Human-level saccade detection performance using deep neural networks

Marie E. Bellet[1†], Joachim Bellet[2−4,†], Hendrikje Nienborg[2], Ziad M. Hafed[2,4*] & Philipp Berens[1,2,5*]

[1] *Institute for Ophthalmic Research, University of Tübingen, Tübingen, Germany*

[2] *Werner Reichardt Centre for Integrative Neuroscience, University of Tübingen, Tübingen, Germany*

[3] *International Max Planck Research School for Cognitive and Systems Neuroscience, Tübingen, Germany*

[4] *Hertie Institute for Clinical Brain Research, University of Tübingen, Tübingen, Germany*

[5] *Bernstein Center for Computational Neuroscience, Tübingen, Germany*

**co-corresponding authors*

†M.E.B. and J.B. contributed equally to this work

Saccades are ballistic eye movements that rapidly shift gaze from one location of visual space to another. Detecting saccades in eye movement recordings is important not only for studying the neural mechanisms underlying sensory, motor, and cognitive processes, but also as a clinical and diagnostic tool. However, automatically detecting saccades can be difficult, particularly when such saccades are generated in coordination with other tracking eye movements, like smooth pursuits, or when the saccade amplitude is close to eye tracker noise levels, like with microsaccades. In such cases, labelling by human experts is required, but this is a tedious task prone to variability and error. We developed a convolutional neural network (CNN) to automatically detect saccades at human-level accuracy and with minimal training examples. Our algorithm surpasses state of the art according to common performance metrics, and will facilitate studies of neurophysiological processes underlying saccade generation and visual processing.

New & Noteworthy

Detecting saccades in eye movement recordings can be a difficult task, but it is a necessary first step in many applications. We present a convolutional neural network (CNN) that can automatically identify saccades with human-level accuracy and with minimal training examples. We show that our algorithm performs better than other available algorithms, by comparing performance on a wide range of datasets. We offer an open-source implementation of the algorithm as well as a web service.

Introduction

Eye tracking is widely used in both animals and humans to study the mechanisms underlying perception, cognition, and action, and it is useful for investigating neurological and neurodegenerative diseases in human patients (Carpenter, 1988; Kowler, 2011; Leigh and Kennard, 2004; Leigh and Zee, 2015; MacAskill and Anderson, 2016). This is in part due to practical reasons: recording eye movements is relatively easy (Duchowski, 2007), while, at the same time, eye movements can be highly informative about brain state (Borji and Itti, 2014; Haji-Abolhassani and Clark, 2014).

The most prominent type of eye movement, in terms of eyeball rotation speed, is a ballistic shift in gaze position, called saccade. This type of eye movement occurs 3-5 times per second, and it can realign the fovea with interesting scene locations within only $\sim$50 ms. Naturally, saccades cause dramatic changes in visual input when they occur, and they therefore impact neural processing in different visual areas and also in a variety of ways (Burr et al., 1994; Colby et al., 1992; Crevecoeur and Kording, 2017; Golan et al., 2017; Reppas et al., 2002; Ross et al., 1997; Sommer and Wurtz, 2008; Yao et al., 2018; Zirnsak et al., 2014). This even happens for the tiniest of saccades, called microsaccades, that occur when gaze is fixed (Bellet et al., 2017; Bosman et al., 2009; Chen and Hafed, 2017; Gur et al., 1997; Hafed, 2011; Hafed et al., 2015; Hass and Horwitz, 2011; Herrington et al., 2009; Leopold and Logothetis, 1998; Yu et al., 2017). Therefore, studies not quantitatively analyzing microsaccades can miss important behavioral and neural modulations in experiments (Hafed, 2013). Saccades and microsaccades are, additionally, key discrete events in eye tracking traces that can be useful for parsing other eye movement epochs (e.g. smooth pursuits, ocular drifts, ocular tremors) for further analysis. Therefore, detecting saccades is typically the first step in any quantitative analysis of behavior or neural activity that might be impacted by these eye movements.

Several algorithms have been proposed for automating the task of saccade detection (reviewed in (Andersson et al., 2017)). For example, Engbert and Mergenthaler developed a method for classifying saccades and microsaccades based on an adaptive threshold (Engbert and Mergenthaler, 2006). This algorithm (which we refer to here as EM) is particularly popular because of its simple implementation and ease of use, as well as its ability to detect even microsaccades. However, this algorithm, like others, may still mislabel some microsaccades due to high eye tracker noise (as is typical with video-based eye trackers) as well as small catch-up saccades occurring during smooth pursuit. Other existing algorithms (Larsson et al., 2013; Pekkanen and Lappi, 2017) have the added advantage of providing additional labels for fixations and post-saccadic oscillations (PSO) in eye position.

Despite their success, several shortcomings still render the use of existing algorithms either less reliable than desired or, at the very least, cumbersome. While the performance of many published algorithms is promising (Andersson et al., 2017; Pekkanen and Lappi, 2017), it does not reach the level of trained human experts. Also, none of the existing algorithms show convincing performance for all eye movement-related events that may need to be analyzed (e.g. fixations, saccades, PSO, blinks, smooth pursuits). In addition, equipment-dependent hyperparameters, such as thresholds, need to be chosen for most algorithms, a fact that renders broad usability difficult. For example, even simple changes in eye tracking hardware, involving changes in sampling frequency or measurement noise, require re-tuning of such parameters. Re-tuning is also needed when the ranges of eye movement amplitudes being studied are modified (e.g. microsaccades versus larger saccades). Perhaps most importantly, objective parameter estimation in existing algorithms is currently a challenging task because of a limited amount of available reliably labelled data. Finally, in many cases, applying available online resources is not straightforward. As a result of all of the above shortcomings, current laboratory practice often still involves experimenters spending substantial amounts of time to carefully relabel at least parts of their data after automatic saccade detection.

Here we propose a convolutional neural network (CNN) for classifying eye movements. The architecture of the network is inspired by U-Net, which has successfully been used for image segmentation (Ronneberger et al., 2015). We evaluated our network (U'n'Eye) on four challenging datasets containing small saccades occurring during fixations or smooth pursuits. On these datasets, U'n'Eye reached the performance level of human experts in labelling saccades and microsaccades, while being much faster. The network also beat state-of-the-art algorithms on a benchmark dataset not just for saccade detection, but also for PSO. As we show here, our network can be trained quickly, even on a standard laptop, and with minimal amounts of training data. More importantly, our network's adaptability to different datasets makes U'n'Eye the novel state-of-the-art eye movement detection algorithm. We provide an easily accessible web service for running U'n'Eye (`http://uneye.berenslab.org`), as well as an open source implementation (`https://github.com/berenslab/uneye`). Our labelled datasets will also be freely available upon publication.

Methods

Datasets. All experiments used for collecting the datasets were approved by ethics committees at Tübingen University. Human subjects provided informed, written consent in accordance with the Declaration of Helsinki. Monkey experiments were approved by the regional governmental offices of the city of Tübingen.

Dataset 1 was collected from human subjects using the Eyelink 1000 video-based eye tracker (SR Research, Ltd) sampling eye position at 1 kHz. The dataset contains mostly microsaccades and small-amplitude memory-guided saccades. It contains 2000 trials of 1 second. Out of these 2000 trials, 1000 were selected to compare U'n'Eye to other algorithms via cross-validation (Fig. 4). We named these trials "set1A". When testing for the impact of missing labels on performance (Fig. 7B), we used the other 1000 trials, "set1B", to train networks and tested them on set1A.

Dataset 2 was collected from three male, rhesus macaque monkeys implanted with scleral search coils (in one eye for each of the monkeys). The dataset contains catch-up saccades generated during smooth pursuit. Eye position was again sampled at 1 khz. For the trials containing smooth pursuit of sinusoidal target motion trajectories in this dataset, the data were obtained from the experiments described in (Hafed et al., 2008; Hafed and Krauzlis, 2008). For the trials containing pursuit of constant speed, the experimental conditions are described in (Buonocore et al., 2018). Eye movement calibration for search coil data was done according to the procedures in (Tian et al., 2016). The overall dataset consists of 2000 segments of 1 second of eye traces. Like in the case of dataset 1, we split the set into two sets of 1000 segments each, "set2A" and "set2B". set2A was used to compare U'n'Eye to Daye and Optican's (Daye and Optican, 2014) algorithm (Fig. 4).

Dataset 3 was collected from a single male macaque monkey using the Eyelink 1000 video-based system sampling eye position at 500 Hz. The dataset contains microsaccades generated during fixation. The data was obtained from experiments described in (Kawaguchi et al., 2018). It consists of 403 segments of 1.438 seconds. Similarly to datasets 1 and 2, we split the data in two subsets, "set3A" and "set3B". Set3A contains 350 segments and set3B 53 segments. Set3A was used for comparing U'n'Eye's performance to other algorithm's (Fig. 4).

For the results shown in Fig. 7D, we used setA of all datasets for training and the respective setB for testing.

Dataset 4 was collected from the same eye-tracker as dataset 1 but with different sets of subjects. It comes from a recently published study (Bellet et al., 2017) in which subjects had to keep fixation at the center of the screen before a peripheral target appearance. We selected 630 segments of 750 ms from each of 10 subjects (4725 seconds in total). The dataset includes not only successful trials, in which subjects maintained fixation, but also trials containing blinks or saccades outside of the fixation window. Again, we split the dataset into 2 subsets. Set4A contained 330 segments per subject and was used to train networks. Set4B contained 300 segments per subject and was used to test the performance of the networks.

In all datasets, we manually detected saccades using a custom-made GUI in Matlab. The

GUI displayed horizontal and vertical eye position traces, as well as filtered radial eye velocity. The GUI internally estimated saccade onset and end times using a combination of velocity and acceleration thresholds (Chen and Hafed, 2013). The user then manually interacted with the GUI to delete false alarms, correct false negatives, and adjust estimation of onset and offset timing.

Simulated saccades. To test the performance of our network on noisy labelled data, we designed artificial eye traces for which we knew the ground truth. Saccades ranging from 0.5 to 60° were simulated using an adaptation of a model for saccade waveforms (Dai et al., 2016). The model is a sum of a soft ramp functions, which follows the relationship between amplitude and peak velocity observed in real saccades (Dai et al., 2016). Since the model is originally one dimensional, we adapted it so that it generates two dimensional trajectories. Saccade generation in time was made to follow a Poisson process with λ equal to 3 saccades per second. Simulated blinks were also added by inducing sharp transients in the eye traces. Finally, a Gaussian white noise with a standard deviation of 0.02° was added to the trace. Then, as described in Results, we trained U'n'Eye under a variety of conditions in which we intentionally removed a subset of saccade labels during training, in order to explore robustness to missing labels (Fig. 7).

U'n'Eye: our convolutional neural network. The architecture of the convolutional neural network (CNN) was inspired by U-Net, a CNN first used for image segmentation (Ronneberger et al., 2015). Here we modified U-Net to meet the requirements of an eye movement classifier. The network was built of seven convolutional layers with kernel size 5, each followed by a linear-rectifying unit (ReLU) and a BatchNorm layer, both described in detail in Results. Batches consisted of samples of the same duration. The input to the network was eye velocity which was computed as the first-order difference of the eye position signal. The input was of dimension N x T x 2, where N is the batch-size, T the number of time-points, and 2 the number of coordinates (horizontal and vertical eye velocity). The number of input time-points could be variable but had to be a multiple of 25 bins due to the max pooling operations. The output of the network was a matrix of dimension N x K x T, where K was the user-defined number of classes. For example, we could have a "saccade" and "fixation" class in the networks of Fig. 3 and we could also add other classes like "PSO" in the network of Fig. 6.

We applied a softmax (Christopher, 2016) activation function to the output of the last convolutional layer x:

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{1}$$

where x_i is the layer corresponding to class i. Thus, the network's output y represented the sample-by-sample conditional probability of each class (e.g. "fixation" or "saccade") given the eye-velocity x and the network weights w:

$$Y_k = p(k = 1|x, w) \tag{2}$$

The final prediction of the algorithm represented the class that maximized this conditional probability:

$$\hat{k} = argmax_k \, p(k = 1|x, w) \tag{3}$$

We chose the kernel sizes of the convolutional and max pooling operations in a way to capture a relevant signal range around each time-point. Based on the given kernel sizes of the network, it can be shown that the prediction of one time-bin is influenced by the preceding and following 89 time-bins of the velocity signal (2B, red color).

Network training. We trained the network with mini-batches whose size depended on the total number of training samples. We performed 10 training iterations in each epoch. Over-fitting on the training set was prevented by computing the loss on a validation set and stopping training when the validation loss increased for three successive epochs. We used a multi-class error function, which, for two classes, equals the cross entropy loss. Weight-regularization was done with L2-penalty (Christopher, 2016), which corresponds to a Gaussian prior with zero mean over the network weights. The optimal parameter λ was determined to be 0.01. The loss function was thus defined as:

$$L = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\, log\, y(x_n\,, w) + \lambda\, ||w||_2^2 \tag{4}$$

where N is the number of time points and K the number of classes. The ground truth label t_{nk} equals 1 if the time point n belongs to class k. Gradient computation was done with PyTorch autograd method.

We used the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.001. Adam is a stochastic gradient-based optimizer that uses adaptive learning rates for different weights of the network. An additional step-decay by a factor of 2 was applied to the current learning rates when the loss on the validation set increased during one epoch.

Post-processing. In the case of binary prediction into the classes fixation and saccade, we provided the possibility to define thresholds for minimum saccade duration and minimum saccade distance. If thresholds were given, saccades closer than the minimum distance were merged and saccades shorter than the minimum duration were removed. We obtained the results reported here with a minimum saccade distance threshold of 10 ms for dataset 1, 3 and 4 and 5 ms for dataset 2, because we previously observed that some saccades occurred very close in time in this dataset. For datasets 1-4, we used a minimum saccade duration threshold of 6 ms. The same thresholds were used for the algorithm by Engbert & Mergenthaler (Engbert and Mergenthaler, 2006).

Data augmentation. U'n'Eye performs better with a bigger training set. However, we aimed to reduce the amount of saccades that a user should provide to train U'n'Eye. In this study, to increase the number of training samples, the input eye positions were rotated and added to the original training samples:

$$x_2 = xcos(\theta) + ysin(\theta) \tag{5}$$

$$y_2 = -xsin(\theta) + ycos(\theta) \tag{6}$$

where x and y are the horizontal and vertical eye positions. We used $\theta = (1/4\pi, 3/4\pi, 5/4\pi, 7/4\pi)$ radians. Thus, we could increase by five fold the size of our training set without causing over-fitting.

Performance measures To evaluate the eye movement detection performance of our network, we used the following metrics: Cohen's kappa, F1 score, and onset and offset time differences.

Cohen's kappa is a sample-based statistic. It reflects how much two coders agree on the class that each time-bin belongs to, while controlling for chance agreement of the two coders. It is given by:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{7}$$

where p_0 is the proportion of time-bins for which two coders agree, and p_e is the proportion of time-bins for which agreement can be expected by chance.

For a binary classification of fixation versus saccades, the Cohen's kappa value p_e is given by

$$P_e = \frac{1}{N^2} \times \sum_{k=1}^{K} nk_{coder1} \times nk_{coder2} \tag{8}$$

where nk_{coderX} is the number of time-bins coder X assigned to class k.

The F1 score is a measure of classification accuracy that combines precision and recall of a predictor. Precision is defined as the proportion of correctly classified saccades over all predicted saccades. Recall is defined as the proportion of correctly classified saccades over all saccades in the ground truth. The F1 score is the harmonic mean of these two measures. It is given by:

$$F1 = 2 \times \frac{TP}{2 * TP + FN + FP} \tag{9}$$

where TP is the number of true positives, FN the number of false negatives, and FP the number of false negatives. For all true positive saccades, we compared saccade timing between the ground truth and prediction by calculating the absolute time differences between true and predicted saccade onsets and offsets.

Evaluation on a benchmark dataset We evaluated U'n'Eye performance on a benchmark dataset by Andersson et al. (Andersson et al., 2017). This dataset comprises 500 Hz eye-tracking data from humans looking at images, movies, or moving dots. It contains human labels for the events fixations, smooth pursuits, saccades, PSO and blinks. Events that the human experts did not assign to any of these classes were labelled as "others". For some trials, the dataset contained labels from two different human coders. For other trials, only one label was available. We trained 20 independent networks with different random initializations on the data with labels from one human coder (coder RA). Performance was then tested on the trials with labels from two coders, which makes our result comparable with previously reported results (Pekkanen and Lappi, 2017). Note that we were not able to reproduce the inter-rater measures reported by Andersson et al. (Andersson et al., 2017) in line with the results of Pekkanen and Lappi (Pekkanen and Lappi, 2017). For comparability with the NSLR-HMM algorithm (Pekkanen and Lappi, 2017), we excluded the event labels "other" for the calculation of Cohen's kappa scores. The performance on the class "blinks" was not compared to other algorithms since it was not reported.

Evaluation of other algorithms We compared U'n'Eye perfomance on our datasets to several already published algorithms. For datasets 1 and 3, which contain microsaccades occuring during fixation of a static target, we evaluated the performance of three algorithms designed for microsaccade detection (Engbert and Mergenthaler, 2006; Otero-Millan et al., 2014; Sheynikhovich et al., 2018) and one algorithm designed for saccade detection in a high noise regime (Pekkanen and Lappi, 2017).

The algorithm by Engbert & Mergenthaler (Engbert and Mergenthaler, 2006) is commonly used as an unsupervised method to detect microsaccades. It selects saccades based on a threshold that depends on the level of the noise in the velocity. One parameter, called λ, can be also be fit to the data to obtain better results. λ is multiplied with the velocity noise in order to determine a threshold for saccade selection. Here we chose λ values that maximized the metric of interest on the training data from our cross-validations. This was done in order to give this algorithm the benefit of the doubt in our comparisons. Importantly, prior to saccade detection, we smoothed the eye traces using a 5-point average independently of the sampling frequency, as described by Engbert & Mergenthaler (Engbert and Mergenthaler, 2006).

The approach from Otero-Millan et al. (Otero-Millan et al., 2014) is an unsupervised method. It gives an estimate of saccade onset and offset timing and thus can be compared in terms of both the Cohen's kappa and F1 metrics.

Another unsupervised method has been developed by Sheynikhovich et al. (Sheynikhovich et al., 2018). This algorithm only gives an estimate of microsaccade occurence at one point in time without determining onset and offset. We thus compared only the performance in terms of F1 score for this algorithm. We considered as true positive any saccade detected $\pm$ 10 ms away from a ground truth saccade.

For dataset 2, we evaluated the performance of the method by Daye and Optican (Daye and Optican, 2014), which uses particle filters to detect saccades embedded in high velocity eye movements. The algorithm was kindly provided by the authors. To increase performance, we detected saccades independently in the horizontal and vertical channel and then merged the predictions.

This is because the Daye and Optican algorithm only considers as a saccade an event crossing a threshold both in horizontal and vertical components at the same time, which increases the number of false negatives. To increase performance, the parameters were tuned differently for trials containing sinusoidal pursuit than for those containing linear pursuit, again to give the algorithm the benefit of the doubt when comparing to U'n'Eye. To detect saccades in sinusoidal pursuit, the parameters were set to $\Omega = 10^{-4}$, $\xi = 3 \,.\, 10^{-4}$, N = 100, m = 20, $\lambda = 5 \,.\, 10^{-3}$, $\psi = 2 \,.\, 10^{-3}$. To detect saccades in linear pursuit, the parameters were set to $\Omega = 10^{-3}$, $\xi = 3 \,.\, 10^{-3}$, N = 100, m = 20, $\lambda = 5 \,.\, 10^{-3}$, $\psi = 10^{-4}$.

For all unsupervised algorithms, the 10 testing subsets from the cross-validation data were evaluated at once to yield better clustering estimates.

The results of the algorithm by Pekkanen and Lappi (Pekkanen and Lappi, 2017) on datasets 1 and 3 were obtained by estimating the model's parameters via cross-validation using the same training folds as for U'n'Eye. The estimated parameters were kindly provided by the authors.

Compute time The computation times of our algorithm reported here were achieved on a personal computer with a 2 GHz Intel Core i5 processor at 8 GB RAM running on Mac OS X 10.11.6.

Code and data availability A web service for running the algorithm is available at `http://uneye.berenslab.org`. All code is available from `https://github.com/berenslab/uneye`. Data will be available upon publication.

Results

Design of a convolutional neural network (CNN) for eye movement classification. We developed a CNN that predicts the state of the eye for each time point of an eye trace. The aim of the network was to segment eye movement recordings (Fig. 1) into epochs containing saccades/microsaccades (orange highlights) versus epochs not containing these eye movements (but see also below for additionally classifying PSO using our network). Our primary goal was to have a network that can seamlessly handle the challenging scenarios of tiny microsaccades during fixation (Fig. 1A), small catch-up saccades embedded in relatively high smooth pursuit eye velocities (Fig. 1B), and microsaccades and saccades occurring in recordings with higher noise levels associated with video-based eye trackers when compared to, say, scleral search coil techniques (Fuchs and Robinson, 1966; Judge et al., 1980) (Fig. 1C). We therefore trained and tested the network on three different challenging datasets (see Methods and Table 1), which contain labels for fixations and saccades manually determined by human experts. To test the ability of our network to generalize across eye movement traces recorded from different individuals, we also included a fourth dataset (Fig. 1D), which was obtained from 10 different subjects using the same eye tracker as in dataset 1 (Methods). Testing generalizability was also achieved using with a fifth and final dataset containing artificially generated noisy eye movement traces, in which the ground truth for saccade times was known (Methods) (Fig. 1E). Finally, we compared our network's performance to different existing algorithms, both on our datasets and also on a publicly available benchmark dataset (Larsson et al., 2013) (`http://dev.humlab.lu.se/www-transfer/people/marcus-nystrom/annotated_data.zip`).

The network operates on the eye velocity signal and requires no other preprocessing. Eye velocity is computed as the differential of eye position (Methods), and chunks of eye velocity signals are then input to the network. Briefly, the network's architecture is based on the U-Net, a CNN for pixel-by-pixel image segmentation (Ronneberger et al., 2015), which we modified to process one dimensional signals and output a predictive probability for each eye movement class at every time point (Fig. 2A). A major change compared to the original U-Net architecture is that we introduced batch normalization (BatchNorm) layers (Klibisz et al., 2017). BatchNorm layers subtract a mean from their input and divide it by a standard deviation. Both of these parameters are estimated for each layer over mini-batches of training samples during learning. This method normalizes the distribution of activations across the network layers, allowing for higher learning rates and reducing over-fitting (see below) (Ioffe and Szegedy, 2015). We also applied a rectified linear unit (ReLu) function between each convolutional and batch normalization layer. The ReLu function, or heaviside step function, introduces nonlinearities in the network, allowing it to apply arbitrary-shaped functions to the input data. Finally, the U-shaped architecture of the network leads to temporal downsampling and upsampling in the hidden layer representations (Fig. 2). Downsampling is achieved by max pooling (MaxPool) operations that reduce the dimensionality of the network content, extracting relevant features. Upsampling is realized by transposed convolution. Convolutional kernels and max pooling operations together lead to the integration of information over time. Due to the network design, the probability assigned to each time bin can be influenced by ± 89 preceding and following time bins (Fig. 2B). Thus, U'n'Eye takes into account a large enough signal in order to make point predictions of the correct eye movement class.

U'n'Eye achieves human-level performance. Our network achieved human-level performance after training on our datasets. We first illustrate this with three example scenarios for detecting saccades (Fig. 3). For illustrative purposes, we also show how the commonly used algorithm (EM) might perform for the examples; we later provide an exhaustive quantitative comparison with several more algorithms (Fig 4). In the first example, a small microsaccade occurred with substantial oscillation in eye position towards movement end, and with the amplitude of the movement being near the eye tracker noise level (Fig. 3A). Human coder 1 considered the post saccadic oscillation as part of the saccade, and so did our network trained on his training set (compare the binary classification output of the coder and Network 1 below the eye movement traces in Fig. 3A). On the other hand, coder 2 determined that the saccade ended earlier, and our network trained on his training set did the same (again, compare the classification output for Human coder 2 and Network 2). Thus, our network could match the criterion used by an individual human coder very well. Moreover, our network successfully avoided a false detection by the EM algorithm on these traces. In the second example, the EM algorithm missed all three saccades, which is perfectly reasonable since this algorithm was never designed to work in association with smooth pursuit eye movements, but our network successfully flagged them (Fig. 3B). Finally, the eye movement in the third example was collected with a video-based eye tracker having substantially more noise (Fig. 3C). In this case, one false detection made by the EM algorithm was successfully excluded by our network.

To present more quantitative performance measures, we first tested our network on our in-house datasets (Fig. 1) and compared its performance to that of commonly used or recently published algorithms . For our network, we performed 10-fold cross-validation separately for datasets 1-3. In each cross-validation round, 90% of the data was used for training the network, and the remaining 10% were used to test performance. A separate validation set from each dataset was used to detect over-fitting of the network. To prevent such over-fitting, we regularized the weights of the network using the L2 penalty (Christopher, 2016) (Methods), preventing the parameters of the network from deviating excessively from zero. Furthermore, we made use of early stopping. For this, a separate validation set was used, and the validation set error was computed in each epoch. Training was stopped at the point of smallest validation set error. For datasets 1 and 2, 950 sec of eye traces were used for cross-validation and 50 sec for validation. Thus, each training set contained 855 sec of data. For dataset 3, 330 sec were used for cross-validation and 23 sec for validation, resulting in 297 sec of data in each training set. For the other algorithms that we tested, we used the same cross-validation approach in the case of supervised algorithms (EM (Engbert and Mergenthaler, 2006), Pekkanen & Lappi (Pekkanen and Lappi, 2017)). Note that we used the EM algorithm as a supervised method since we fitted its single parameter on our training data (Methods). For unsupervised methods (Otero-Millan et al. (Otero-Millan et al., 2014), Sheynikhovich et al. (Sheynikhovich et al., 2018), Daye & Optican (Daye and Optican, 2014)), the identical 10 test sets were evaluated without using the training set (Methods).

Finally, similarity of the algorithms' predictions to human labels was evaluated using three metrics. First, we calculated Cohen's kappa, which is a sample-by-sample similarity measure that takes chance agreement of two predictors into account (Cohen, 1960). Second, we calculated the F1 score, which is an accuracy measure that considers precision and recall of a classifier.

Recall corresponds to the number of correctly detected saccades divided by the number of saccades that were labelled by the human expert. Precision, on the other hand, is the number of correctly classified saccades divided by the total number of saccades detected by the classifier (Methods). The F1 score is defined as the harmonic mean of both, and it thus only measures how accurate saccades were detected without taking into account their timing (i.e. exact saccade onset and offset times). Correctly labelling saccade onset and offset can be crucial for further analyses. Therefore, for our third and final metric, we additionally computed the absolute time difference in onset and offset of correctly predicted saccades and of saccades labelled by the human experts. This measure reflects how well an algorithm agrees with the human coder in terms of saccade start and end.

U'n'Eye reached high similarity to the human coder (Fig. 4A, B, blue) and outperformed all the other compared algorithms (Fig. 4A, B). U'n'Eye also detected saccade onset and offset in high agreement with the human labels. On average, saccade onset differences to human labels were smaller than 3 ms, and saccade offset differences were smaller than 4 ms. Saccade onset and offset labels by the other algorithms deviated more strongly from the human-labelled saccades (Fig. 4C, D ; Table 2). This indicates that U'n'Eye's saccade predictions were more human-like.

In the more challenging dataset 2, in which saccades occurred during smooth pursuit eye movements, U'n'Eye outperformed the algorithm by Daye and Optican (Daye and Optican, 2014), which was designed to overcome this difficulty. Here, saccade peak velocity was close to the instantaneous velocity of the ongoing smooth pursuit movements. In fact, the minimum saccade peak velocity in this dataset was smaller than the median instantaneous velocity during pursuit

(Table 1). Yet, U'n'Eye succeeded in detecting such saccades. This was because the network architecture utilized a substantial time window (Fig. 2), allowing it to infer changes in the state of the eye even if the instantaneous velocity is low compared to the surrounding eye trace.

We next addressed the question of whether U'n'Eye can achieve a similar level of inter-human agreement when multiple human experts analyze the same data. For this, we used dataset 1 because, among the four datasets, it contained saccades with the widest range of amplitudes (from as small as 0.02° up to a size of 11°; see Table 1 for a reason why saccades as small as 0.02° were possible). We could thus assess inter-rater agreement for a broad range of saccades. Dataset 1 was labelled by a second independent human coder (Fig. 3, upper panel; Fig. 4, dataset 1). Coder 1 estimated saccade timing based on a combination of the raw eye traces and the smoothed radial velocity, whereas Coder 2 used the raw eye traces only. We trained independent networks either with labels from Coder 1 or Coder 2 (Network 1 and Network 2, respectively), and we tested the networks' performance on the 10 test sets from the 10-fold cross validation routine described above, both against ground truth labels from Coder 1 or Coder 2. U'n'Eye's saccade labels were as similar to both human coders as the human labels were to each other (Table 3). In terms of the F1 score, the inter-human agreement was not significantly different from the network-human agreement (Table 4). Interestingly, Network 1 showed higher similarity scores than Coder 2 when both were compared to labels of Coder 1 in the test sets, and vice versa for Network 2 and Coder 2. This is reflected by larger Cohen's kappa scores and smaller onset and offset differences (Table 4, all $p < 5{\cdot}10^{-5}$ after Bonferroni correction for multiple comparisons, Student's paired samples t-test for Cohen's Kappa and F1 scores, and independent samples t-test for on- and offset differences).

This indicates that U'n'Eye's saccade estimation surpasses inter-rater consistency.

U'n'Eye misses only a small fraction of microsaccades We then analyzed the patterns of agreement and disagreement between U'n'Eye and human labelling. For true positive saccades, the two dimensional histogram of detected movements reflected the typical main sequence relationship between peak velocity and amplitude of saccades (Fig. 5 A,D,G) (Zuber et al., 1965). A few false positives were present within the range of the main sequence, suggesting that the human coder forgot to label some saccades (for example, see the movement in the inset in Fig. 5B). Concerning the rare false negatives that occurred, some of them had fairly large amplitudes (beyond eye tracker noise). Closer inspection revealed that there were pairs of successive saccades that had very short inter-saccadic intervals. The network lumped them into one movement, whereas the human coders separated them. Most remaining disagreements between the human and the network were associated with the smallest microsaccades, closest to eye tracker noise levels.

U'n'Eye: new state-of-the-art eye movement classifier. In order to compare our algorithm to state-of-the-art methods for eye movement classification, we next evaluated its performance on a benchmark dataset (Larsson et al., 2013), which has previously been used for the comparison of 12 eye movement classifiers (Andersson et al., 2017; Pekkanen and Lappi, 2017). The dataset comprises 500 Hz eye tracking recordings from humans watching videos, images, or moving dots, and it contains human labels for fixations, smooth pursuits, saccades, PSO (Fig. 6A), and blinks. We therefore used U'n'Eye as a multi-class classifier to predict saccades, PSOs, and blinks (Fig. 6B). Fixations and smooth-pursuit eye movements were both assigned to the fixation class. U'n'Eye output a predictive probability for each class (Fig. 6D), with the prediction value corresponding to the class that maximized this predictive probability (Fig. 6C). We trained U'n'Eye on one part of the data and evaluated its performance on the test trials listed in Andersson et al. (their Table 11 (Andersson et al., 2017)). When considering the whole benchmark dataset, U'n'Eye outperformed the state-of-the-art classifiers for saccades and PSOs (Table 5). Moreover, U'n'Eye's performance lied within the range of the inter-coder agreement of the two human experts who labelled the dataset (Table 5). This result indicates that U'n'Eye is very well suited for multi-class eye movement classification.

Practical considerations for U'n'Eye usage. To better understand the practical aspects of using our approach, we additionally assessed how U'n'Eye performs under different training scenarios. The results of this section can be used as good practice guidelines by the users in their own applications that employ our algorithm.

First, we studied how the amount of training data impacts saccade detection performance. In practice, the available number of annotated training samples might be limited. In order to achieve good performance of U'n'Eye, small training sets were sufficient (Fig. 7A). Even with only 50 seconds of labelled data, our network outperformed other algorithms. Using more training samples led to a further increase of performance. Training time was also no limiting factor, since training a new network even on a CPU took only about 2 minutes for every minute of training data (Fig. 7A).

In machine learning, the quality of the training data is also crucial for the performance of a classifier, since the latter directly learns from the human ground truth labels. Human labelling, however, is prone to mistakes and lapses: saccades might be missed by the human coder, leading to noisy labels. We therefore assessed how U'n'Eye's performance was influenced by noise-corrupted labels. We evaluated the network's performance when trained on real data (dataset 1) from which we artificially removed a fixed fraction of saccade labels. U'n'Eye was robust to the presence of noisy labels in the training data: even with 20% of missing labels, our network outperformed other algorithms (Fig. 7B). We also trained the network on simulated data (Fig. 1E) for which we knew the ground truth. While noise-corrupted labels in the training data impaired saccade detection performance as expected, this effect could be compensated for by using a larger amount of training data (Fig. 7C). This indicates that U'n'Eye can achieve good performance even if the human coder misses some saccades in the training set.

Next, we studied how well an already trained network can be applied to label saccades in new data, for which no training labels are available. Our results show that this is possible if the new data has broadly the same signal characteristics as the data used for training the network (i.e. if it was sampled with the same eye tracker during a sufficiently similar task). To illustrate this, we trained networks on our first three datasets and evaluated their performance on each dataset plus an additional dataset 4 on which none of the networks was trained. Dataset 4 was similar to dataset 1 in that it was recorded with the same eye tracker in human subjects performing fixations (Table 1). Therefore, a network trained on dataset 1 performed very well not only in detecting saccades in the same dataset, but also in dataset 4 (Fig. 7D). Overall, good performance was guaranteed when the test set exhibited similar statistics as the training set, or was exposed in training to a sufficiently wide variety of training samples (Fig. 7D, last column).

Likewise, our network extrapolated well over subjects, for example in large cohort studies with many different observers (as is often the case in clinical investigations of neurological diseases). We studied whether a network trained on data from one subject was able to detect saccades well in data from another subject. To this end, we trained separate networks on data from ten individual human subjects in dataset 4 and applied them to all other subjects. Overall, performance on data that came from the same subject as the training data was only marginally higher than performance on data that came from a different subject (F1 mean $\pm$ std: 0.96 ± 0.01 vs. 0.92 ± 0.08, Fig. 7E). The higher standard deviation of inter-subject performance was due to the apparent difference between data from certain subjects (Fig. 7E). We therefore advise users to combine training data from a few subjects in order to obtain a network that is able to deal with different signal statistics (Fig. 7E, network trained on all). Note that for the network trained on a combination of subjects, we made sure to keep the number of training samples the same as for networks trained on individual subjects. Thus, the better performance was a result of having more variable samples in the training set and not of more training examples being available.

Eye movement representation becomes disentangled along network layers. We finally had a closer look at how the network achieves the separation of two eye states (e.g. fixations and saccades; Fig. 8A). In the velocity domain, saccades and fixations can show highly overlapping distributions (Fig. 8B). This explains why velocity threshold-based algorithms can fail to distinguish fixations from saccades (Fig. 4). Here, we showed that U'n'Eye can differentiate between fixations and saccades with high accuracy (Fig. 4). The classification was based on the output layer of the network. To illustrate how this decision arises throughout the hidden layers, we performed principal component analysis (PCA) on the features of each convolutional layer. The fraction of explained variance by the first two principal components (PCs) reflects the U-shaped architecture of the network (Fig. 8C): in the middle layers, information is distributed across more components than in early and late layers. We projected the hidden layer activations onto the PC space and labelled time bins according to their ground truth labels (fixation or saccade, Fig. 8D). We observed in higher layers that the two classes were better separated (Fig. 8D). Finally, in the output layer, fixations and saccades became linearly separable (Fig. 8E). Thus, through training, the network effectively learns to extract relevant features and to project those onto a plane where the two eye movement classes are linearly separable.

Discussion

In this study, we presented U'n'Eye, a convolutional neural network for eye movement classification. We demonstrated that U'n'Eye achieved human-level performance in the detection of saccades and microsaccades. In addition, the network was able to predict other classes of eye movements, which we exemplified with the detection of blinks and PSOs in a benchmark dataset.

Furthermore, we showed that U'n'Eye achieved excellent performance both when trained on a single type of data with labels from one coder and when trained on different datasets with labels from two coders. While datasets 1 and 3 used in this study contained data with only one type of visual task and labels from one coder each, dataset 2 was composed of two different pursuit tasks and contained labels from two different human coders. Moreover, dataset 4 allowed us to conclude that our algorithm has generalization properties, and can therefore be used when training on a subset of individuals and then testing with large cohorts of subsequent subjects measured with similar eye tracking technology. The dataset by Andersson et al. also contained greatly varying types of saccades and other eye movements. Still, U'n'Eye achieved good performance when trained and tested on this dataset. Note that the network might fail to detect eye movements when tested on data that show a very different distribution than the data it was trained on. We therefore recommend to either train a network with a variety of data or to train separate specialized networks for each task.

In this regard, our approach falls in the class of supervised learning algorithms, as opposed to methods not requiring parameter estimation based on annotated data (Engbert and Mergenthaler, 2006; Otero-Millan et al., 2014; Sheynikhovich et al., 2018). However, we typically see in different scenarios that casting an algorithmic issue as a supervised problem helps in terms of performance. For example, we recently showed that supervised techniques perform as well as, or better than, unsupervised ones for spike inference from calcium imaging data (Berens et al., 2018; Theis et al., 2016). Similarly, Mathis et al. recently showed that supervised learning provides superior animal tracking with few annotated samples (Mathis et al., 2018). We showed here that the situation is similar for eye movement detection. Importantly, we showed that performance generalizes to new unseen data sets and subjects, yielding better performance than any of the unsupervised algorithms. Of course, there is some manual work involved in preparing the training samples for our network, but we posit here that this amount of manual work is significantly less intensive than the manual post-processing that we typically perform with other saccade detection algorithms.

U'n'Eye is publicly available and provides a user friendly interface as well as a web service in which users can upload their data and receive classification outputs (Methods). No parameter tuning is needed even for training (e.g. learning rate, and so on) since the standard settings were found to work well across datasets. Instead, an experimenter just needs to provide a few hundred seconds of labelled data to train the network once. Even if some labels are missing in the training data, U'n'Eye can still reach high performance. We recommend, however, to use only carefully annotated data for training, as this will improve results.

Of the few algorithms that are capable of detecting saccades as well as PSO (Larsson et al., 2013; Pekkanen and Lappi, 2017; Zemblys et al., 2017), U'n'Eye achieves highest performance.

Note that Zemblys et al. also recently proposed a deep learning method for eye movement detection (Zemblys et al., 2018). Their approach consists of generating a large training set out of a small human-labelled dataset using a generative neural network. A second network is then trained on this data to classify eye movements. This method reports performance similar to that of U'n'Eye in a subset of the benchmark dataset by Andersson et al. (Andersson et al., 2017), but it remains to be seen how this algorithm performs on more exhaustive tasks like the ones that we reported here. For example, the applicability to data containing smooth pursuit has not been demonstrated. Conversely Startsev et al. (Startsev et al., 2018) recently published a deep learning approach showing reasonable performance, but again tested only on a subset of the benchmark dataset containing smooth pursuit.

Recently, a Bayesian approach for the detection of microsaccades based on a generative model has been proposed (Mihali et al., 2017). Inherently, Bayesian methods provide estimates of uncertainty, in addition to estimates of the quantity of interest. Indeed, it is an interesting future perspective to combine U'n'Eye with Bayesian Deep Learning techniques to provide uncertainty estimates for the detected eye movements (Gal and Ghahramani, 2015).

Future work will include combining datasets with different characteristics, such as different sampling frequencies, in order to obtain a network that can generalize on a large range of data. Such a network could be used by a large part of the scientific community, which would allow for reproducibility of scientific results. We recommend that anyone who uses our algorithm to publish the weights of the trained network so that eye movement detection can be reproduced. For our own trained networks, all weights have been published online (`https://github.com/berenslab/uneye`) along with the code of the network. This has the advantage that users with similar data characteristics to one of our three datasets (e.g. microsaccades during fixation with a video-based eye tracker as in dataset 3) can directly use our weights from the proper dataset without having to retrain their own network. We will also make all three datasets publicly available, facilitating the further development for eye movement detection algorithms.

Of course, it should be noted that some prediction errors may still occur with U'n'Eye. However, such errors fall within the range of inter-rater variability across humans anyway. Also, even when U'n'Eye does make mistakes, the predictive probability that it outputs can be used to retrieve missed events (e.g. see the black arrow in Fig. 6). For example, detecting peaks in the predictive probability output that did not cross the threshold can accelerate eventual manual post-processing.

Finally, U'n'Eye's capacity to learn non-linear relationships between an eye trace and some annotated labels opens new horizons in neuroscience: the network could be used to understand the properties of neural activities related in a complex manner to eye movements. For example, the disentanglement in later layers (Fig. 8) could be used to quantitatively analyze the activity patterns of pre-motor neurons in the brainstem, which themselves ultimately transform brain processing into individual ocular muscle innervations. Furthermore, U'n'Eye could be turned into a generative model for eye movements, as was shown for neural networks that are used for image classification (Gatys et al., 2015). The information about eye movements that is contained in the network architecture might in the future be used to identify variations in eye movement characteristics that could hint at underlying pathologies.

References

Andersson R, Larsson L, Holmqvist K, Stridh M, Nyström M. One algorithm to rule them all? an evaluation and discussion of ten eye movement event-detection algorithms. *Behavior research methods* 49: 616–637, 2017.

Bellet J, Chen CY, Hafed ZM. Sequential hemifield gating of α-and β-behavioral performance oscillations after microsaccades. *Journal of neurophysiology* 118: 2789–2805, 2017.

Berens P, Freeman J, Deneux T, Chenkov N, McColgan T, Speiser A, Macke JH, Turaga SC, Mineault P, Rupprecht P et al. Community-based benchmarking improves spike rate inference from two-photon calcium imaging data. *PLoS computational biology* 14: e1006157, 2018.

Borji A, Itti L. Defending yarbus: Eye movements reveal observers' task. *Journal of vision* 14: 29–29, 2014.

Bosman CA, Womelsdorf T, Desimone R, Fries P. A microsaccadic rhythm modulates gamma-band synchronization and behavior. *Journal of Neuroscience* 29: 9471–9480, 2009.

Buonocore A, Skinner J, Hafed ZM. Eye-position error influence over" open-loop" smooth pursuit initiation. *bioRxiv* p. 404491, 2018.

Burr DC, Morrone MC, Ross J. Selective suppression of the magnocellular visual pathway during saccadic eye movements. *Nature* 371: 511, 1994.

Carpenter RH *Movements of the Eyes, 2nd Rev* Pion Limited, 1988.

Chen CY, Hafed ZM. Postmicrosaccadic enhancement of slow eye movements. *Journal of Neuroscience* 33: 5375–5386, 2013.

Chen CY, Hafed ZM. A neural locus for spatial-frequency specific saccadic suppression in visual-motor neurons of the primate superior colliculus. *Journal of neurophysiology* 117: 1657–1673, 2017.

Christopher MB *PATTERN RECOGNITION AND MACHINE LEARNING.* Springer-Verlag New York, 2016.

Cohen J. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20: 37–46, 1960.

Colby C, Goldberg M et al. The updating of the representation of visual space in parietal cortex by intended eye movements. *Science* 255: 90–92, 1992.

Crevecoeur F, Kording KP. Saccadic suppression as a perceptual consequence of efficient sensorimotor estimation. *eLife* 6, 2017.

Dai W, Selesnick I, Rizzo JR, Rucker J, Hudson T 2016 A parametric model for saccadic eye movement In: *Signal Processing in Medicine and Biology Symposium (SPMB), 2016 IEEE*, p. 1–6. IEEE.

Daye PM, Optican LM. Saccade detection using a particle filter. *Journal of neuroscience methods* 235: 157–168, 2014.

Duchowski AT. Eye tracking methodology. *Theory and practice* 328, 2007.

Engbert R, Mergenthaler K. Microsaccades are triggered by low retinal image slip. *Proceedings of the National Academy of Sciences* 103: 7192–7197, 2006.

Fuchs AF, Robinson DA. A method for measuring horizontal and vertical eye movement chronically in the monkey. *Journal of applied physiology* 21: 1068–1070, 1966.

Gal Y, Ghahramani Z. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158* , 2015.

Gatys LA, Ecker AS, Bethge M. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* , 2015.

Golan T, Davidesco I, Meshulam M, Groppe DM, Mégevand P, Yeagle EM, Goldfinger MS, Harel M, Melloni L, Schroeder CE et al. Increasing suppression of saccade-related transients along the human visual hierarchy. *eLife* 6, 2017.

Gur M, Beylin A, Snodderly DM. Response variability of neurons in primary visual cortex (v1) of alert monkeys. *Journal of Neuroscience* 17: 2914–2920, 1997.

Hafed ZM. Mechanisms for generating and compensating for the smallest possible saccades. *European Journal of Neuroscience* 33: 2101–2113, 2011.

Hafed ZM. Alteration of visual perception prior to microsaccades. *Neuron* 77: 775–786, 2013.

Hafed ZM, Chen CY, Tian X. Vision, perception, and attention through the lens of microsaccades: mechanisms and implications. *Frontiers in systems neuroscience* 9: 167, 2015.

Hafed ZM, Goffart L, Krauzlis RJ. Superior colliculus inactivation causes stable offsets in eye position during tracking. *Journal of Neuroscience* 28: 8124–8137, 2008.

Hafed ZM, Krauzlis RJ. Goal representations dominate superior colliculus activity during extrafoveal tracking. *Journal of Neuroscience* 28: 9426–9439, 2008.

Haji-Abolhassani A, Clark JJ. An inverse yarbus process: Predicting observersâĂŹ task from eye movement patterns. *Vision research* 103: 127–142, 2014.

Hass CA, Horwitz GD. Effects of microsaccades on contrast detection and v1 responses in macaques. *Journal of vision* 11: 3–3, 2011.

Herrington TM, Masse NY, Hachmeh KJ, Smith JE, Assad JA, Cook EP. The effect of microsaccades on the correlation between neural activity and behavior in middle temporal, ventral intraparietal, and lateral intraparietal areas. *Journal of Neuroscience* 29: 5793–5805, 2009.

Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* , 2015.

Judge SJ, Richmond BJ, Chu FC. Implantation of magnetic search coils for measurement of eye position: an improved method. *Vision research* , 1980.

Kawaguchi K, Clery S, Pourriahi P, Seillier L, Haefner RM, Nienborg H. Differentiating between models of perceptual decision making using pupil size inferred confidence. *Journal of Neuroscience* 38: 8874–8888, 2018.

Kingma DP, Ba J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* , 2014.

Klibisz A, Rose D, Eicholtz M, Blundon J, Zakharenko S, Fast, simple calcium imaging segmentation with fully convolutional networks In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2017 p. 285–293.

Kowler E. Eye movements: The past 25 years. *Vision research* 51: 1457–1483, 2011.

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

Leigh RJ, Kennard C. Using saccades as a research tool in the clinical neurosciences. *Brain* 127: 460–477, 2004.

Leigh RJ, Zee DS *The neurology of eye movements*, Vol. 90 Oxford University Press, USA, 2015.

Leopold DA, Logothetis NK. Microsaccades differentially modulate neural activity in the striate and extrastriate visual cortex. *Experimental Brain Research* 123: 341–345, 1998.

MacAskill MR, Anderson TJ. Eye movements in neurodegenerative diseases. *Current opinion in neurology* 29: 61–68, 2016.

Mathis A, Mamidanna P, Cury KM, Abe T, Murthy VN, Mathis MW, Bethge M 2018 Deeplabcut: markerless pose estimation of user-defined body parts with deep learning Technical report, Nature Publishing Group.

Mihali A, van Opheusden B, Ma WJ. Bayesian microsaccade detection. *Journal of vision* 17: 13–13, 2017.

Otero-Millan J, Castro JLA, Macknik SL, Martinez-Conde S. Unsupervised clustering method to detect microsaccades. *Journal of vision* 14: 18–18, 2014.

Pekkanen J, Lappi O. A new and general approach to signal denoising and eye movement classification based on segmented linear regression. *Scientific reports* 7: 17726, 2017.

Reppas JB, Usrey WM, Reid RC. Saccadic eye movements modulate visual responses in the lateral geniculate nucleus. *Neuron* 35: 961–974, 2002.

Ronneberger O, Fischer P, Brox T 2015 U-net: Convolutional networks for biomedical image segmentation In: *Internatitonal Conference on Medical image computing and computer-assisted intervention*, p. 234–241. Springer.

Ross J, Morrone MC, Burr DC. Compression of visual space before saccades. *Nature* 386: 598, 1997.

Sheynikhovich D, Bécu M, Wu C, Arleo A. Unsupervised detection of microsaccades in a high-noise regime. *Journal of vision* 18: 19–19, 2018.

Sommer MA, Wurtz RH. Brain circuits for the internal monitoring of movements. *Annu. Rev. Neurosci.* 31: 317–338, 2008.

Startsev M, Agtzidis I, Dorr M. 1d cnn with blstm for automated classification of fixations, saccades, and smooth pursuits. *Behavior Research Methods* p. 1–17, 2018.

Theis L, Berens P, Froudarakis E, Reimer J, Rosón MR, Baden T, Euler T, Tolias AS, Bethge M. Benchmarking spike rate inference in population calcium imaging. *Neuron* 90: 471–482, 2016.

Tian X, Yoshida M, Hafed ZM. A microsaccadic account of attentional capture and inhibition of return in posner cueing. *Frontiers in systems neuroscience* 10: 23, 2016.

Yao T, Treue S, Krishna BS. Saccade-synchronized rapid attention shifts in macaque visual cortical area mt. *Nature communications* 9: 958, 2018.

Yu G, Yang M, Yu P, Dorris MC. Time compression of visual perception around microsaccades. *Journal of neurophysiology* 118: 416–424, 2017.

Zemblys R, Niehorster DC, Holmqvist K. gazenet: End-to-end eye-movement event detection with deep neural networks. *Behavior research methods* p. 1–25, 2018.

Zemblys R, Niehorster DC, Komogortsev O, Holmqvist K. Using machine learning to detect events in eye-tracking data. *Behavior research methods* p. 1–22, 2017.

Zirnsak M, Steinmetz NA, Noudoost B, Xu KZ, Moore T. Visual space is compressed in prefrontal cortex before eye movements. *Nature* 507: 504, 2014.

Zuber B, Stark L, Cook G. Microsaccades and the velocity-amplitude relationship for saccadic eye movements. *Science* 150: 1459–1460, 1965.

Acknowledgements We thank Konstantin-Friedrich Willeke and Antimo Buonocore for providing help with labelling saccade data, Murat Ayhan for input on DNNs and Adam von Daranyi for setting up the web service. This work was funded by the German Ministry of Education and Research (FKZ 01GQ1601) and the German Research Foundation (EXC307, EXC2064, SFB 1233, BE5601/4-1).

Author contributions MB, JB, ZH and PB designed the project; MB and JB developed the algorithm; MB implemented the algorithm; JB simulated, acquired and labelled data; MB and JB analyzed the data; HN provided data; ZH and PB supervised the project; MB, JB, ZH and PB wrote the paper.

Competing Interests The authors declare that they have no competing financial interests.

Correspondence Correspondence and requests for materials should be addressed to Z.M.H. and P.B. (email: ziad.m.hafed@cin.uni-tuebingen.de, philipp.berens@uni-tuebingen.de).

Table 1

Dataset	1	2	3	4
Subjects	Humans	Monkeys	Monkeys	Humans
Eye tracker	Eyelink 1000	Search coil	Eyelink 1000	Eyelink 1000
Sampling frequency (Hz)	1000	1000	500	1000
Saccade type	Microsaccades and memory saccades	Saccades during smooth pursuit	Microsaccades	Microsaccades and saccades
Mean dur ± std (ms)	44.58 ± 15.42	37.51 ± 8.81	23.12 ± 6.52	31.66 ± 8.93
Median dur (ms)	42	36	22	31
Minimum dur (ms)	11	18	8	8
Maximum dur (ms)	169	97	54	110
Mean amp ± std (°)	0.69 ± 0.93	1.07 ± 0.70	0.22 ± 0.13	0.33 ± 0.28
Median amp (°)	0.43	0.96	0.20	0.24
Min amp (°)	0.02	0.04	0.010	0.008
Max amp (°)	11.34	7.03	1.27	2.66
Mean peak vel ± std (°/s)	102.46 ± 68.82	68.23 ± 42.98	208.41 ± 65.95	61.93 ± 35.18
Median peak vel (°/s)	81.91	56.59	198.86	52.96
Min peak vel (°/s)	17.81	11.49	85.28	15.32
Max peak vel (°/s)	547.72	450.44	560.28	423.18
Median instant. vel (°/s)	5.63	15.70	10.20	5.63

Table 2

Dataset	Algorithm	F1	Cohen's Kappa	Δ Onset (ms)	Δ Offset (ms)
#1	U'n'Eye	**0.96** ± 0.01	**0.89** ± 0.02	**2.66** ± 0.34	**4.11** ± 0.41
	EM	0.87 ± 0.03	0.66 ± 0.02	5.39 ± 0.49	11.28 ± 1.00
	OM	0.85 ± 0.03	0.68 ± 0.03	3.80 ± 0.48	11.50 ± 0.77
	S	0.95 ± 0.02	-	-	-
	PL	0.92 ± 0.02	0.68 ± 0.03	5.51 ± 0.88	8.53 ± 0.60
#2	U'n'Eye	**0.96** ± 0.01	**0.92** ± 0.01	**1.70** ± 0.29	**2.19** ± 0.37
	DO	0.84 ± 0.03	0.49 ± 0.03	9.99 ± 0.19	9.22 ± 0.35
#3	U'n'Eye	**0.94** ± 0.01	**0.82** ± 0.02	**2.23** ± 0.22	**3.99** ± 0.60
	EM	0.77 ± 0.04	0.58 ± 0.04	3.22 ± 0.53	6.87 ± 0.66
	OM	0.68 ± 0.07	0.55 ± 0.06	3.48 ± 0.55	4.90 ± 0.64
	S	0.70 ± 0.04	-	-	-
	PL	0.83 ± 0.02	0.64 ± 0.03	2.74 ± 0.39	6.94 ± 0.50

Table 3

	Cohen's kappa	F1	Δ Onset (ms)	Δ Offset (ms)
Coder 1 vs Coder 2	0.83 ± 0.02	**0.98** ± 0.01	3.72 ± 0.39	7.10 ± 0.34
Network 1 vs Coder 1	**0.89** ± 0.02	0.96 ± 0.01	**2.65** ± 0.34	**4.11** ± 0.41
Network 2 vs. Coder 2	**0.89** ± 0.01	0.96 ± 0.01	**2.00** ± 0.11	**4.81** ± 0.33
Network 2 vs. Coder 1	0.85 ± 0.01	0.96 ± 0.01	3.34 ± 0.34	5.58 ± 0.33
Network 1 vs. Coder 2	0.86 ± 0.01	0.96 ± 0.01	2.82 ± 0.32	6.57 ± 0.53

Table 4

Metric	Comparison	Test	t-value	p-value
Kappa to C1	N1 vs C2	paired t-test	18.38	$2.98 . 10^{-7}$
Kappa rel. to C2	N2 vs C1	paired t-test	10.88	$3.69 . 10^{-10}$
F1 rel. to C1	N1 vs C2	paired t-test	-3.52	$5.08 . 10^{-2}$
F1 rel. to C2	N2 vs C1	paired t-test	-3.7	$5.19 . 10^{-2}$
Onset distance rel. to C1	N1 vs C2	indep. t-test	-6.6	$2.98 . 10^{-5}$
Onset distance rel. to C2	N2 vs C1	indep. t-test	-13.6	$5.26 . 10^{-10}$
Offset distance rel. to C1	N1 vs C2	indep. t-test	-17.9	$5.28 . 10^{-12}$
Offset distance rel. to C2	N2 vs C1	indep. t-test	-15.3	$7.33 . 10^{-11}$

Table 5

Event		Coder MN	U’n’Eye	NSLR-HMM	LNS
Saccades	Img	0.91	0.89	-	0.81
	Dot	0.80	0.79	-	0.75
	Vid	0.88	0.89	-	0.81
	All	0.89	**0.88**	0.82	0.81
PSOs	Img	0.76	0.72	-	0.64
	Dot	0.59	0.59	-	0.53
	Vid	0.73	0.68	-	0.63
	All	0.73	**0.70**	0.53	0.64
Blinks	Img	0.92	0.84	-	-
	Dot	0.77	0.71	-	-
	Vid	0.82	0.84	-	-
	All	0.91	0.83	-	-

Figure legends

Figure 1. Examples of eye traces containing saccades for detection. (A) Microsaccades during fixation recorded with a video-based eye tracker. (B) Catch-up saccades during smooth tracking recorded with scleral search coils. (C) Microsaccades during fixation recorded with a video-based eye tracker. (D) Microsaccades during fixation recorded with the same video-based eye tracker as in A but for different sets of subjects. (E) Simulated saccades. In all panels, 2D plots on the left are the 2D representation of the eye trajectory over 1 second of recording, and to the right of them are the horizontal and vertical components of the corresponding traces presented as a function of time; in this case, an upward deflection in the shown traces corresponds to a rightward or upward eye movement for the horizontal and vertical components, respectively. Note that in B, we refer to the non-saccadic smooth change in eye position as "fixation" for simplicity, since the primary goal of our algorithm was to detect saccades, irrespective of whether they happened during fixation or embedded in smooth pursuit eye movements.

Figure 2. U'n'Eye. (A) Network architecture. The input matrix contains horizontal and vertical eye velocity. T is the number of input time points (see B), and K is the user-defined number of eye movement classes (e.g. "fixation" versus "saccade" in a binary classifier). The different network layers are described in the text. (B) The output probability of one time bin is influenced by 89 time samples before and after this time bin. For each layer of the network, the red color indicates the range of influence of the time bin indicated by the red dot in the output. Traces show the projection of the layer's output onto its first principal component. The outputs of convolutional layers 6 and 7 resemble the final classifier's output probability (Softmax), whereas early convolutional layers 1 and 2 seem to perform noise reduction.

Figure 3. Examples of eye traces from our first three datasets. Saccades are labelled by either human coders, different instances of U'n'Eye, or a popular algorithm from the literature included here for illustrative purposes (also see Fig. 4 for detailed performance comparisons to several algorithms). (A) An example microsaccade exhibiting substantial PSO. The top two traces

show eye position as a function of time in an identical format to Fig. 1. Below the eye position traces, we show labels for "fixation" or "saccade" made by two human experts (Human coder 1 and Human coder 2) as well as predictions of two separate networks. Network 1 was trained on labels from Human coder 1, and Network 2 was trained on labels from Human coder 2. Note how each network matched the performance of its corresponding human coder. The very bottom row shows the performance of the Engbert and Mergenthaler (Engbert and Mergenthaler, 2006) algorithm (referred to as EM in all remaining figures), which suffered from a false alarm later in the trace due to eye tracker noise. (B) Saccades embedded in smooth pursuit eye movements. Here, our network successfully detected three catch-up saccades, all of which were missed by the EM algorithm, which was not designed to work with eye movement records containing smooth pursuit. The reason that these saccades were missed is that the saccades were directed opposite to the ongoing pursuit, resulting in momentary reductions in eye speed, as opposed to increases. (C) An example microsaccade embedded in high eye tracker noise. Once again, the EM algorithm suffered from a false alarm due to eye tracker noise.

Figure 4. High performance of U'n'Eye. Each panel shows results from one performance metric described in the text, and on each of our first three datasets. For each metric, we show the median across 10 different cross-validation runs. The boxes show two quartiles of the distributions. (A) F1 score summarizing precision and recall performance between two predictors. The first predictor was always a human coder (considered as ground truth). Therefore, the first column indicates agreement between a second coder (labelled Human in the figure) to the original coder used to train our network. (B) Cohen's kappa measuring sample to sample agreement. (C) Average absolute difference in the timing of saccade onset times. (D) Same as C but for saccade offset times. In all cases, our network (highlighted by gray rectangles) demonstrated superior performance (the arrows on the far right side indicate the direction of superior performance for each metric). EM: Engbert & Mergenthaler (Engbert and Mergenthaler, 2006), OM: Otero-Millan et al. (Otero-Millan et al., 2014), S: Sheynikhovich et al. (Sheynikhovich et al., 2018), PL: Pekkanen & Lappi (Pekkanen and Lappi, 2017), DO: Daye & Optican (Daye and Optican, 2014). Note that we only tested dataset 2 on DO because only this algorithm was explicitly designed to deal with

smooth pursuit eye movements.

Figure 5. Location on the main sequence of detected and undetected saccades. Left panels show saccades that were detected both by a human expert and U'n'Eye. The detected saccades expectedly followed the main sequence relationship between peak velocity and movement amplitude. Middle panels show saccades that were detected only by U'n'Eye. Most saccades were small and close to the eye tracker noise, likely being cautiously unlabelled by human coders. In the inset, a large saccade was detected by U'n'Eye but not by the human coder, suggesting a possible lapse by the latter. Right panels show saccades missed by U'n'Eye. Most of these were very small.

Figure 6. Multiclass labelling by U'n'Eye. (A) An example saccade showing substantial post-saccadic oscillation (PSO) from the data in (Larsson et al., 2013). (B) An example full trace from the same dataset showing sequences of saccades, PSO's, and blinks. (C) For the trace in B, ground truth labels are shown, in addition to labels by U'n'Eye. The latter successfully classified all ground truth labels, except for one instance marked by a black vertical arrow. (D) Nonetheless, the predictive probabilility of the network still showed a transient for the missed microsaccade (black arrow), suggesting that additional post-processing may be used to improve the performance of U'n'Eye even more. For example, the user could manually inspect significant transients in predictive probability.

Figure 7. Robustness of U'n'Eye performance under a variety of training regimes. (A) Upper panel: U'n'Eye saccade detection performance (red) as a function of amount of training samples. Lower panel: linear increase of training time with the number of training samples on a CPU. Data shows mean +/- standard deviation across the same training epochs as in the upper panel. The colored horizontal lines with labels refer to the performance of other algorithms from the literature that we tested. (B) U'n'Eye saccade detection performance as a function of the fraction of saccade labels missing in 300 seconds of training data. The colored horizontal lines refer to the same algorithms as in A. Also, the red line in A and B shows mean +/- standard deviation across networks trained on three different subsets of dataset 1. Performance of U'n'Eye and other

algorithms was evaluated on 1000 seconds of test data from dataset 1. (C) U'n'Eye saccade detection performance in simulated data with missing labels for different amounts of training samples N in seconds. (D) U'n'Eye saccade detection performance for different combinations of training and test sets. Each number in a square (and its associated color code) indicates the F1 score for training on one dataset and testing on another. The column labelled 1+2+3 on the far right shows results when the network was trained on all three datasets simultaneously (but ensuring the same amount of training data as in the other columns of the figure). (E) U'n'Eye saccade detection performance for combinations of different human subjects in training and test data from dataset 4. Each column shows the results of training on a single subject from the dataset, in a similar format to D. The final column on the right indicates that training the network on a (small) population of subjects yields best performance, and the rest of the figure indicates that additional subjects can then be tested with the pre-trained network without much loss in performance. The same color scale was used for D and E.

Figure 8. Disentanglement of fixations and saccades throughout the network. (A) Example eye trace with a microsaccade. (B) Distribution of dataset 2 in the velocity domain. Fixations and saccades (shown in bluish and orangish colors, respectively) showed overlapping distributions. (C) Fraction of explained variance by the two first principal components (PCs) of the network's convolutional layers. There was a reduction in the middle layers followed by a peak at the final seventh layer. (D) Projection of hidden layer activations by eye traces of dataset 2 onto the first two principal components. Fixation and saccade classes became better separated throughout the hidden layers. (B - D) Dots indicate the time points of the example eye trace in A, and the rest of the background data show the entire dataset time samples. (E) The probability output allowed for a linear separation of the two classes. Time points with a saccade predictive probability above 0.5 were classified as a saccade.

Table legends

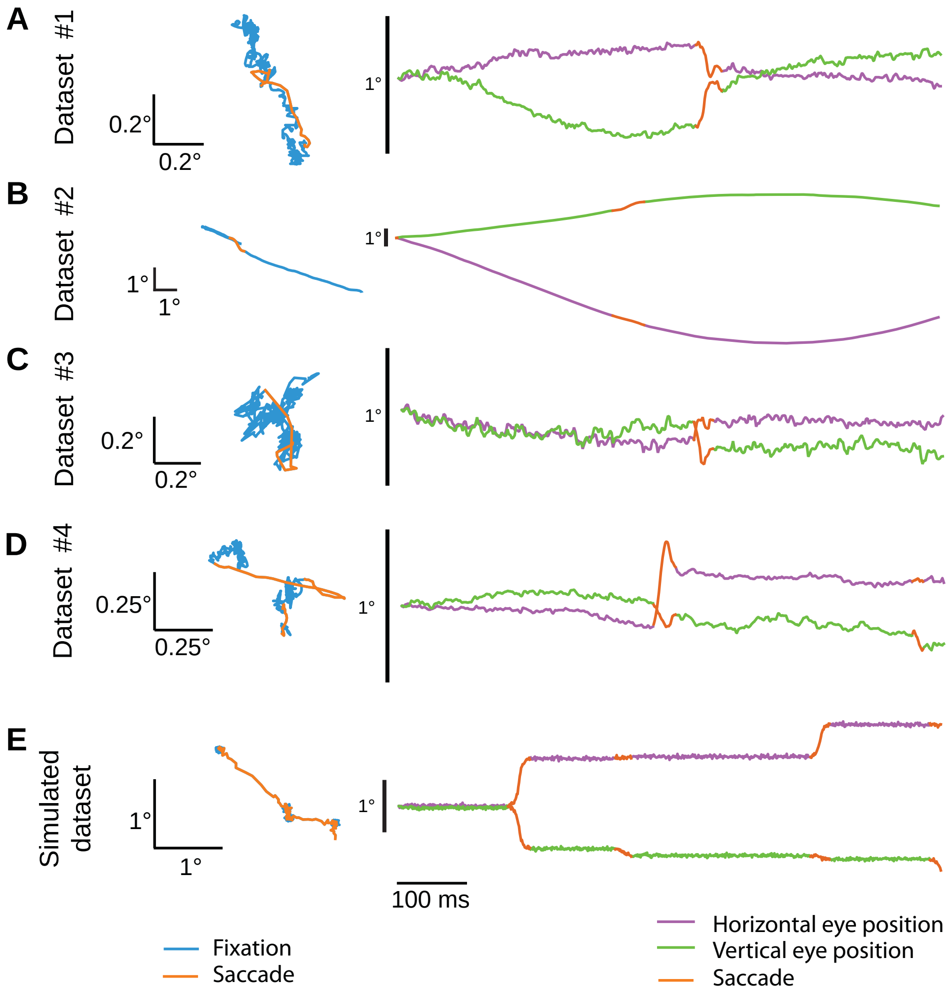
Table 1. Dataset characteristics. All statistics refer to saccades. Note that minimum saccade amplitude may appear very low due to the existence of some saccades that had very strong dynamic overshoot (a substantial saccadic movement followed by one lobe of a PSO almost to the original eye position before saccade onset). The statistics of the simulated dataset are described in Methods.

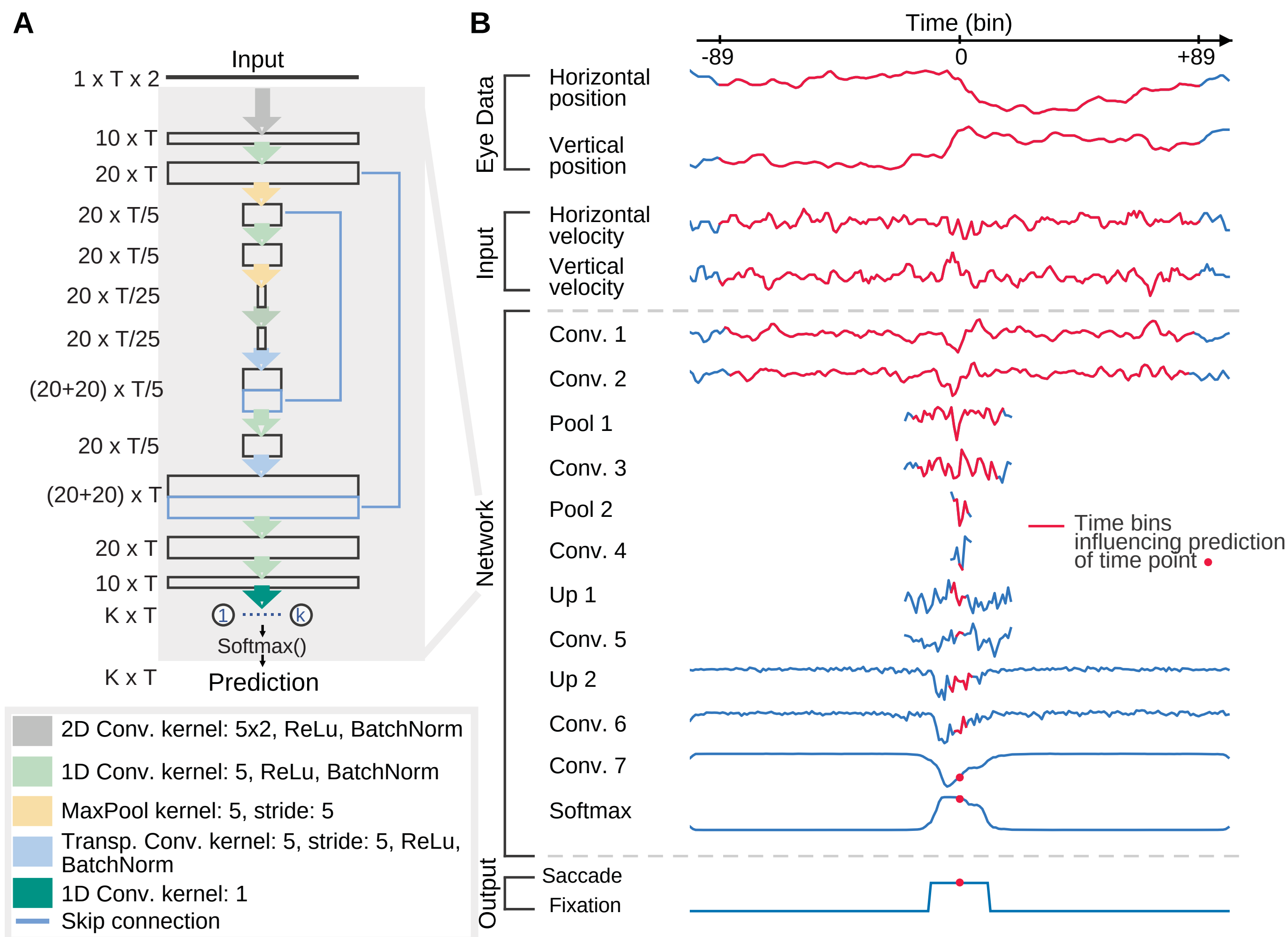
Table 2. Comparison of U'n'Eye performance to other algorithms on datasets 1, 2, and 3. In bold are the best performances for each dataset. In all cases, U'n'Eye achieved highest performance. Values report mean and standard deviation across validation sets.

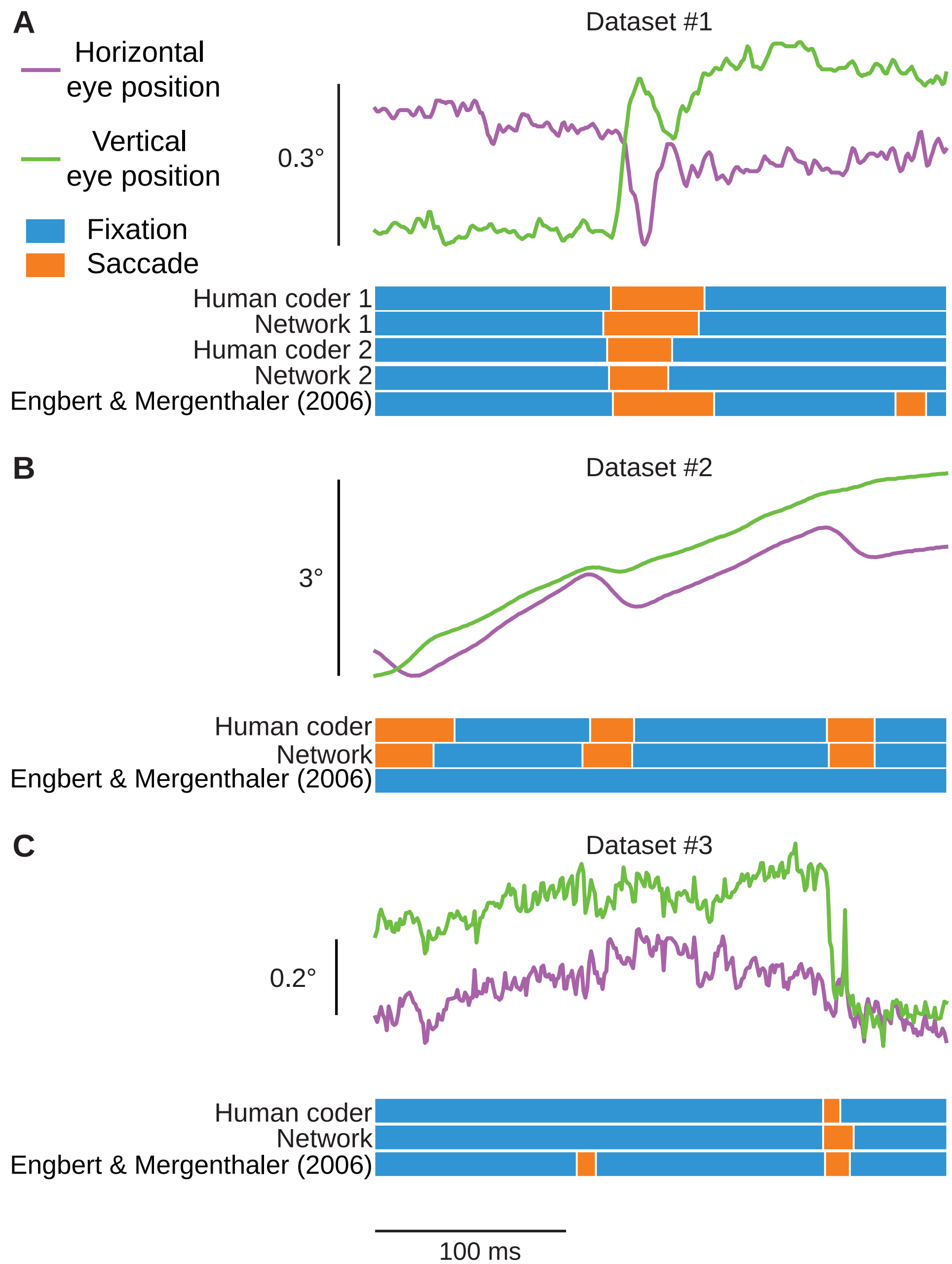
Table 3. Inter-rater comparison. The first row shows the similarity measures between labels from two human experts (Coder 1 and Codre 2). Network 1 was trained on labels from Coder 1, and Network 2 was trained on labels from Coder 2. In bold are comparisons leading to best performances. Values report mean and standard deviation across cross-validations. Inter-coder agreement was evaluated on the 10 test samples from cross-validation.

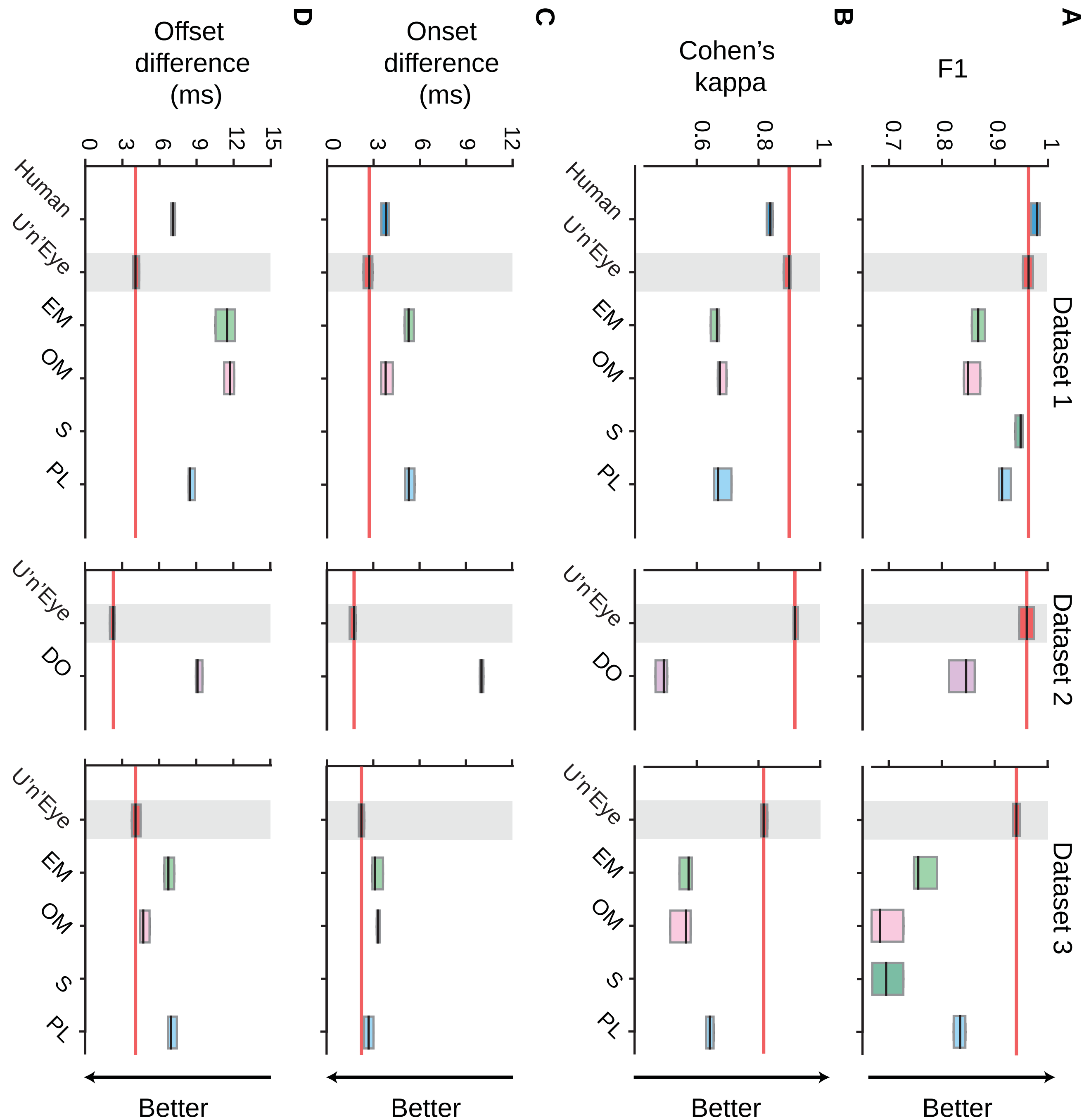
Table 4. Statistical tests in inter-rater comparison. Network 1 (N1) was trained on labels from Coder 1 (C1), and Network 2 (N2) was trained on labels from Coder 2 (C2). All p-values were Bonferroni corrected for multiple comparisons.

Table 5. Performance of U'n'Eye compared to state-of-the-art algorithms. NSLR-HMM and LNS values were taken from the respective publication (NSLR-HMM (Pekkanen and Lappi, 2017), LNS (Andersson et al., 2017)). For U'n'Eye, values are the median across 20 independent networks. In bold are the values reached by the best performing algorithm.

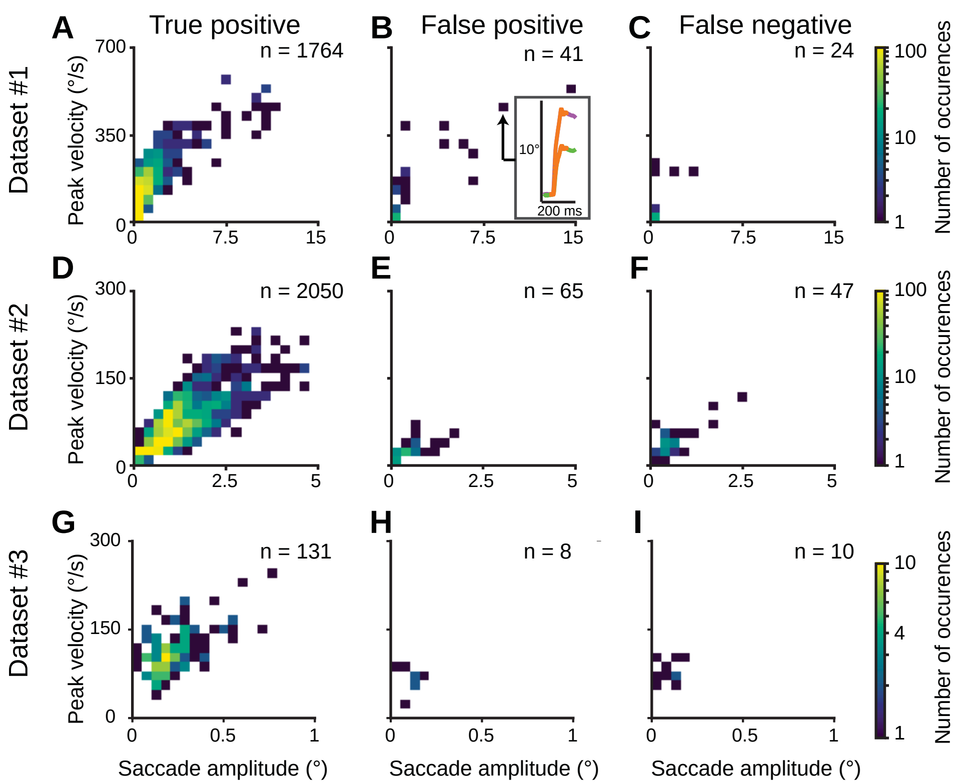

A
Dataset #1
0.2°
0.2°
1°
B
Dataset #2
1°
1°
1°
C
Dataset #3
0.2°
0.2°
1°
D
Dataset #4
0.25°
0.25°
1°
E
Simulated dataset
1°
1°
1°
100 ms
Fixation
Saccade
Horizontal eye position
Vertical eye position
Saccade


A
Input
1 x T x 2
10 x T
20 x T
20 x T/5
20 x T/5
20 x T/25
20 x T/25
(20+20) x T/5
20 x T/5
(20+20) x T
20 x T
10 x T
K x T
1
k
Softmax()
K x T
Prediction
2D Conv. kernel: 5x2, ReLu, BatchNorm
1D Conv. kernel: 5, ReLu, BatchNorm
MaxPool kernel: 5, stride: 5
Transp. Conv. kernel: 5, stride: 5, ReLu, BatchNorm
1D Conv. kernel: 1
Skip connection
B
Time (bin)
-89
0
+89
Eye Data
Horizontal position
Vertical position
Input
Horizontal velocity
Vertical velocity
Network
Conv. 1
Conv. 2
Pool 1
Conv. 3
Pool 2
Conv. 4
Up 1
Conv. 5
Up 2
Conv. 6
Conv. 7
Softmax
Time bins influencing prediction of time point
Output
Saccade
Fixation

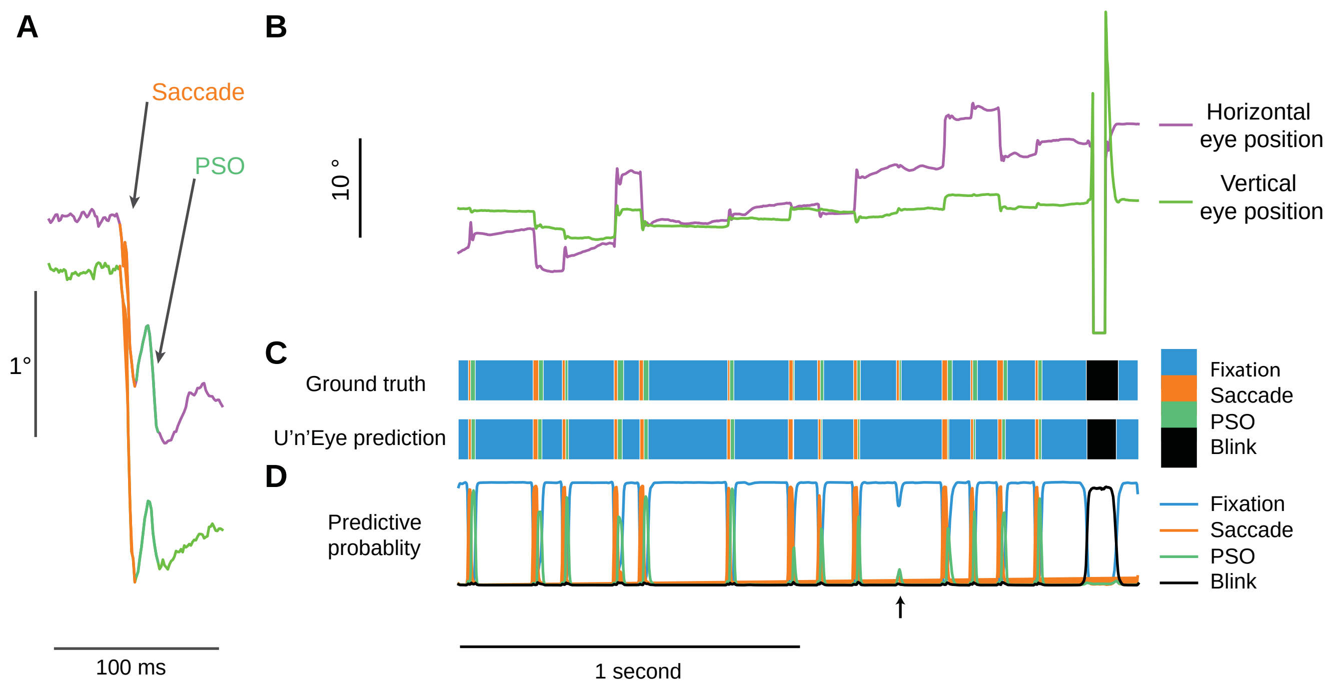

A
Horizontal eye position
Vertical eye position
Fixation
Saccade
Dataset #1
0.3°
Human coder 1
Network 1
Human coder 2
Network 2
Engbert & Mergenthaler (2006)
B
Dataset #2
3°
Human coder
Network
Engbert & Mergenthaler (2006)
C
Dataset #3
0.2°
Human coder
Network
Engbert & Mergenthaler (2006)
100 ms

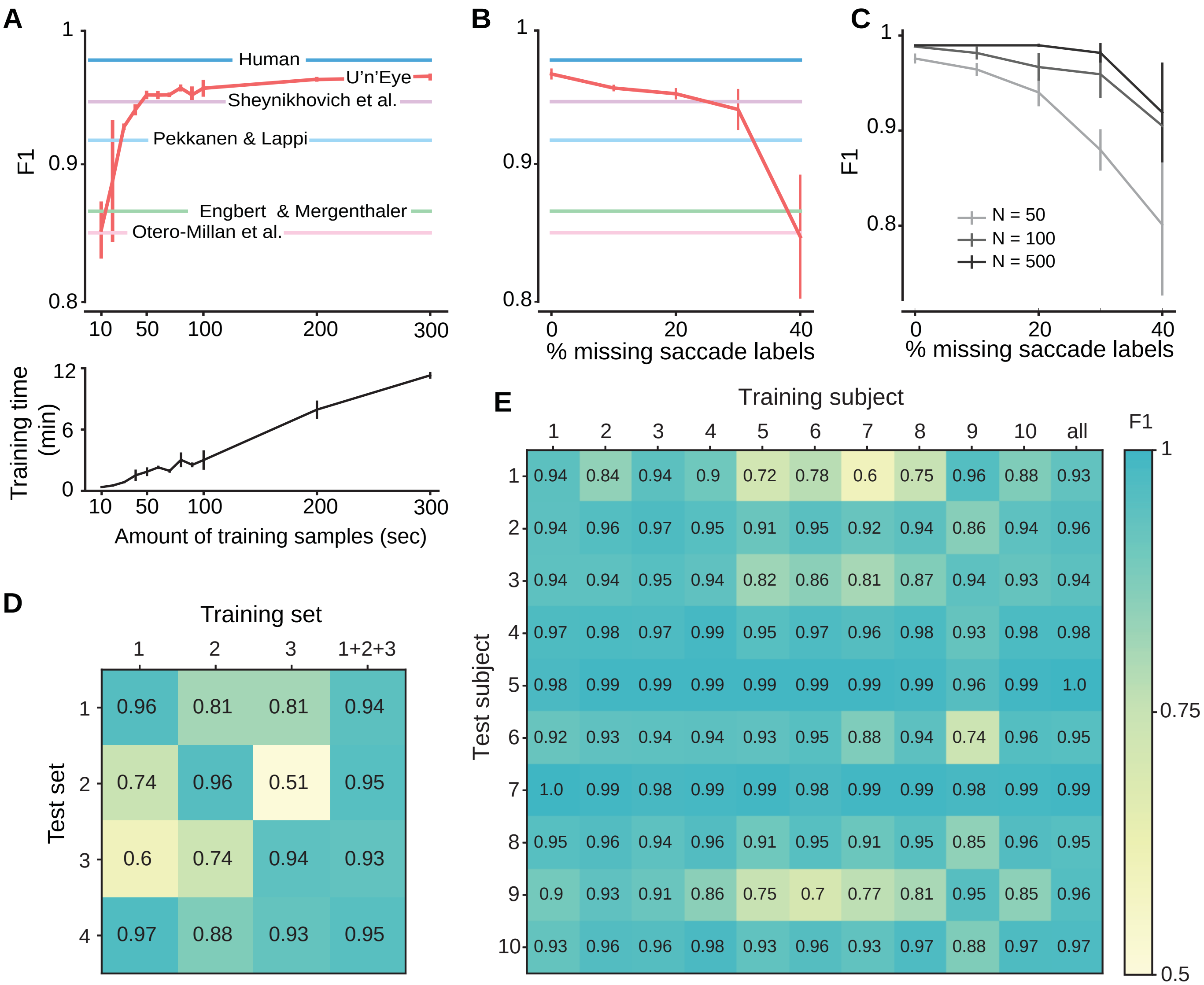


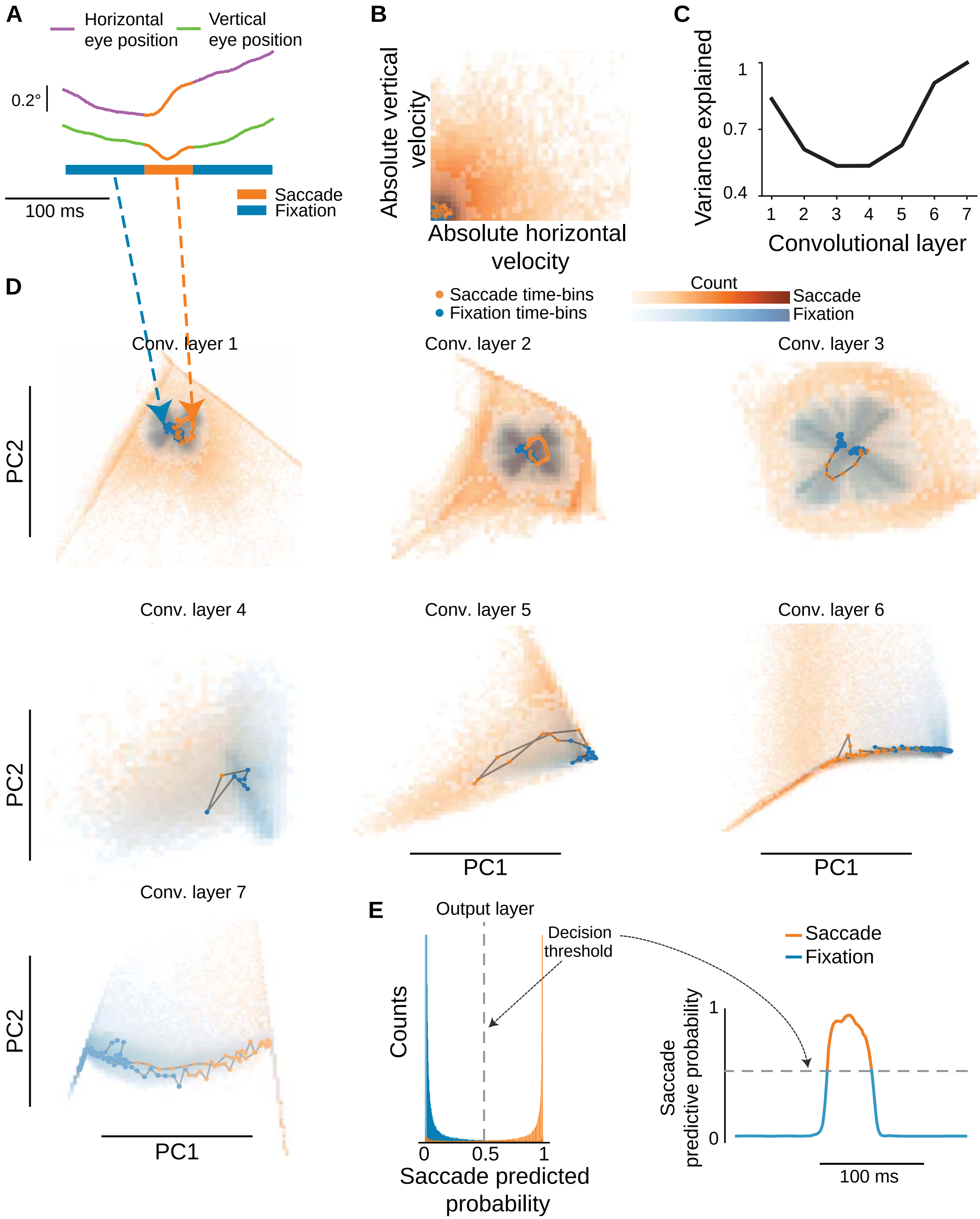
A
B
C
D
F1
Cohen's kappa
Onset difference (ms)
Offset difference (ms)
Dataset 1
Dataset 2
Dataset 3
Human
U'n'Eye
EM
OM
S
PL
DO
Better
0.7
0.8
0.9
1
0.6
0.8
1
0
3
6
9
12
0
3
6
9
12
15



A
True positive
B
False positive
C
False negative
Dataset #1
Peak velocity (°/s)
n = 1764
n = 41
n = 24
10°
200 ms
Number of occurences
D
E
F
Dataset #2
n = 2050
n = 65
n = 47
G
H
I
Dataset #3
n = 131
n = 8
n = 10
Saccade amplitude (°)


A
B
Saccade
PSO
1°
100 ms
10 °
Horizontal eye position
Vertical eye position
C
Ground truth
U'n'Eye prediction
Fixation
Saccade
PSO
Blink
D
Predictive probablity
Fixation
Saccade
PSO
Blink
1 second


A
F1
1
0.9
0.8
Human
U'n'Eye
Sheynikhovich et al.
Pekkanen & Lappi
Engbert & Mergenthaler
Otero-Millan et al.
10
50
100
200
300
Training time (min)
12
6
0
Amount of training samples (sec)
B
% missing saccade labels
0
20
40
C
N = 50
N = 100
N = 500
D
Training set
1
2
3
1+2+3
Test set
1 0.96 0.81 0.81 0.94
2 0.74 0.96 0.51 0.95
3 0.6 0.74 0.94 0.93
4 0.97 0.88 0.93 0.95
E
Training subject
1 2 3 4 5 6 7 8 9 10 all
Test subject
1 0.94 0.84 0.94 0.9 0.72 0.78 0.6 0.75 0.96 0.88 0.93
2 0.94 0.96 0.97 0.95 0.91 0.95 0.92 0.94 0.86 0.94 0.96
3 0.94 0.94 0.95 0.94 0.82 0.86 0.81 0.87 0.94 0.93 0.94
4 0.97 0.98 0.97 0.99 0.95 0.97 0.96 0.98 0.93 0.98 0.98
5 0.98 0.99 0.99 0.99 0.99 0.99 0.99 0.99 0.96 0.99 1.0
6 0.92 0.93 0.94 0.94 0.93 0.95 0.88 0.94 0.74 0.96 0.95
7 1.0 0.99 0.98 0.99 0.99 0.98 0.99 0.99 0.98 0.99 0.99
8 0.95 0.96 0.94 0.96 0.91 0.95 0.91 0.95 0.85 0.96 0.95
9 0.9 0.93 0.91 0.86 0.75 0.7 0.77 0.81 0.95 0.85 0.96
10 0.93 0.96 0.96 0.98 0.93 0.96 0.93 0.97 0.88 0.97 0.97
F1
1
0.75
0.5


A
Horizontal eye position
Vertical eye position
0.2°
100 ms
Saccade
Fixation
B
Absolute vertical velocity
Absolute horizontal velocity
C
Variance explained
1
0.7
0.4
1 2 3 4 5 6 7
Convolutional layer
D
Saccade time-bins
Fixation time-bins
Count
Saccade
Fixation
Conv. layer 1
Conv. layer 2
Conv. layer 3
Conv. layer 4
Conv. layer 5
Conv. layer 6
Conv. layer 7
PC2
PC1
E
Output layer
Decision threshold
Counts
0 0.5 1
Saccade predicted probability
Saccade
Fixation
Saccade predictive probability
1
0
100 ms