixation duration distribution that is comparable to the literature. Nevertheless, there might be many reasons why evaluating eye movement event detector without using the annotated events is desirable. For example, the purpose of the evaluation may simply be fulfillable without it (e.g. comparing the distributions of fixation durations produced by two algorithms). One could also argue that the ground truth precisely reflecting the state of the visual system either does not exist or cannot be sufficiently accurately approximated by expert annotations, hence comparing algorithms' to expert coding is not worth the labour-intensive manual annotation. In some cases synthetic stimuli and instructions to the observers can already sufficiently define the eye movements, and therefore the stimuli properties can be used for the evaluation. Sometimes it may even be impossible to record the eye tracking signal for the evaluation (e.g. due to eye tracker license restrictions). The overview of evaluation methods that can be applied in such situations can be found in the first three parts of "Evaluation methods".

The comparison between the ground truth annotations and the output of an algorithm can be performed on the level of samples (comparing the labels of each individual gaze point in the two sources) or events (comparing the timing and the labels of uninterrupted sequences of samples with the same label). For sample-level evaluation the correspondence between gaze samples is already established: The compared sources of eye movement labels – i.e. the ground truth and the output of the algorithm – are inherently describing the exact same set of the considered entities (the gaze samples). For the evaluation in terms of whole eye movement events, however, some correspondence has to be established between the two sets before many of the traditionally applied forms of quantitative evaluation can take place. We call this process *event matching* and describe various approaches developed to perform it in great detail in "Event matching methods". The properties of the matching technique can influence not just the "sensibleness" of the resulting evaluation score, but even the possibility of subsequently quantifying a particular type of differences in the two sets of labels. Although many works in the literature to date report sample-level scores only, these do not present a reliable performance estimation and are strongly affected by the majority class. Based on both theoretical considerations (cf. e.g. "Sample-level evaluation") and practical event matcher analysis in "Interaction between the performance metrics and event matchers" we conclude that *event-level* evaluation should be preferred on the whole as more adequately representing the intuitive understanding of the eye movement detection algorithm quality.

When choosing the evaluation method it is very important to take the prospective end "user" of the evaluation results into account. Some researchers might only be interested whether one or the other algorithm works better for their application, or whether the considered eye movement detector is, in some sense, "good" or "better than another" in term of e.g. reproducing saccade main sequence, while disregarding how many events the algorithm failed to detect. Such an evaluation effectively treats the detector as a black box. The particular ways to improve the algorithm are typically not directly produced, although, of course, can be somewhat inferred or guessed at.

Another group of users, meanwhile, might be interested in e.g. how many saccades were missed or mislabeled as other events, and might want to know where exactly in

the data these errors occur. This evaluation prospective is mostly the domain of event detection algorithm developers, whose main interest (aside from showcasing the superiority of their method) is pinpointing the main sources of the algorithm's errors, with the purpose of improving it in the long run.

With the advent of machine learning-based event detection algorithms (Bellet et al., 2019; Startsev et al., 2019a; Zemblys et al., 2019b, to name a few), it has also become relevant to motivate the choice of an evaluation approach not just by differentiating between a handful of algorithms (e.g. in the form of evaluation results tables): Suitability for guiding the training of an algorithm or optimizing its parameters becomes crucial in the absence of a human expert interpreting the results. This purpose puts the ultimate weight on the fairness and conciseness of the evaluation, since all of the relevant aspects of the algorithm's predictions need to be effectively summarized by a single number.

Some aspects of different evaluation methods and measures (especially on the level of events) contribute to their interpretability, bias, and suitability for various purposes and testing scenarios. While the *fairness* of the evaluation (which can be interpreted as not misleading into a false sense of good and reliable performance, either in the absolute sense or when comparing several different approaches to one another) is relevant for any use case, the priorities of evaluation *descriptiveness* and *conciseness* vary between different "target groups". With the eye movement detection algorithm developers' perspective being central to this review, we attempt to realistically balance our recommendations between being descriptive (i.e. easily enabling a clear understanding of what mistakes an algorithm makes) and concise (i.e. not requiring pages of statistics to report) in such a way that it would be feasible to incorporate in a research publication.

Comparing algorithms that detect different sets of eye movement types

When choosing an evaluation method for a particular comparison between a set of different algorithms, special attention needs to be paid to whether the selected approach will fairly reflect the performance of the evaluated eye movement detectors *in comparison to one another*. These considerations thus go beyond the descriptiveness and other properties of the evaluation method in the context of a single considered detector. In particular, it is important to consider whether all of the compared algorithms detect the same set of eye movement event types. If they do not, different evaluation methods will differently reflect this, affecting the outcome of the comparison.

In principle, if not the same sets of eye movement types are detected by all the compared algorithms, binary evaluation approaches and measures that quantify the detection quality per eye movement type should be employed. And this should, indeed, be sufficient for sample-level evaluation. Event-level evaluation, however, is more intricate, as event matching is also affected by detecting or not detecting various eye movement types. As explained in more detail in "Multiclass vs. binary evaluation", we recommend using *sample-level binary remapping prior to event matching* to ensure comparability between different algorithms, even though it is neither the most concise, nor the most descriptive approach (since it is difficult to keep track of what exact misclassification took place, or whether the event was missed altogether). In contrast, to obtain a detailed insight into the prediction patterns of a particular algorithms (or to compare the algorithms that all detect the same set eye movement types), multiclass matching should be employed as a more descriptive approach, even if binary evaluation measures are used afterwards.

Ensuring descriptive and concise evaluation

As concluded in the previous parts of this review, event-level evaluation is a step towards a more intuitive estimate of the algorithm's performance, contributing to the fairness of the evaluation on the whole, as it enables e.g. distinguishing between slight timing offsets and systematic fragmentation or merge errors. Furthermore, the ability to quantify and discuss the performance of an eye movement detector in terms of complete events opens up many possibilities that are simply unavailable for sample-level analyses, such as exploring event timing, thus potentially increasing the descriptiveness of the analysis as well.

One of the desired characteristics of a descriptive evaluation is providing the developer and the user of the algorithm with sufficiently precise and interpretable insight into the source of the inaccuracies in the algorithm's detections, not merely *some* evaluation score. Here, event matching is crucial, as it is exactly the element of the evaluation pipeline that deals with the following reasoning: Which events need to be considered together in order to judge how well the algorithm performed in this particular instance? Thus, in order to draw the conclusions that correspond to the intuitive understanding of the errors that the detection algorithm is making, event matching used during the evaluation has to correspond to the intuitive ideas of the experts in the field. This has proven to be difficult to achieve in practice, and while work has been done to incorporate that intuition into the matching schemes, there is still room for improvement: Each of the methods we

examined in this review has its pros and cons, and it is important to know them before choosing the one to use.

In order to demonstrate the properties of various evaluation metrics and event matching approaches, we used a recent real large-scale (ca. 2 h) dataset of eye movement events during movie watching (Agtzidis et al., 2020), for which the predictions of fifteen algorithms were provided alongside the expert annotations. On these data, we applied and systematically compared a number of combinations of different event matchers (Table 4) and evaluation metrics (Table 3). Our results support the conclusion that reliable eye movement event detection performance evaluation requires a combination of an *intuitive event matcher* that does not bias the evaluation by establishing counterintuitive matches that inflate the scores of the considered metric, and a *metric that compensates for class imbalance*.

In the following, we will first summarize the role and importance of good event *matching* – the cornerstone of any event-level evaluation pipeline, and discuss the contribution of match strictness to the quality of ensuing evaluation. We then discuss the benefits and limitations of different types of quantitative evaluations that can be applied after the events are matched.

Event matching

Matching eye movement events between the two compared sets is the key to performing most of the quantitative analyses, e.g. assessing the similarity between detections of an algorithm and the ground truth on the level of events. Technically, *any* event matching strategy would enable this, though perhaps not equally well.

For example, as we both argue theoretically in "Event matching methods" and practically demonstrate in "Interaction between the performance metrics and event matchers", using either sample-level evaluation or the Majority voting (Hoppe & Bulling, 2016) approach to event-level evaluation leads to consistently overly-optimistic performance estimates, almost regardless of the choice of a metric, as these methods are mostly insensitive to event fragmentation and merge errors. While this is more related to the *fairness* of the evaluation, event matching can have a very direct influence on the *descriptiveness* as well. For instance both the Overlap (Hauperich et al., 2020) and the Earliest overlap (Hooge et al., 2018) event matching strategies were originally proposed in a binary setting, i.e. matching only the events of interest (e.g. fixations), and disregarding the rest. This restricts one from using standard evaluation measures that are either inherently multiclass (Cohen's kappa, MCC) or focus on the "negative" class (specificity). For the purposes of the analysis in this review we extended the Overlap and Earliest overlap matchers to a multiclass setting, enabling their direct application in combination with multiclass metrics.

In general, only matching events of a single class can make the overall evaluation process more difficult, as it rules out the usage of very concise multiclass evaluation metrics, for example, and is less flexible on the whole. When developing a general-purpose event matcher, we, therefore, recommend considering a multiclass scenario first and foremost. Note that this does not limit event matching in itself (i.e. the matcher can still only match fixations to fixations and saccades to saccades, etc.), but ensures the matching can be applied even when several eye movement types are annotated and detected.

Another important property of an event matching strategy is whether it enables matching only individual events to one another (*one-to-one matching*), or whether matches between event groups are allowed as well: In practice, eye movement event matchers that go beyond one-to-one matches – EDEC (Ward et al., 2006)[17] and Window-based matching (Kothari et al., 2020) – focus on correctly handling merges and fragmentations, leading to the creation of matchers that can be characterized as *one-to-many* and *many-to-one* matchers, respectively (cf. additional remarks on this terminology in "Event-level evaluation").

The extension of EDEC by Steil et al. (2018) introduced the "merge-fragmentation" error type, thus potentially making it a *many-to-many* matcher. However, we do not recommend using this variant especially for high-frequency eye tracking data, since it can lead to large parts of the eye movement labels being labelled as a single error (cf. "Event-driven error characterization").

The major benefits of using one-to-many and many-to-one instead of one-to-one matching are (1) the potential to quantify event merging or fragmentation (which is impossible to achieve with one-to-one matchers, at least directly), and (2) the potentially more precise subsequent event timing evaluation (what we referred to as "event quality metrics" in "Event quality metrics"). The latter also relates to the way in which a matcher handles fragmentation and merging (illustrated in Fig. 9): E.g. for the fragmentation case, if only one of the detected events is matched to the ground truth event, any event timing comparison will show large timing errors, even though the on- and offset of the ground truth event were detected precisely. Thus, without a thorough manual examination of the predictions and event matches, the resulting timing scores might be misleading. While one-to-many matching solves these issues, developing an intuitive and robust matching strategy is not a trivial task, and existing

[17] EDEC does not include an explicitly defined event matcher, thus we here mean the matching that can be *derived* by interpreting the results of EDEC – see Appendix E.

one-to-many methods exhibit unintuitive behavior, either potentially suppressing or misattributing the errors (cf. examples in "Window-based matching" and "Event-driven error characterization"). Another approach for more precise assessment of event timing errors in the presence of merging and fragmentation was proposed by Hooge et al. (2018). It consists in applying a one-to-one event matcher (Earliest overlap) twice – once in the natural temporal direction to assess the timing of onsets, and once more in the reverse direction to assess the timing of event offsets. Since one event can thus be matched to two different ones, this effectively creates a one-to-many event matching strategy. However, such multi-step process also separates the computation of event detection and event timing scores, making the whole procedure somewhat less flexible.

Our review and empirical analysis of the event matchers indicate that the most reliable and concise algorithm performance estimation is likely achieved by employing the Maximum overlap or Maximum IoU matchers. These yield similar scores in both multiclass and binary settings, while at the same time considering all the errors the algorithm makes. In addition such evaluation enables very descriptive subsequent analysis of algorithm errors – be it confusion matrix, analysis of eye movement metrics, or event timing evaluation (though Earliest overlap matcher is more suitable for the latter purpose).

The event matchers that can match several events to one, and thus capable of handling event merging and fragmentation, could add a further layer of evaluation, enabling direct quantification of these error types. However, currently available one-to-many event matching strategies lack the robustness compared to the more straightforward one-to-one event matching: For both Window-based and EDEC matching strategies, it is relatively easy to produce an example ground truth and predicted event sequences, for which the resulting matching would be counter-intuitive.

We believe that while recent works have produced reliable one-to-one matching approaches, future improvements in the area of one-to-many eye movement event matching are needed. This should eventually bring the whole eye movement event detection research field closer to a generally applicable and flexible evaluation, and thus closer to much improved comparability of evaluation between publications.

Match strictness The intuitiveness of some event matching strategies is undermined by potentially registering two events that overlap by a single sample as a successful match, and thus communicating a correct detection to the subsequent event detection evaluation. Not all event matchers account for the quality of the produced matches in their pipeline (cf. Table 4), and those that do, might still allow arbitrary-quality matches (although potentially having given priority to higher-quality ones). As such, there is an obvious use case for a generic procedure that would enforce certain guarantees on the events that end up being matched.

In principle, any minimal match criteria can be imposed (e.g. degree of overlap, or other timing-based criteria). The important issue is that the verification of these criteria should ideally be *integrated* into the matching procedure: Filtering out the poor quality matches only *after* the event matching may have a different effect on the evaluation compared to doing so *inside* the matching algorithm. If the implementation of the event matching algorithm iterates over candidate matches in the order of their quality (cf. "Note on implementation differences") *and* the match acceptance criterion is represented by a threshold on the same quality measure (e.g. an IoU threshold for the Maximum IoU matcher (Startsev et al., 2019a)), applying the filtering post-hoc or inside the matching algorithm will be identical. In any other case (e.g. a criterion based on the alignment of events' on- and offsets for a Maximum overlap matcher, or a Maximum IoU matcher with a start-to-end iteration order), there will be a difference between the two ways of applying the match filtering criteria: If a ground truth event has multiple candidate matches, some fulfilling the acceptance criteria and some not, selecting from those during the event matching itself leaves the possibility to guarantee that a match will be found. With the post-hoc filtering, however, it is possible that the chosen match will not fulfill the defined acceptance criteria, resulting in the ground truth event being registered as "undetected" (unmatched), changing the evaluation outcome.

Such a simple modification was originally proposed in Startsev et al. (2019a), where the *Maximum IoU* matcher was presented together with its variant that rejected match candidates, where event pairs had the intersection-over-union statistic lower than a certain threshold. This allowed for varying the degree of evaluation strictness with a single parameter, and thus for comparing the performance of a number of eye movement detectors over a range of strictness levels (Startsev et al., 2019a, Fig. 4).

Overlap as a minimal requirement for matching An interesting question regarding the criteria that should be imposed on the matched events is raised by the differences in the behavior of any of the overlap-based matches and the *Window-based* matching (Kothari et al., 2020): The latter method allows for short events to be matched even when they do not overlap, as long as both their onsets and their offsets are not shifted by over a fixed threshold (the authors used 25 ms for saccade events, but a threshold can be changed by the pipeline user). In contrast, the other automatic event matching methods do not allow matches that are not overlapping by at least one sample.

Both of these approaches have their downsides: On the one hand, a temporal offset of more than the duration of the respective event should usually be enough to declare the ground truth event as missed and the predicted events as a false detection (i.e. no need for registering a match). On the other hand, as in the example given in "Event-driven error characterization", there may be a case where systematic temporal offsets are introduced by the detector (e.g. via using noise filtering without delay compensation). In this case it could be desirable to register the misaligned events as matches, and later quantify their temporal offsets.

On the whole, we recommend a combination of using the evaluation scheme that only registers events as matched when they overlap and manually inspecting the predictions. The former covers the prevalent cases, with no noticeable systematic temporal offset that is not accounted for, while the latter would hopefully catch such systematic errors (especially if the developer is aware of the potential problems of the event detection algorithm related to signal filtering; or if they are alerted by the extremely low scores obtained in the automatic evaluation procedure).

Evaluation measures

There are two relatively well-defined groups of metrics used to quantify the quality of the eye movement detector outputs in the literature: *detection performance* and *event quality* measures. The former (cf. "Levenshtein distance" and "Confusion matrix-based measures") consist mostly of various confusion matrix-based analyses (with Levenshtein distance being an exception), and focus on quantifying how many events in the ground truth received a correct or incorrect label from the eye movement detector, how many events were falsely detected, etc. These metrics are typically summarizing a (part of) confusion matrix to allow quickly assessing some aspect of either overall or per-class performance of the algorithm.

While detection performance metrics ignore e.g. the differences between the ground truth and detected event on- and offsets (as long as the event matcher registered a match), event quality metrics (cf. "Event quality metrics") focus on assessing the alignment between the matched events. Typically, these metrics ignore events that are not matched (e.g. an undetected ground truth event would not affect the average event quality score), which speaks strongly for using them *in combination* with detection performance metrics. An exception from this pattern is the event-quality IoU metric as it was used in (Startsev et al., 2019a), where the average statistic was computed over all ground truth events, with 0 assumed where no match was found. This is, however, not something that should be generally recommended, as it mixes the incomplete picture of detection quality (e.g. false detections are ignored) with event quality, thus reducing the interpretability of the measure.

Detection performance metrics An example of a relatively descriptive event-level evaluation could be reporting the detection performance via a confusion matrix (Zemblys et al., 2019b, Fig. 9, e.g.), thus providing a very detailed overview of the algorithm's performance, both for the end user and the algorithm developer: One can examine the number or proportion of mislabeled events, as well as which events get mislabeled most frequently. This could enable them to decide whether the algorithm is suitable for detecting a certain event type. For the developer, examining the confusion matrix additionally provides valuable insights by highlighting the most problematic eye movement event confusion or miss cases, etc.

Concise evaluation is enabled by calculating classification performance metrics described in "Evaluation metrics", summarizing the corresponding confusion matrix. However metrics such as accuracy, precision, sensitivity, specificity, F1-score, as well as sample-level JI are highly affected by imbalanced data (i.e. usually the prevalence of fixations). Therefore, they were assigned a low usability rank in our analysis (see Table 3). Levenshtein distance produces scores very similar to accuracy, and consequently also receives a low usability rank. It should thus be avoided due to data imbalance effects as well as other objections to its suitability for eye movement data (cf. Levenshtein distance").

It is worth highlighting that we do not, in principle, advocate against using the metrics with low "usability on their own" rank, since in most cases one *does* possess information about class balance, and can compare the scores to those of another algorithm. Also, reporting multiple scores ranked as "low", for example, is a perfectly acceptable practice: E.g. precision and recall are a great way to quickly dissect the errors that the algorithm is making (labelling "additional" samples/events that do not belong to the examined class, or missing the "true" samples/events of it), although this shifts the balance from concise to descriptive evaluation.

Two metrics received a "high" usability-on-their-own rank: Cohen's kappa and Matthews Correlation Coefficient (MCC). Both of these are designed to handle imbalanced data, making them specifically suitable to be used without detailed awareness of the eye movement class prevalence in a particular dataset. We note, however, that while Cohen's kappa is widely reported in the literature, it can be severely affected by rare classes (Delgado & Tibau, 2019), and can produce counterintuitively bad scores for event-level evaluation of chance-level predictions (Startsev et al., 2019). MCC, on the contrary, is seemingly more robust but nowhere near as widely spread as Cohen's kappa.

Limitations of a confusion matrix One fundamental descriptiveness-related limitation of the confusion matrix-based analysis, however, surfaces when considering event-level evaluation that can be operating with one-to-many matches, such as would be representing merging or fragmenting the events. For instance, it is difficult to objectively reflect via a confusion matrix the situation when a fixation is fragmented into three shorter fixations with two falsely detected saccades in-between. The falsely detected saccades can be easily accounted for (e.g. as +2 added to the cell of the matrix corresponding to the row of "undefined" and the column of "saccade"). The fixation fragmentation is problematic, however: Mostly likely, registering one correct detection (i.e. +1 at the intersection of the "fixation" row and column) and two false detections (+2 for the "undefined" row and "fixation" column intersection) would adequately reflect both the good and the bad aspects of such a detection (i.e. a fixation is detected, but is heavily fragmented). However, after these counts are recorded into the confusion matrix, the *connection* between the event fragments is broken: No subsequent analysis will be able to quantify the degree of event fragmentation, or report on what event types mostly cause the fragmentation.

Binary vs. multiclass metrics The choice between binary and multiclass evaluation metrics effectively provides the researchers with a trade-off between descriptiveness and conciseness of the evaluation: Binary metrics allow examining the performance of the algorithm per each eye movement type, but need to be reported for every class of interest. If conciseness is more crucial, multiclass measures will provide a single score for all type, thus making e.g. ranking several algorithms easier, but at the cost of lower level of detail of the evaluation.

Event quality measures Based on our overview of the event quality measures, no strong preference for one or the other measure can be declared, and, aside from the ease of interpretation for a particular researcher, the metrics differ relatively little. The only set of measures substantially different from the others (in terms of descriptiveness and conciseness) are related to the relative timing offset analysis proposed by Hooge et al. (2018): These report the mean and standard deviation of the differences between the matched event on- and offsets. While this can be a very descriptive measure (computed in ms and tied to a particular "end" of the respective events), the amount of reported statistics may seem excessive for a brief summary of performance: When multiple eye movement types are considered, mean and standard deviation need to be reported for every event class, separately for event on- and offsets.

As the users of eye movement detectors may want to assess whether using a particular algorithm will likely significantly alter subsequently analyzed event statistics (e.g. average fixation duration, mean saccade amplitude, etc.), these can be directly incorporated in the event quality analysis: Any of the rich number of eye movement statistics can be used in this context (cf. "Evaluation based on eye movement metrics"), possibly combined with e.g. Bland-Altman analysis ("Event quality metrics"), thus providing exactly the necessary data to a potential user.

Ensuring comparability to the literature

First of all, it is important to note that the vast majority of research to date is limited to sample-level evaluation. Event-level evaluation is not as of yet standardized, so comparability of event-level analyses can only be ensured by executing the same evaluation procedure – either published together with the algorithm one is comparing against, or own pipeline executed for the predictions of the baseline algorithms. As we have stressed in this review, implementation details of the event-level evaluation have a direct effect on the resulting statistics, which are difficult to assess in advance, so any re-implementation effort of such a pipeline needs to be carefully validated before comparing the results to the original publication.

In terms of sample-level analysis, Cohen's kappa is, on the whole, one of the most reported measure of agreement in eye movement literature, therefore reporting it has the benefit of wide comparability across different publications. F1-score would probably take the second spot in the popularity list, and thus has similar benefits.

When performing comparison to a particular publication, it may be advisable to use or implement the evaluation approach in that work, even if it may be poor in terms of its descriptiveness or even fairness – this can serve as a preliminary comparison method and will ensure that the results of that particular work are reproduced correctly. In order to improve the quality or level of detail of the evaluation, adding further evaluation strategies is perfectly reasonable, of course.

It has to be noted, however, that any measure in any evaluation scenario is only fully comparable between two instances of computing it when both the evaluation *pipeline* (especially for event-level analyses) and the *dataset* are the same. Differences in the datasets, for instance, will likely include variations in class balance, which will inevitably affect most evaluation measures. The presence or absence of certain event types (e.g. smooth pursuit) can even affect binary evaluation strategies focusing on fixation detection. Alternatively, comparing the performance of the same algorithm for several datasets reveals the generalization ability of the algorithm.

Conclusions

This review of a seemingly narrow topic of eye movement event detection evaluation demonstrates it being a complex, continually evolving subject, full of trade-offs and compromises. To be able to both choose and develop sensible evaluation pipelines, an overview of existing approaches is important. The purpose of this review is to provide the reader with the information and tools enabling them to make an informed decision about what methods to use to fulfill their particular evaluation goals. We assembled the recommendations covering a broad spectrum of topics spanning complete evaluation process – from dataset selection or collection and how these data are used for validation and evaluation ("Evaluation protocols"), to high-level principles of the evaluation ("Evaluation methods"), to the specific information about quantitatively comparing the output of an eye movement detection algorithm to "gold standards" (typically – manual annotations) – both the metrics that can be used in such analysis ("Evaluation metrics") and the strategies employed to evaluate eye movement events instead of single gaze tracking samples ("Event matching methods").

In place of a brief summary, we suggest the following as best practices in eye movement event evaluation:

- To assess the quality of an eye movement detector, whenever possible or practical *quantitatively compare its outputs to expert annotations*. While these may not be the coveted "gold standard" of eye movement event labels, they provide a re-usable common ground for evaluation and lend themselves to extracting the most informative insights into the error patterns of the examined event detector.
- *Carefully report the complete evaluation pipeline* and provide the *annotations* used in place of the ground truth, the algorithm *outputs*, as well as the *code* for the evaluation and the detection algorithm together with the results. Ensure that the chosen pipeline allows for *fair algorithm performance assessment* (cf. "Evaluation protocols" and the summary in "Ensuring fair evaluation").
- *Use event-level evaluation strategies.* Event fragmentation is one of the most frequent problems of existing algorithms, and explicitly accounting for it is only possible with event-level evaluation. Refer to "Summary" of the "Evaluation metrics" section for remarks on the various event matching approaches that form the cornerstone of this kind of evaluation, as well as "Interaction between the performance metrics and event matchers" for the empirical observations on their properties. Specifically for event matching, we recommend the Maximum IoU matcher with highest-to-lowest IoU iteration order for reliable one-to-one matching, and encourage further refinement of one-to-many matching strategies.
- *Use a combination of the metrics that quantify event detection* (i.e. confusion matrix-based analyses; from these, we particularly recommend Matthews correlation coefficient as a reliable measure) *and the timing of detected events* (what we refer to as event quality measures). Refer to "Summary" for remarks on the properties of individual evaluation metrics, as well as "Interaction between the performance metrics and event matchers".
- When focusing on the detection patterns of a *single eye movement detector, use multiclass evaluation strategies* (including multiclass event matching). These provide the most descriptive and most concise summary of the errors an algorithm is making.
- When comparing a *diverse set of eye movement detectors* (e.g. not all detecting the same set of eye movement types), *use binary evaluation strategies* with binary sample-level remapping prior to event matching (cf. "Multiclass vs. binary evaluation"). This ensures a comparable evaluation of each eye movement type detection, regardless of the algorithms' detection patterns for other event types.
- Regardless of the metrics used e.g. for concisely comparing between a set of algorithms, *provide a confusion matrix to describe the algorithm(s) of special interest*. The readers can then intuitively interpret the typical errors of the respective methods, and even calculate a specific metric in order to compare results between the studies. Make sure the confusion matrix includes a special row and column that would *allow reporting both false detections and missed events* of each eye movement type.

We provide the implementation of all the metrics and event-level evaluation strategies we compared in "Interaction between the performance metrics and event matchers" in the code repository accessible via https://github.com/r-zemblys/EM-event-detection-evaluation. While using this exact implementation requires converting the input data to the corresponding format, the repository can always be used as a source of building blocks for designing own evaluation pipeline, as we provide the event matching and subsequent evaluation as separate steps, ready to be modified when necessary. With making this range of tools publicly available, we hope to encourage the reproducibility of both eye movement detection research in general and the quantitative evaluation of the algorithms in particular.

Appendix A: Eye movement "detection" vs "classification"

In the literature, the process of parsing raw gaze data into events is called "eye movement (event) detection" or "eye movement (event) classification". In case an algorithm is only meant to detect one particular eye movement type, it could be referred to as e.g. a fixation "detection" or a fixation "classification" algorithm. Occasionally, "identification" and "segmentation" are used as well (Salvucci and Goldberg, 2000; Pekkanen & Lappi, 2017; Hessels et al., 2017, to name a few).

The two terms most commonly used in this context – "detection" and "classification" – are usually used arbitrarily, not detailing the reason behind the applicability of the particular terminology. Moreover, many papers use "detection" and "classification" interchangeably, even when referring to the same algorithm (Larsson et al., 2015; Andersson et al., 2017; Hooge et al., 2018). Hessels et al. (2018) argue that the term "classification" is to be preferred because, according to the authors, the word "detection" would imply that "an oculomotor event [...] is objectively present in the eye-tracker signal and all one needs to do is detect where it is" (Hessels et al., 2018, p. 7), which is not a universally agreed-upon statement.

As already stated in the introduction, we strive to use the term "detection", as in our view it better describes the end result of the eye tracking data processing system that we want to evaluate. Namely, these systems provide their user with some representation of the information, where eye movement events of certain types begin and end. This means that, effectively, eye movement localization and classification are performed. In many fields outside eye tracking, e.g. image or signal processing, this process has an established name – detection. Note that in the case of one-dimensional event label data, the distinction between detection and segmentation observed for images, etc., is erased.

The algorithms that are used in such pipelines to reach the end result are not limited in their implementation, and can in practice be anything from simple thresholding (Salvucci & Goldberg, 2000) to machine learning-based classifiers (e.g. Zemblys et al., 2018) or sequence-to-sequence deep learning models (e.g. Zemblys et al., 2019b; Startsev et al., 2019a), whether they operate with individual gaze samples, windows of those, or entire recordings. To avoid confusion, however, this should not change the name used for the overarching system, even if this system merely records sequences of uninterrupted eye movement labels produced by the underlying algorithm into corresponding "events".

Therefore, as well as in order to use the same terminology as in other research fields where there is a clear distinction between classification and detection, we use the term "event detection" throughout this paper.

Appendix B: Clean observer and stimuli separation

Figure 19 illustrates for the problem of separating both the observers and the stimuli material for training and testing of the model, when the considering a dataset where one set of observers viewed one set of stimuli. Consider e.g. allocating 25% of the stimuli material and 25% of the observers for testing, thus in terms of gaze samples making the test set roughly only 1/16 (in red) of the full data collection. In addition, none of the observers *or* stimuli used in testing can be used for training purposes. This would mean that further ca. 38% (in grey) have to be discarded entirely, leaving just over half (ca. 56%, in green) of the data eligible for algorithm development. This would make cross-validation confusing, and reduce the utilization of the dataset.

Appendix C: Time-wise splitting for eye tracking data

To test the time-wise splitting strategy (Fig. 20 and "How to split the data"), we reproduced the evaluation set-up from (Startsev et al., 2019a), where a deep learning

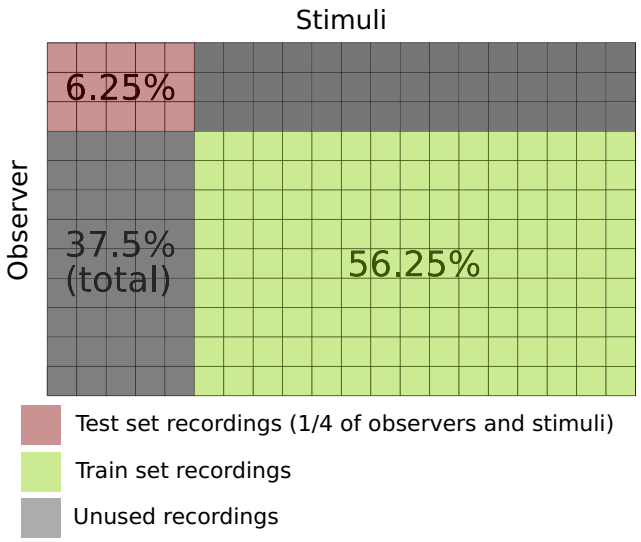

Fig. 19 Splitting a dataset consisting of recordings for a number of observers, each having been exposed to the same stimuli

model is trained in stimulus-based cross-validation procedure (i.e. "leave-one-video-out"). The test set corresponds to the "left-out" fold (i.e. the eye tracking recordings for a single video), while the recordings for the remainder of the videos are used as a source for both training and validation data for the model optimization procedure.

In the original experiment, training and validation examples (inputs to the network – windows of gaze data with corresponding labels) ware randomly drawn from the non-test-video gaze recordings with a stride of ca. $1/4\,s$. In principle, therefore, windows of gaze samples (reaching up to 1 s in the original work; windows of 0.5 s were used here, as well as in the vast majority of tests in the original paper) that are very close to one another (or even contain largely the same gaze samples) can end up one in validation, one in training sets.

In this experiment, we compared the usage of this random splitting to the time-wise splitting (see Fig. 20 for an intuitive illustration) in order to separate training and validation sets for the network optimization process. The validation set (in both cases – 10% of the data) was used for early stopping (determining the optimal point to stop the training in order to avoid overfitting) – a modification compared to the original paper.

We present the evaluation in Table 5, obtained with the same evaluation pipeline as in the original paper. In our experiments, time-wise training-validation splitting led to a consistently better-performing model (for all eye movements and both sample- and event-level F1-scores), with absolute F1 score differences within 2%, except for event-level pursuit detection score that showed a 5% improvement.

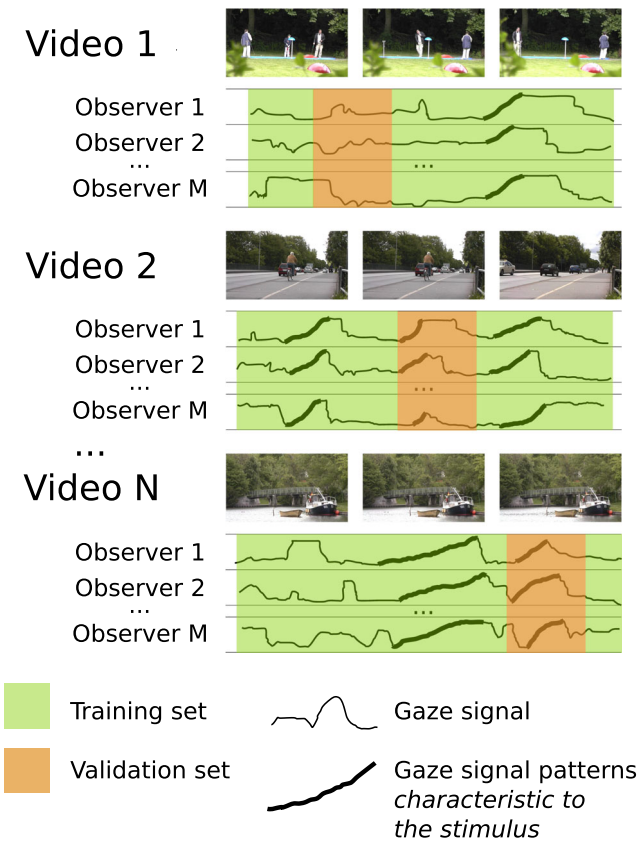

Fig. 20 Schematic illustration for *time-wise* splitting of the dataset with multiple observers viewing the same stimuli: Selecting the validation set (in orange) as the parts of gaze recordings corresponding to *the same time moments* of each video for all observers. This way, we avoid including similar stimulus-specific gaze patterns (bold line segments) in training (green) and validation (orange) sets, thus obtaining overly optimistic validation scores due to possible overfitting

Appendix D: Corner cases for sample-level metrics

In Table 6 we provide the scores of various metrics we discussed in "Evaluation metrics" in a set of corner-case scenarios. Unlike in "Interaction between the performance metrics and event matchers", we do not aim for a realistic evaluation scenario here, but rather strive to conceptually simplify the compared scenarios. We do this in order to make the judgements about the observed scores easier.

To this end, we assembled a set of situations that would reveal how each evaluation metric behaves in various "corner cases". For the "ground truth" source, we generate synthetic event sequences that coarsely simulate a scenario found in real eye tracking data, namely the alternating sequence of fixations and saccades. For this particular test,

Table 5 Evaluation results contrasting typical random splitting of training set examples (with "examples" being potentially overlapping gaze signal windows here) vs. the proposed time-wise splitting. The two splitting modes were used for separating training and validation sets during he optimization process of otherwise identical neural networks. Test sets are identical in both cases. Best result for each metric is **boldified**

Metric	Random splitting	Time-wise splitting
Fixation: sample F1	0.931	**0.937**
Fixation: event F1	0.878	**0.890**
Saccade: sample F1	0.872	**0.892**
Saccade: event F1	0.939	**0.943**
Pursuit: sample F1	0.683	**0.693**
Pursuit: event F1	0.529	**0.580**

we generate 100 positive-negative event pairs where each pair is comprised of 90 positive and 10 negative samples. This can represent for example, 100 fixations followed by 100 saccades. While the organization into events does not matter for most of the sample-level metrics, Levenshtein distance depends on the order in which samples are ordered.

The corner-case predictions, to which the ground truth from above is compared, include the following scenarios: (i) all positive class labels, (ii) all negative, (iii) random, (iv) randomly shuffled ground truth sample-level labels, (v) predictions opposite to the ground truth labels, and (vi) randomly shuffled ground truth events. The latter case simulates event merge and fragmentation errors while preserving the same sample class distribution (Startsev et al., 2019). Since some of the metrics require labels of both positive and negative class to be computed, we additionally include the tests where all but one labels in the "predictions" belong to the majority or minority class, respectively.

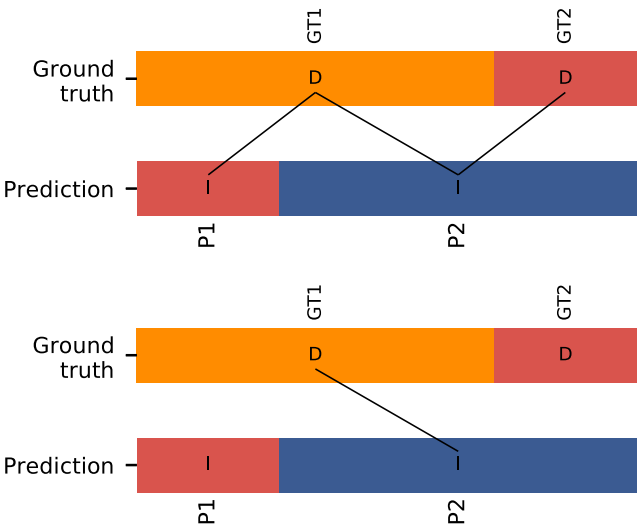

Fig. 21 Potential ambiguity of event matching derived from EDEC (top), and its possible resolution via an additional maximum overlap matching criterion. Blue, red, and orange are fixations, saccades, and smooth pursuits, respectively. Black lines between the ground truth and predicted events indicate event matches. D are deletion, I – insertion errors

Appendix E: Event matching using Event-driven error characterization (EDEC)

Here we provide a potential approach for obtaining explicit event matches using the output of the segment-level evaluation (Ward et al., 2006, Fig. 2), right that we used to illustrate how EDEC could work as an event matcher e.g. in Figs. 14 and 15. EDEC implementation first subdivides the eye movement events into segments (i.e. intervals during which *neither* ground truth *nor* prediction change). After this step, all events in the ground truth and in predictions that share a segment can be easily matched to one another.

In this way, events consisting of multiple segments may be matched to several events in the other sequence (corresponding to merge and fragmentation errors, e.g.). This effectively makes the proposed approach a one-to-many and many-to-one matcher. As with any other such matching strategy, this leads to an ambiguity of accounting for complex event matches in e.g. confusion matrix-based analysis (cf. "Evaluation measures")

Table 6 Classification scores for binary synthetic ground truth data with 90% of samples belonging to the positive class. The values that could not be computed because of division by 0, are undefined for a particular case, or similar, are indicated with "-"

	All majority	All but one majority	All minority	All but one minority	Random	Shuffle	Opposite	Random event shuffle
NLD	0.1	0.1	0.9	0.9	0.47	0.16	0.8	0.09
Accuracy	0.9	0.9	0.1	0.1	0.5	0.82	0	0.83
Balanced accuracy	0.5	0.5	0.5	0.5	0.5	0.5	0	0.52
Precision	0.9	0.9	-	0	0.9	0.9	0	0.9
Sensitivity	1	1	0	0	0.5	0.9	0	0.9
Specificity	0	0	1	1	0.5	0.1	0	0.13
F1-score	0.95	0.95	0	0	0.65	0.9	0	0.9
AUC	0.5	0.5	0.5	0.5	0.5	0.5	0	0.52
Jaccard index	0.9	0.9	0	0	0.48	0.82	0	0.82
κ	0	0	0	0	0	0	-0.22	0.03
Pearson's r	-	0	-	-0.03	0	0	-1	0.03
Spearman's ρ	-	0	-	-0.03	0	0	-1	0.03
Kendall's τ	-	0	-	-0.03	0	0	-1	0.03
MCC	-	0	-	-0.03	0	0	-1	0.03

An additional source of ambiguity comes from the potential many-to-many matches that could result from applying this matching scheme. E.g. in the top part of Fig. 21, both GT1 and GT2 have a match with P2, but GT1 is also matched with P1. This results in a group of events (GT1, GT2, P1, and P2) that are all "matched" to one another, if match transitivity were to be followed. Since this would be impractical (as well as unintuitive) for any further evaluation, the conflicting matches would need to be dealt with prior to any further steps in the pipeline. The bottom pair of scarf plots in Fig. 21 demonstrates an example of solving match conflicts by using the maximum overlap (Zemblys et al., 2019b) approach. Since GT1 and P2 overlap the most, they are matched creating smooth pursuit–fixation error, while the remaining events may be considered as unmatched, enabling the application of traditional evaluation metrics.

Appendix F: Under- and over-estimated performance using balanced accuracy

To demonstrate the potential pitfalls of using balanced accuracy for event detection evaluation, we picked two scenarios, both in the Hollywood2EM dataset (Agtzidis et al., 2020): First, a non-expert's annotations ("student coder") were compared to the final labelling that was corrected by an experienced coder. While showing some differences, the two sets of labels largely agree, as they were produced following the same principles and definitions. The second scenario is comparing the labels of the I-VVT algorithm to the final expert annotations. I-VVT is a simple two-threshold method for eye movement classification that is highly influenced by noise, and on a qualitative level provides very fragmented predictions. Figure 22 shows five multiclass metrics for the student coder scenario (top) and the I-VVT algorithm (bottom). These were calculated using different sample- and event-level evaluation approaches, which are noted on the horizontal axis.

Aside from the balanced accuracy, the various event-level performance scores (from "Earliest overlap" onward on the horizontal axis) conform with the intuitive idea of detection quality: The labels of the manual annotators, similar in nature and principle, achieve high scores, and the noisy prediction labels have very poor event-level scores. Note, however, the idiosyncratic behavior of balanced accuracy (in orange): While yielding consistently lower scores in comparison to all other metrics when events of the two human annotators are evaluated, balanced accuracy for the I-VVT algorithm is instead considerably higher. In fact, when using the Overlap matcher, balanced accuracy for I-VVT is close to that of the manual annotator (0.75 vs. 0.85, respectively). This signals that at least this particular

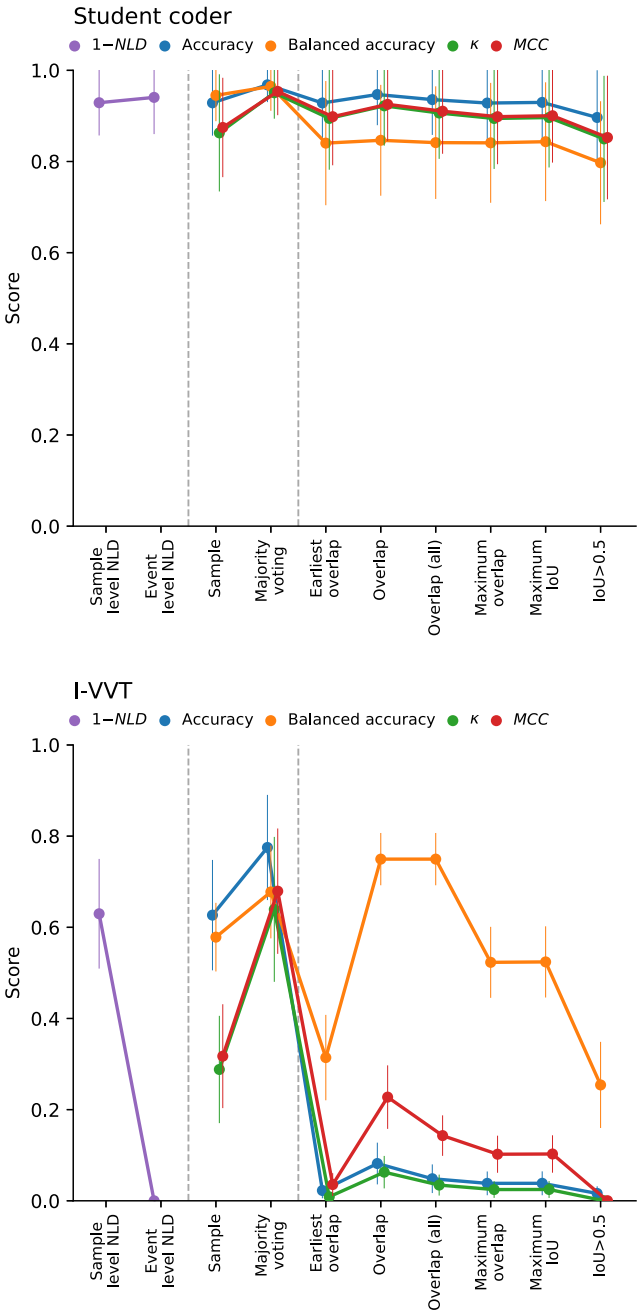

Fig. 22 Example of balanced accuracy score under-estimating (top, human coder) and over-estimating (bottom, I-VVT algorithm) event-level eye movement event detection performance

combination of event matching and the metric would not form an adequate basis for a fair performance evaluation. While the other matching strategies in combination with balanced accuracy yield lower scores, they still suggest a substantially higher prediction quality than any other measure combined with any other event matching strategy.

This odd behavior of balanced accuracy stems from how the event matching is achieved, and how unmatched events contribute to the performance score (as I-VVT is prone

to event fragmentation, and all the event matchers tested here are one-to-one, such evaluation will produce numerous unmatched events in the predicted sequence). In multiclass-setting evaluation, unmatched events are typically treated as errors of some kind, and should intuitively contribute to lowering the final score. When confusion matrix analysis is applied, the way in which this is done depends on whether the unmatched event occurs in the ground truth or the prediction sequence: Missed events (i.e. unmatched events in the ground truth), typically contribute to the intersection of the row corresponding to the event type and a *special column* dedicated to undetected events; false detections (i.e. unmatched predicted events) contribute to the intersection of the *special row* dedicated to non-existent events and the column corresponding to the event type.

Balanced accuracy, in its most widespread form, equals the average of per-class sensitivity scores, i.e. the average of values on the diagonal of the row-wise normalized confusion matrix, including the one in the special row for false detections in the case of event-level evaluation. Since the value on the diagonal in the latter will be zero (a false detection cannot be "correct" by definition), including it in the averaging will always drive the scores down (cf. the top plot in Fig. 22). On the other hand, by averaging *row*-normalized values on the diagonal, the potentially multitudinous false detections (e.g. for I-VVT) will not be accounted for in the normalization, driving the values on the diagonal high (cf. the bottom plot in Fig. 22).

The latter problem is exacerbated by same-type-only event matching (e.g. the Overlap event matcher), since the row-normalized values on the diagonal will be exactly 1.0, making the combination of such matching strategies and balanced accuracy particularly bad for representing the true prediction quality of an eye movement detector.

Appendix G: Effect of treating unmatched negatives as true negatives

In binary setting, the Earliest overlap and Overlap event matching methods do not allow for the negative-class events (i.e. not belonging to the event type under evaluation) to be matched, meaning that the matching output will not contain any true negatives. As a consequence, Cohen's kappa and MCC become less or equal to zero, according to our observations in Fig. 16b, indicating a below-chance level of performance. This makes the evaluation and comparison of the algorithm performance highly impractical.

One possible approach to enable the evaluation in such situations is to convert unmatched negatives to true negatives (Zemblys et al., 2020). In Fig. 23 we provide such evaluation. Comparing these results to the original values

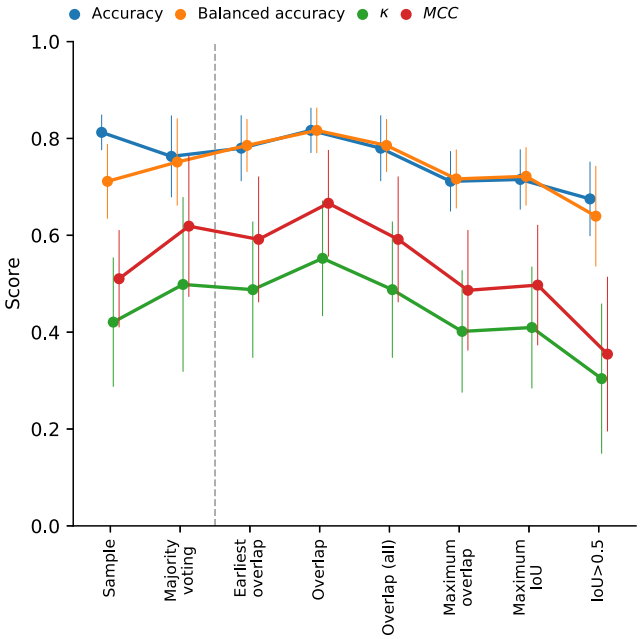

Fig. 23 Average sample- and event-level accuracy, balanced accuracy, Cohens's Kappa (κ), and Matthews correlation coefficient (MCC) scores of 15 event detection algorithms, evaluated in a binary setting. All scores represent average performance of binary fixation, saccade and smooth pursuit detection. Event-level scores were calculated using the Majority voting, Earliest overlap, Overlap (two modes), Maximum overlap, and Maximum IoU event matching methods. Unmatched negatives were converted to true negatives. Error bars show ± 1 standard deviation

reported in Fig. 16b, we observe that (i) most noticeably, MCC and κ scores are non-zero for Earliest overlap and Overlap matchers; (ii) accuracy and balanced accuracy become very similar and much higher across the board, since *all* negative-class events that used to be treated as false or missing detections are effectively treated as correctly detected now; (iii) MCC and κ also increased (where the comparison can be made – for Maximum Overlap and IoU matchers), but by much less in comparison.

Declarations

Open Practices Statement The library is implemented in Python – a free and open-source programming language, as we want to encourage more researchers to use our codebase for result comparability, without requiring to obtain often expensive licenses of popular academic software packages.

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ..., et al (2016). Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning

Research, (Vol. 48, pp. 173–182). New York, New York, USA: PMLR.

Agtzidis, I., Startsev, M., & Dorr, M. (2019). 360-degree video gaze behaviour: A ground-truth data set and a classification algorithm for eye movements. In *Proceedings of the 27th ACM international conference on multimedia*, MM 19, (pp. 1007–1015). New York, NY, USA: ACM.

Arabadzhiyska, E., Tursun, O.T., Myszkowski, K., Seidel, H.-P., & Didyk, P. (2017). Saccade landing position prediction for gaze-contingent rendering. *ACM Transactions on Graphics (TOG)*, *36*(4), 1–12.

Agtzidis, I., Startsev, M., & Dorr, M. (2020). Two hours in Hollywood: A manually annotated ground truth data set of eye movements during movie clip watching. Journal of Eye Movement Research, 13, 4.

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.

Blignaut, P., & Wium, D. (2014). Eye-tracking data quality as affected by ethnicity and experimental design. *Behavior Research Methods*, *46*(1), 67–80.

Byrt, T., Bishop, J., & Carlin, JB. (1993). Bias, prevalence and kappa. *Journal of Clinical Epidemiology*, *46*(5), 423–429. http://www.sciencedirect.com/science/article/pii/089543569390018V.

Bellet, M.E., Bellet, J., Nienborg, H., Hafed, Z.M., & Berens, P. (2019). Human-level saccade detection performance using deep neural networks. *Journal of Neurophysiology*, *121*(2), 646–661.

Bland, J.M., & Altman, D.G. (1986). Statistical methods for assessing agreement between two methods of clinical measurement. *The Lancet*, *327*(8476), 307–310.

Bland, J.M., & Altman, D.G. (1995). Comparing methods of measurement: Why plotting difference against standard method is misleading. *The Lancet*, *346*(8982), 1085–1087.

Bulling, A., Ward, J.A., & Gellersen, H. (2012). Multimodal recognition of reading activity in transit using body-worn sensors, ACM Transactions on Applied Perception (TAP) 9, no. 1, 1–21.

Bulling, A., Ward, J.A., Gellersen, H., & Troster, G. (2010). Eye movement analysis for activity recognition using electrooculography, IEEE Transactions on Pattern Analysis and Machine Intelligence 33, no. 4, 741–753.

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

Cramir, H. (1946). Mathematical methods of statistics. Princeton U. Press, Princeton, 500.

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 1–13.

Chiu, C.-C., Sainath, T.N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., . . . , et al (2018). State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (pp. 4774–4778), IEEE.

Dalveren, G.GM., & Cagiltay, N.E. (2019). Evaluation of ten open-source eye-movement classification algorithms in simulated surgical scenarios. *IEEE Access*, *7*, 161794–161804.

DeLong, ER., DeLong, DM., & Clarke-Pearson, D.L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, *44*(3), 837–845. http://www.jstor.org/stable/2531595.

Dar, A.H., Wagner, A.S., & Hanke, M. (2020). REMoDNaV: Robust eye-movement classification for dynamic stimulation. *Behavior Research Methods*, *53*, 399–414.

Dai, W., Selesnick, I., Rizzo, J.-R., Rucker, J., & Hudson, T. (2016). A parametric model for saccadic eye movement, 2016 IEEE Signal Processing in Medicine and Biology Symposium (SPMB), pp. 1–6.

Dernoncourt, F., Bui, T., & Chang, W.lter. (2018). A framework for speech recognition benchmarking. In *Interspeech*, (pp. 169–170).

Delgado, R., & Tibau, X.-A. (2019). Why Cohen?s kappa should be avoided as performance measure in classification. *PloS one*, *14*(9), 1–26.

Duchowski, A.T. (2007). *Eye tracking methodology: Theory and practice*. Berlin, Heidelberg: Springer-Verlag.

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

Friedman, L. (2020). Brief communication: Three errors and two problems in a recent paper: gazenet: End-to-end eye-movement event detection with deep neural networks (zemblys, niehorster, and holmqvist). *Behavior Research Methods*, *52*, 1671–1680.

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*, *50*(4), 1374–1397.

Fuhl, W., Santini, T., Kuebler, T., Castner, N., Rosenstiel, W., & Kasneci, E. (2018). Eye movement simulation and detector creation to 5065 reduce laborious parameter adjustments, arXiv:1804.00970.

Gatys, L.A., Ecker, A.S., & Bethge, M. (2016). Image style transfer using convolutional neural networks, Proceedings of the IEEE Conference on Computer 5070 Vision and Pattern Recognition (CVPR).

Gorodkin, J. (2004). Comparing two K-category assignments by a K-category correlation coefficient. *Computational Biology and Chemistry*, *28*(5–6), 367–374.

Gutiérrez, J., Che, Z., Zhai, G., & Le Callet, P. (2020). Saliency4ASD: Challenge, dataset and tools for visual attention modeling for autism spectrum disorder. Signal Processing: Image Communication, 116092.

Greenhill, S.J. (2011). Levenshtein distances fail to identify language relationships accurately. *Computational Linguistics*, *37*(4), 689–698.

Giavarina, D. (2015). Understanding Bland Altman analysis. *Biochemia Medica*, *25*(2), 141–151.

Goltz, J., Grossberg, M., & Etemadpour, R. (2019). Exploring simple neural network architectures for eye movement classification. In *Proceedings of the 11th ACM symposium on eye tracking research & Applications*, ETRA '19, (pp. 1–5). New York, NY, USA: Association for Computing Machinery.

Hessels, R.S., Andersson, R., Hooge, I.TC., Nyström, M., & Kemner, C. (2015). Consequences of eye color, positioning, and head movement for eye-tracking data quality in infant research. *Infancy*, *20*(6), 601–633.

Hessels, R.S., Hooge, I.gnace.T.C., & Kemner, C. (2016). An in-depth look at saccadic search in infancy. *Journal of Vision*, *16*(8), 1–14.

Hessels, R.S., Niehorster, D.C., Kemner, C., & Hooge, I.gnace.T.C. (2017). Noise-robust fixation detection in eye movement data: Identification by two-means clustering (I2MC). *Behavior Research Methods*, *49*(5), 1802–1823.

Hessels, R.S., Niehorster, D.C., Nyström, M., Andersson, R., & Hooge, I.gnace.T.C. (2018). Is the eye-movement field confused about fixations and saccades? A survey among 124 researchers. *Royal Society Open Science*, *5*(8), 1–23.

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., & Van de Weijer, J. (2011). Eye tracking: A comprehensive guide to methods and measures. OUP Oxford.

Hooge, I.gnace.T.C., Niehorster, D.C., Nyström, M., Andersson, R., & Hessels, R.S. (2018). Is human classification by experienced untrained observers a gold standard in fixation detection?. *Behavior Research Methods*, *50*(5), 1864–1881.

Hoppe, S., & Bulling, A. (2016). End-to-end eye movement detection using convolutional neural networks, arXiv:1609.02452, 1–15.

Holmqvist, K., Nyström, M., & Andersson, R. (2011). Participants know best: Influence of calibration method on accuracy. *Journal of Vision*, *11*(11), 503–503.

Holmqvist, K., Nyström, M., & Mulvey, F. (2012). Eye Tracker Data Quality: What It is and How to Measure It. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, (pp. 45–52). New York, NY, USA: Association for Computing Machinery.

Houpt, J.W., Frame, M.E., & Blaha, L.M. (2018). Unsupervised parsing of gaze data with a beta-process vector auto-regressive hidden Markov model. *Behavior Research Methods*, *50*(5), 2074–2096.

Hauperich, A.K., Smithson, H.E., & Young, L.K. (2020). What makes a microsaccade? A review of 70 years research prompts a new detection method, Journal of Eye Movement Research 12, no. 6, 1–22.

Hanley, J.A., & McNeil, B.J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*(1), 29–36.

Hessels, R.S., & Hooge, I.TC. (2019). Eye tracking in developmental cognitive neuroscience ? The good, the bad and the ugly, Developmental Cognitive Neuroscience 40, 1–11.

Judd, T., Ehinger, K., Durand, F., & Torralba, A. (2009). Learning to predict where humans look. In *2009 IEEE 12th International Conference on Computer Vision*, (pp. 2106–2113).

Kinsman, T., Evans, K., Sweeney, G., Keane, T., & Pelz, J. (2012). Ego-motion compensation improves fixation detection in wearable eye tracking. In *Proceedings of the symposium on eye tracking research & applications*, ETRA '12, (pp. 221–224). New York, NY, USA: Association for Computing Machinery.

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.

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.

Katostaras, T., & Katostara, N. (2013). Area of the ROC curve when one point is available. *Studies in Health Technology and Informatics*, *191*, 219–221.

Kothari, R., Yang, Z., Kanan, C., Bailey, R., Pelz, J.B., & Diaz, G.J. (2020). Gaze-in-wild: A dataset for studying eye and head coordination in everyday activities. *Scientific Reports*, *10*(1), 1–18.

Kunze, K., Utsumi, Y., Shiga, Y., Kise, K., & Bulling, A. (2013). I know what you are reading: Recognition of document types using mobile eye tracking, Proceedings of the 2013 International Symposium on Wearable Computers, pp. 113–116.

Lappi, O. (2016). Eye movements in the wild: Oculomotor control, gaze behavior & frames of reference. *Neuroscience & Biobehavioral Reviews*, *69*, 49–68.

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.

Larsson, L., Schwaller, A., Nystöm, M., & Stridh, M.artin. (2016). Head movement compensation and multi-modal event detection in eye-tracking data for unconstrained head movements. *Journal of Neuroscience Methods*, *274*, 13–26.

Larsson, L., Nyström, M.arcus., Ardö, H., Åström, K., & Stridh, M. (2016). Smooth pursuit detection in binocular eye-tracking data with automatic video-based performance evaluation. *Journal of Vision*, *16*(15), 1–18.

Laurutis, V., & Zemblys, R. (2009). Bayesian decision theory application for double-step saccades. *Elektronika ir Elektrotechnika*, *92*(4), 99–102.

Lopez, J.SA. (2009). Off-the-shelf gaze interaction. Ph.D. Thesis.

Manning, C., Raghavan, P., & Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, *16*(1), 100–103.

Mital, P.K., Smith, T.J., Hill, R.L., & Henderson, J.M. (2011). Clustering of gaze during dynamic scene viewing is predicted by motion, Cognitive Computation 3, no. 1, 5–24.

Munn, S.M., Stefano, L., & Pelz, J.B. (2008). Fixation-identification in dynamic scenes: Comparing an automated algorithm to manual coding. In *Proceedings of the 5th symposium on applied perception in graphics and visualization*, APGV '08, (pp. 33–42). New York, NY, USA: Association for Computing Machinery.

Matthews, B.W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) – Protein Structure*, *405*(2), 442–451. https://www.sciencedirect.com/science/article/pii/0005279575901099.

Meyer, C.H., Lasker, A.G., & Robinson, D.A. (1985). The upper limit of human smooth pursuit velocity. *Vision Research*, *25*(4), 561–563.

Niehorster, D.C., Zemblys, R., Beelders, T., & Holmqvist, K. (2020). Characterizing gaze position signals and synthesizing noise during fixations in eye-tracking data, Behavior Research Methods.

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

Nyström, M., Andersson, R., Holmqvist, K., & Van De Weijer, J. (2013). The influence of calibration method and eye physiology on eyetracking data quality. *Behavior research methods*, *45*(1), 272–288.

Otero-Millan, J., Alba Castro, J.L., Macknik, S.L., & Martinez-Conde, S. (2014). Unsupervised clus5225 tering method to detect microsaccades, Journal of Vision 14, no. 2, 1–17.

Pekkanen, J., & Lappi, O. (2017). A new and general approach to signal denoising and eye movement classification based on segmented linear regression. *Scientific Reports 5230*, *7*(1), 1–13.

Peng, H., Li, B., He, D., & Wang, J. (2019). Identification of fixations, saccades and smooth pursuits based on segmentation and clustering. *Intelligent Data Analysis*, *23*(5), 1041–1054.

Rigas, I., & Komogortsev, O.V. (2017). Current research in eye movement biometrics: An analysis based on BioEye 2015 competition, Image and Vision Computing 58, 129–141.

Rigas, I., Komogortsev, O., & Shadmehr, R. (2016). Biometric recognition via eye movements: Saccadic vigor and acceleration cues. *ACM Transactions on Applied Perception (TAP)*, *13*(2), 1–21.

Salvucci, D.D., & Goldberg, J.H. (2000). Identifying fixations and saccades in eye-tracking protocols, Proceedings of the 2000 Symposium on Eye Tracking Research & Applications, pp. 71–78.

Schenk, S., Dreiser, M., Rigoll, G., & Dorr, M. (2017). GazeEverywhere: Enabling gaze-only user interaction on an unmodified desktop PC in everyday scenarios. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, (pp. 3034–3044). New York, NY, USA: Association for Computing Machinery.

Santini, T., Fuhl, W., Kübler, T., & Kasneci, E. (2016). Bayesian identification of fixations, saccades, and smooth pursuits. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, (pp. 163–170). New York, NY, USA: Association for Computing Machinery.

Startsev, M., Agtzidis, I., & Dorr, M. (2019a). 1D CNN with BLSTM for automated classification of fixations, saccades, and smooth pursuits. *Behavior Research Methods*, *51*(2), 556–572.

Startsev, M., Agtzidis, I., & Dorr, M. (2019b). Characterizing and automatically detecting smooth pursuit in a large-scale ground-truth data set of dynamic natural scenes. *Journal of Vision*, *19*(14), 10:1–10:25.

Startsev, M., Göb, S., & Dorr, M. (2019). A novel gaze event detection metric that is not fooled by gaze-independent baselines. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, ETRA '19, (pp. 1–9). New York, NY, USA: Association for Computing Machinery.

Startsev, M., & Zemblys, R. (2019). Discussion and standardisation of the metrics for eye movement detection. ETRA '19, Tutorial presented at the 11th ACM Symposium on Eye Tracking Research & Applications. https://etra.acm.org/2019/tutorials.html; slides available via https://emdetectionmetrics.page.link/etra2019.

Steil, J., Huang, M.X., & Bulling, A. (2018). Fixation detection for head-mounted eye tracking based on visual similarity of gaze targets. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, ETRA '18, (pp. 1–9). New York, NY, USA: Association for Computing Machinery.

Stuart, S., Hickey, A., Vitorio, R., Welman, K., Foo, S., Keen, D., & Godfrey, A. (2019). Eye-tracker algorithms to detect saccades during static and dynamic tasks: a structured review. Physiological Measurement, 40, 2.

Startsev, M., & Dorr, M. (2020). Supersaliency: A Novel Pipeline for Predicting Smooth Pursuit-Based Attention Improves Generalisability of Video Saliency. *IEEE Access*, *8*, 1276–1289.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*(3), 1–21.

Swan, G., Goldstein, R.B., Savage, S.W., Zhang, L., Ahmadi, A., & Bowers, A.R. (2020). Automatic processing of gaze movements to quantify gaze scanning behaviors in a driving simulator. Behavior Research Methods, pp 1–20.

Van Gompel, R.PG., Fischer, M.H., Murray, W.S., & Hill, R.L. (Eds.) (2007). *Eye movements: A window on mind and brain*. Amsterdam: Elsevier.

Voloh, B., Watson, M.R., König, S., & Womelsdorf, T. (2020). MAD saccade: Statistically robust saccade threshold estimation via the median absolute deviation, 5305 Journal of Eye Movement Research 12, no. 8.

Ward, J.A., Lukowicz, P., & Tröster, G. (2006). Evaluating performance in continuous context recognition using event-driven error characterisation, International Symposium on Location-and Context-Awareness, pp. 239–255.

Wadehn, F., Weber, T., Mack, D.J., Heldt, T., & Loeliger, H.-A. (2019). Model-based separation, detection, and classification of eye movements. *IEEE Transactions on Biomedical Engineering*, *67*(2), 588–600.

Zemblys, R., Niehorster, D.C., & Holmqvist, K. (2019). Correction to: "Using machine learning to detect events in eye-tracking data". *Behavior Research Methods*, *51*(1), 451–452.

Zemblys, R., Niehorster, D.C., & Holmqvist, K. (2019). gazeNet: End-to-end eye-movement event detection with deep neural networks. *Behavior Research Methods*, *51*(2), 840–864.

Zemblys, R., Niehorster, D.C., & Holmqvist, K. (2020). Evaluating three approaches to binary event-level agreement scoring. A reply to Friedman (2020). Behavior Research Methods.

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.