Fully Convolutional Neural Networks for Raw Eye Tracking Data Segmentation, Generation, and Reconstruction

Wolfgang Fuhl
Human-Computer Interaction
Eberhard Karls University Tübingen
Germany, Tübingen, Sand 13
wolfgang.fuhl@uni-tuebingen.de

Yao Rong
Human-Computer Interaction
Eberhard Karls University Tübingen
Germany, Tübingen, Sand 13
yao.rong@uni-tuebingen.de

Enkelejda Kasneci
Human-Computer Interaction
Eberhard Karls University Tübingen
Germany, Tübingen, Sand 13
enkelejda.kasneci@uni-tuebingen.de

Abstract—In this paper, we use fully convolutional neural networks for the semantic segmentation of eye tracking data. We also use these networks for reconstruction, and in conjunction with a variational auto-encoder to generate eye movement data. The first improvement of our approach is that no input window is necessary, due to the use of fully convolutional networks and therefore any input size can be processed directly. The second improvement is that the used and generated data is raw eye tracking data (position X, Y and time) without preprocessing. This is achieved by pre-initializing the filters in the first layer and by building the input tensor along the z axis. We evaluated our approach on three publicly available datasets and compare the results to the state of the art.

I. INTRODUCTION

Eye movements [1]–[4] are the basis to get more information about a person [5], [6]. Most research papers investigate intentions, cognitive states [7], [8], workload [9] and attention [10], [11] of a person. The eye movements are used to generate more complex features for machine learning [12], [13] to classify or regress the desired information [14], [15]. This knowledge about a person is important in multiple fields, like automated driving [16] and for measuring the work load of a surgeon [5], [6]. For the eye movements themselves, there are also application areas like the recognition of eye diseases [17] and the foveated rendering [18]. The fields of eye tracking applications are becoming more and more diverse, but even today there are still a multitude of challenges. One of these challenges is the reliable classification of eye movements based on raw data. The commonly used algorithms require the determination of a large number of thresholds [19]. But most algorithms are bound to certain sampling rates of the eye tracker and do not work even if the signal is very noisy [20]–[26]. Newer approaches avoid these limitations by using machine learning methods [27]–[30]. This allows the algorithm to be re-trained for any eye tracker. The preprocessing of the data is still used by these methods; however, it brings restrictions regarding data in which the preprocessing does not work as intended. Another problem of machine learning is the necessity of annotated data. For this purpose simulators have already been presented [31], [32] that address this challenge.

In this paper, we present an approach that is not bound to a window size. We achieve this by the exclusive use of convolution layers that are spatial invariant and not bounded to an input size. Compared to other machine learning approaches, our approach uses raw data as input, eliminating preprocessing. This has the advantage that our approach works autonomously and does not depend on the effectiveness of other methods. Furthermore, we show that our approach can be used for the classification, generation and reconstruction of eye tracking data.

Contribution of this work:

1. Processing of raw eye tracking data directly.
2. Window free approach (fully convolutional).
3. Eye tracking data reconstruction.
4. Variational autoencoders for data generation.

II. RELATED WORK

Since this work proposes an approach for the classification of eye movements as well as the reconstruction and generation, we have divided this section into two subsections.

A. Eye Movement Classification

The two most famous and most common algorithms in the field of eye movement classification are Identification by Dispersion Threshold (IDT) [33] and Identification by Velocity Threshold (IVT) [33]. In the former, the data is first reduced [34]. Then, two thresholds are used to distinguish between fixations and saccades. The first threshold limits the dispersion of the measurement points and the second threshold limits the minimum duration of fixations. In the second algorithm, however, only one threshold is used, which limits the eye movement velocity. If an eye movement is above this threshold, it is classified as Saccade, otherwise, a fixation is assumed. This second algorithm (IVT) has already been extended by adaptive methods to determine the threshold [35]. A further improvement in the signal noise level adaption was achieved by using the Kalmann filter (IKF) [36]. Here, the Kalmann filter is used to predict the next value, resulting in the signal being smoothed online. In addition to

the velocity threshold, a threshold is used for the minimum fixation duration. A similar algorithm has been published in [37]. The difference to IKF is the use of the χ^2-test instead of the Kalmann filter. Not only has the IVT algorithm been extended, but also the IDT algorithm. The first extension is the F-tests Dispersion Algorithm (FDT) [38]. The F-test provides the probability whether several data points belong to the same class. Since the F-test always expects a normal distribution, it is relatively susceptible to noise in the data. In the Covariance Dispersion Algorithm (CDT) [39], the F-test was replaced by a covariance matrix. For classification, the CDT requires three thresholds. The first two thresholds are for the variance and the covariance and thus represent an improvement of the dispersion threshold. The third threshold is for the minimum fixation duration. The last approach that followed the idea of IDT is the Identification by a Minimal Spanning Tree (IMST) [37] algorithm. Here a tree structure is calculated on the data, where each data point represents a leaf of the tree. Clusters are formed over the number of branches, which represent fixations and can be seen as a form of dispersion.

The first approach with machine learning was made in for the adaptive thresholds. Hidden Markov Models (HMM) were used to determine the class based on the velocity and the current state of the model [37]. Most models have two states (fixation and saccade) to classify the data. This approach of HMMs was also extended with smooth pursuits [40] and an additional state. In addition to the smooth pursuits, the post saccadic movements (PSM) also became interesting for science. The first algorithm dealing with the detection of PSM was presented in [41]. One year later the Binocular-Individual Threshold (BIT) [42] algorithm was introduced, which uses both eyes to detect PSM. This algorithm also used adaptive thresholds and follows the idea that both eyes perform the same movement. The first algorithm able to detect four eye movements was presented in [43]. This algorithm uses different data cleansing techniques and adaptive thresholds. For eye tracking data with a very high sampling rate an algorithm was presented in [44]. This algorithm is able to detect four eye movement types and is based on several steps in which all data is processed. The first step generates a rough segmentation and, in the following steps, this segmentation is further refined. Meaning, that the algorithm cannot be used online.

Novel approches for eye movement classification use modern machine learning approaches. The first approach to be mentioned here is [45]. This approach uses a conventional neural net with convolution layers and a fixed window size. The data in each window is first transferred to the frequency domain via the fast Fourier transformation and then used as input for the neural net, which classifies the eye movement type. Another approach is described in [46]. Here, a random forest is used to be applicable to bending eye tracker sampling rates. For this the input data is interpolated via cubic splines and 14 different features, like the eye movement speed, are calculated. These 14 features serve as input data for the

random forest and must always be calculated in advance. In addition, postprocessing is performed with Gaussian smoothing of the class probabilities as well as a heuristic for the final classification. A rule based learning algorithm was presented in [2]. Different data streams like the eye movement speed can be provided to the algorithm, whereupon the algorithm learns rule sets consisting of thresholds. Based on these rule sets, new data is classified to eye movement types. The last representative of the modern methods, is a feature which enters the velocities based on their direction into a histogram [1]. This histogram is normalized and can be used with any machine learning method.

B. Eye Tracking Data Reconstruction & Generation

The synthesis of eye movements is still a challenging task today. The Kalman filter makes the prediction of the next gaze point and is able to generate fixations, saccades and smooth pursuits. A disadvantage of this method is that there is no realistic noise in the signal which reflects the inaccuracy of the eye tracker. A rendering based approach was introduced in 2002 [47]. The main focus of this method was on the saccades but the method is also able to generate smooth pursuits and binocular rotations (vergence). A pure data based approach was presented in [48]. These methods simulate eye movements as well as head rotations. A disadvantage of the methods is that the head movements automatically trigger eye movements. Normally, head movements are only triggered when a target is more than $\approx 30°$ apart [49]. Another purely data-driven approach is described in [50]. It is an automated framework that simulates head, eye and eyelid movements. The method uses sound input to generate the movements, which are projected over several normal distributions onto eye, head and eyelid movements. Another approach focused on eye rotation is described in [51] and is based on the description of eye muscles by [52]. A disadvantage of this simulation is that eye movements cannot be generated automatically, but have to be predefined. The last rendering based approach is described in [53]. Here images and gaze vectors are randomly generated and the simulator is used to train machine learning techniques for detection and gaze vector regression. The methods mentioned so far originate from computer graphics and do not have the aim to generate realistic eye movement sequences. Their actual use lies in the interaction with humans [54]. This results in all movements being error free and absolutely accurate, which does not correspond to reality. Furthermore, the procedures described do not include evaluation of visual input or task specific behavior. The first approach to realistic simulation of eye tracking data for static images is described in [55]. This approach uses a random sequence of numbers in combination with statistical models and saliency maps to generate eye tracking data. An extension of this approach that added noise is described in [56]. In addition to noise, jitter based on a normal distribution was added [57]. A multi-layer calculation approach is described in [?], [32]. The simulator allows to generate a random sequence of eye movement sequences and to map them to static, dynamic, and eye tracking

data. This simulator can also generate any sampling rate as well, as it supports dynamic sampling rates.

Machine learning based approaches have already been presented. In [58] deep recurrent neural networks are used to generate eye movements based on static images. A disadvantage of this approach is that it only works on already seen images. An approach which uses Generative Adversiaral Networks (GANs) is described in [59]. This approach uses recurrent layer and a combination of static image and saliency maps to predict a scanpath.

III. METHOD

In this section, we describe our three approaches and how we trained them. Each task (semantic segmentation, reconstruction and generation) has its own subsection and is described in detail together with the training parameters. All networks were trained from scratch with a random initialization. While all of our models work with raw eye tracking data, it has to be mentioned that NaN or Inf values in the input files will corrupt the result. For the reconstruction model, those values have to be set to zero for example.
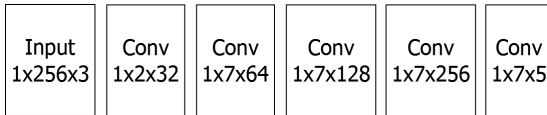
A. Semantic Segmentation

Fig. 1. The eye movement segmentation model used in our experiments.

Our eye movement segmentation network consists of five convolution layers with rectifier linear unit (ReLu) activation units. The input to our network is raw eye tracking data (position x, y and time). For our model, the input data is arranged one after the other (see Figure 1). This results in an input tensor that has a fixed depth of 3, a width of 1, and an arbitrary height. In Figure 1, the height was set to 256. This arrangement has the advantage that the weight tensor of the convolution extends over the whole depth and is only shifted along the height. Therefore, a convolution always sees all three input types (position x, y and time). In our training we used a fixed constant of one hundred as divisor for the input values to gain numerical stability. Without this divisor, it is also possible to train the network, but with lower learning rates, which prolong the whole training.

The first convolution layer has a height of two (see Figure 1). For this layer, it is important to check that, for each superimposed weight (along the height), one is positive and one is negative. Meaning, that after random value initialization, if two superimposed weights in the first layer are both positive, one is set to its negative value and vice versa. This has only to be done for the first layer. All the other layers are randomly initialized without any modification.

The last layer of our model has five output layers. This is due to the use of the softmax loss function and these five layers hold the output probability distributions for the corresponding

eye movement types (Fixation, Saccade, Smooth pursuit, PSM, error) and can be extended. In addition, it can be seen that our network does not use any down or upscaling operation.

1) Semantic Segmentation training parameters: For training on both datasets we used an initial learning rate of 10^{-2} together with the stochastic gradient decent (SGD) [60] optimizer. The parameters for the optimizer are $weightdecay = 10^{-4}$ and $momentum = 0.9$. For the loss function we used the weighted log multi class loss together with the softmax function. After each five hundred epochs, the learning rate was reduced by 10^{-1} until it reached 10^{-6} when the training was stopped. For data augmentation, we used random jitter that changes the value of a position to up to 2% around its original value. In addition, we shifted the entire input scanpath by a randomly selected value (the same value for all entries). We also used different input sizes where it has to be noted that in one batch, all length where equal because, otherwise, computational problems arise due to the not aligned data.
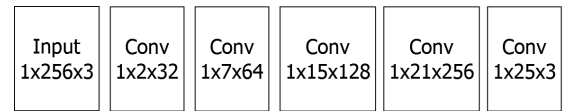
B. Reconstruction

Fig. 2. The model of our eye tracking raw data reconstruction network.

The model we use to reconstruct eye tracking data has the same structure as the segmentation model (Figure 2). The only difference is the output, which corresponds to the eye tracking signal itself. At the beginning, there is the sign-based pre-initialized convolution with the height two. Then follow the convolution layers, where the size of the convolution always doubles after the layer with a convolution height of seven. The last convolution layer reconstructs the signal and has an output depth of three (X, Y, time) and a height of twenty-five.

At this point it must be said that the output as well as the input layer can be extended. An example of this would be three dimensional coordinates, which can be processed and trained with an input and output layer of depth four. Furthermore, as with the segmentation mesh, no input window is required, allowing the mesh to be applied to any input length. It is also possible to train and validate the net with different and varying input lengths.

1) Reconstruction training parameters: For training on both datasets, we used an initial learning rate of 10^{-4} and changed it after ten epochs to 10^{-3}. This was done to avoid numerical problems for the random initialized models, which end up in not a number results (NaN). As optimizer, we used adam [61] with the parameters $weightdecay = 5 * 10^{-4}$, $momentum1 = 0.9$, and $momentum2 = 0.999$. As loss function, we used the L2 loss for the first hundred epochs. Afterwards, we used the L1 loss function to improve the accuracy of the network. The learning rate was decreased by 10^{-1} after each five hundred epochs and the training was stopped at a learning rate of 10^{-6}. For data augmentation,

144

we used random jitter that changes the value of a position to up to 2% from its original value. In addition, we shifted the entire input scanpath by a randomly selected value (the same value for all entries). We also used different input sizes, where it has to be noted that in one batch, all length where equal because otherwise computational problems arise due to the not aligned data. In addition, it is important to note that for the training, only parts without error are selected since otherwise, our network would learn to reconstruct errors or what is most likely, is that it would learn nothing.
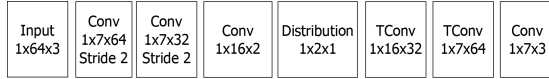
C. Generation

Fig. 3. The Variational Autoencoder (VAE) used for eye tracking data generation.

Figure 3 shows the structure of the variational autoencoder [62] (VAE) used. In comparison to the reconstruction as well as the segmentation net, we did not use the first pre-initialized convolution layer. This is due to the position itself is not used as input nor as output. We used the position change in x and y together with the time difference as input and as target for learning. This was done to make the output dependent to the last position of the scanpath and avoids jumping around the image since a new scanpath is generated based on a number of random values from normal distributions. Therefore, we need to specify randomly an initial start position from where the scanpath is further constructed. The input layer is is followed by two convolution layers, which also reduce the input by half. This was realized by average pooling. The last output layer of the encoder has a depth of two and corresponds to the mean value and the variance of the normal distributions. Then, a layer with depth one follows, which corresponds to Z, the value of the learned distribution. The decoder part of the network then learns to generate new data based on the distributions. For completeness, a brief description of the VAE is given below.

1) Description Variational Autoencoder (VAE): A VAE is similar to an normal autoencoder and consists of an encoding and a decoding part. The main difference is that, instead of encoding an input as a single point, the input is encoded as a distribution. Therefore, the encoder learns to map the input to the parameters of the normal distribution (mean m and
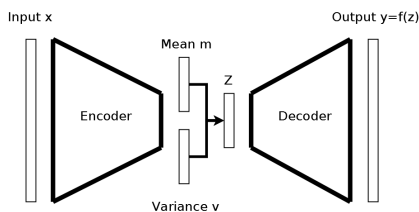
Fig. 4. The concept of an Variational Autoencoder (VAE).

variance v). The decoder in contrast learns to generate new samples based on the output of the normal distribution z.

Since the error cannot be propagated back through the distribution, the reparametrisation trick is used. The calculation of z ($z = N(m, v)$) is replaced with $z = m + v * N(0, 1)$. This calculation of the distribution is derivable and thus the error can be propagated back. Another difference between the VAE and the normal autoencoder is that the error depends not only on the difference between the input (x) and the output (y) but also on the similarity of the distributions. Therefore, the loss function is added another term, the Kullback-Leibler divergence. This divergence computes the distance between two distributions. The whole loss function is therefore computed as $||x - y|||^2 - KL(N(m, v), N(0, 1))$.

2) Generation training parameters: For training, we used an initial learning rate of 10^{-4} and changed it after one hundred epochs to 10^{-3}. As optimizer, we used stochastic gradient decent (SGD) [60]. The parameters for the optimizer are $weight decay = 10^{-6}$ and $momentum = 0.9$. As loss function we used the L2 loss in combination with the KL divergence as described in Section III-C1. The learning rate was decreased by 10^{-1} after each thousand epochs and the traing was stopped at a learning rate of 10^{-6}. We did not use any data augmentation technique since the reparametriztion trick already induces some deformation in the output data.

IV. EVALUATION

The evaluation section is split into three subsections. In each subsection we evaluate our approach for a specific task (semantic segmentation, reconstruction and generation) on multiple publicly available data sets. For evaluation we used the data sets from [40] (DS-SAN) and [20] (DS-AND) were we only used the annotations from MN for the semantic segmentation evaluation. The data set from DS-SAN consists of 24 recordings from six subjects. Each subject made four recordings with different challenges for eye movement detection. The data set contains fixations, saccades, and smooth pursuits. In addition in contains errors from the Dikablis Pro eye tracker, which has a sampling rate of 30 Hz. The subjects were recorded at a distance of 300 mm from the screen using a chin rest. The data set DS-AND contains annotations for fixations, saccades, smooth pursuits, and post saccadic movement (PSM). It was recorded using a SMI HiSpeed 1250 system with a chin and forehead rest. The data set consists of 34 binocular recordings from 17 different students at Lund University and each file has a sampling rate of 500 Hz. They used static and dynamic stimuli during recording.

A. Evaluation Semantic Segmentation

For the comparison of our algorithm to the state of the art we used the algorithms [40] (IBDT), [41] (EV), [63] (I2MC), [43] (LS), [2] (RULE), and [1] (HOV). All algorithms where configured for offline use since only three algorithms are configurable for online use (IBDT, HOV, & RULE). In addition, we used the data unmodified, which means that no errors were removed. For the evaluation itself, we only

TABLE I
DIFFERENT TRAINING CONFIGURATIONS OF THE MACHINE LEARNING
APPROACHES USED TOGETHER WITH THE HOV FEATURE [1].

Name	Configuration
knn5-20	k=5,10,15, or 20
tree1-5	Maximum splits 50
tree2-3	Predictor selection with curvature
tree3-5	Exact categorization
tree3,5	Split criterion deviance

TABLE II
RECALL FOR EACH EYE MOVEMENT TYPE <u>WITH ERRORS</u> IN THE INPUT
DATA. PSM STANDS FOR POST SACCADIC MOVEMENT.

Data	Alg.	Recall				
		Fixation	*Saccade*	*Pursuit*	*Noise*	*PSM*
DS-AND	EV	0.61	0.73	0	0.94	0.02
	IBDT	0.65	0.35	0.63	0	0
	LS	0.91	0.88	0.15	0.13	0
	I2MC	0.02	**0.95**	0	0	0
	RULE	0.79	0.85	0.69	0.67	0.78
	Proposed	**0.94**	0.93	**0.91**	**0.96**	**0.89**
DS-SAN	EV	0.18	0.25	0	1.0	-
	IBDT	0.97	0.28	0.84	0	-
	LS	0.95	0	0.06	0	-
	I2MC	0.92	0.10	0	0	-
	RULE	0.94	0.91	0.89	0.65	-
	HOV & knn5	0.97	0.73	0.91	0.70	-
	HOV & knn10	**0.98,**	0.70	0.91	0.70	-
	HOV & knn15	**0.98**	0.69	0.91	0.70	-
	HOV & knn20	**0.98**	0.68	0.91	0.70	-
	HOV & tree1	0.97	0.92	0.88	0.72	-
	HOV & tree2	0.97	0.91	0.88	0.72	-
	HOV & tree3	0.97	0.92	0.88	0.73	-
	HOV & tree4	0.97	0.92	0.88	0.72	-
	HOV & tree5	0.97	0.92	0.88	0.77	-
	Proposed	**0.98**	**0.95**	**0.94**	0.89	-

TABLE III
THE PRECISION FOR EACH EYE MOVEMENT TYPE <u>WITH ERRORS</u> IN THE
INPUT DATA. PSM STANDS FOR POST SACCADIC MOVEMENT.

Data	Alg.	Percision				
		Fixation	*Saccade*	*Pursuit*	*Noise*	*PSM*
DS-AND	EV	0.68	0.37	0	0.73	0.03
	IBDT	0.72	0.70	0.35	0	0
	LS	0.82	0.33	0.63	0.06	0
	I2MC	0.09	0.08	0	0	0
	RULE	0.79	0.65	0.82	0.59	0.32
	Proposed	**0.91**	**0.81**	**0.90**	**0.88**	**0.73**
DS-SAN	EV	0.78	0.31	0	0.01	-
	IBDT	0.93	0.73	0.76	0	-
	LS	0.77	0	0.23	0	-
	I2MC	0.76	0.071	0	0	-
	RULE	**0.98**	0.86	0.89	0.61	-
	HOV & knn5	0.96	0.91	0.91	0.82	-
	HOV & knn10	0.96	0.93	0.92	0.88	-
	HOV & knn15	0.96	0.94	0.91	0.87	-
	HOV & knn20	0.95	**0.95**	0.91	0.89	-
	HOV & tree1	0.97	0.90	0.89	0.79	-
	HOV & tree2	0.97	0.91	0.89	0.78	-
	HOV & tree3	0.97	0.91	0.89	0.84	-
	HOV & tree4	0.97	0.90	0.89	0.79	-
	HOV & tree5	0.97	0.91	0.88	0.84	-
	Proposed	**0.98**	**0.95**	**0.94**	**0.91**	-

considered the annotated data points. For our approach, we used a four fold cross validation where the data of one subject can only be in one fold. The training configurations and machine learning approaches used together with the HOV feature are shown in Table I. As can be seen we applied three well known machine learning approaches namely k nearest neighbors (knn), decision trees (tree), and support vector machines (svm) with different configurations. Table II and Table III show the results for recall ($TP/(TP+FN)$) and precision ($TP/(TP+FP)$). TP are true positives, FP are false positives, and FN are false negatives. Recall therefore stands for the amount of correctly detected eye movement samples. In contrast to this, precision allows to evaluate the reliability of predictions.

Table II shows that our approach outperforms the other state of the art approaches on both data sets for nearly all eye movement types based on pure correct predictions (Recall). It has to be mentioned that we used the raw input data without smoothing out errors as it is done, for example, in IBDT. For Fixations, the HOV feature in combination with the KNN machine learning algorithm performs equal to our fully convolutional network on the DS-SAN data set. However, For the other eye movement types, our approach outperforms the HOV feature with KNN. For the not machine learning based approaches (EV, IBDT, LS, I2MC), it can be seen that they perform well only for the data with a frequency they are designed for. One example for this is IBDT, which performs well on DS-SAN but not on DS-AND due to the higher frequency. In addition, they can not detect eye movement types that are not included by the creator of the algorithm. The best example for this is I2MC which can only detect fixations and saccades. Another issue with the data sets itself is that the annotations change especially for saccades. In the DS-SAN data set for example, saccades are annotated after the velocity peek while in DS-AND the velocity profile of the saccade is annotated. This issue makes it impossible to do cross data set evaluations.

Table III shows that our approach outperforms the other state of the art approaches on both data sets for all eye movement types based on the reliability of the predictions (Precision). In combination with Table II, this means that our approach does not only detect a majority of the eye movement types correctly, it is also more reliable in its detections. Since our approach did not reach 100% for each eye movement type and the variety of challenges in the real world can not easily be covered by scientific data sets, we think that the eye movement detection is still an open problem. The benefits of our approach is the simple realization with modern neuronal network toolboxes. In addition, it is adaptable to new eye movements and varying annotations but this is true for all machine learning based approaches.

B. Evaluation Reconstruction

For the reconstruction, we also used both data sets ([40] & [20]). For the evaluation, a random file was selected one hundred times from the test data set. In this file, a hundred random length and random start positions were selected to extract sections out of the document. In case one section

Data set	Induced Error	Entire Input Sequence Absolut Error	Induced Errors Only Absolut Error
DS-AND	5%	1.76 px	19.26 px
	10%	2.06 px	22.37 px
	15%	2.37 px	28.84 px
	20%	2.72 px	28.84 px
	25%	3.01 px	31.68 px
	30%	3.29 px	34.38 px
DS-SAN	5%	1.62 px	7.06 px
	10%	1.68 px	7.25 px
	15%	1.69 px	7.38 px
	20%	1.70 px	7.41 px
	25%	1.72 px	7.46 px
	30%	1.79 px	7.67 px

Fig. 5. The y axis represents the mean induced error scaled to a maximum of 1. The y axis represent the deviation percentage of the mean absolute error for the error correction in relation to the maximal mean induced error.

already contained errors, it was discarded and another section was selected. This approach was chosen to evaluate the reconstruction, as our method is not intended to reconstruct an error. To evaluate the reconstruction quality, we injected several fixed percentage amounts of errors for each section. These errors were either the setting of a zero or a random number. Each position for an error injection was selected randomly. As a measure of the quality of the reconstruction, we used the mean absolute error. In addition, we visualized the reconstruction error along the amount of errors injected.

Table IV shows the results for our reconstruction experiment. The first column shows the data set and the second column the amount of induced errors as percentage. As can be seen, the errors for the DS-AND are higher in comparison to the DS-SAN data set. This is due to the higher resolution where the gaze points are mapped. The upper part shows the mean absolute error for reconstructing the entire input sequence. Since the neural network sees already a majority of values for reconstruction, the error is low. In contrast to this, the lower part of Table IV evaluates values only that where changed (Induced error). The reconstruction error for those values increases, but it is interesting to see that for the data set DS-SAN the amount of induced error has only a slight impact in comparison to DS-AND. This is due to the sampling rate and that it is more likely to hit large position changes (Saccades) in DS-AND since Saccades in DS-SAN are only one or a few consecutive samples.

Figure 5 shows the mean absolute error for each induced error in a input sequence as a percentage to the maximum induced error (y axis). In addition, the x axis shows the induced error normalized to 1. Each error input sequence is represented as a red cross. As can be seen for both data sets is that the upper bound is at 2% with some outliers at 3% (DS-AND) and 4% (DS-SAN). The error in DS-AND behaves as expected. Meaning, that a larger error causes a larger reconstruction error. However, it is not the case for DS-SAN. Figure 5 clearly shows that the minimum reconstruction error is around 90% of the induced error. The reason for this is that the x axis represents the mean induced error of an input sequence and the y axis the mean reconstruction error for those errors. Since in DS-SAN it is less likely to hit a saccade, the mean induced error is most likely to be at 90%. This can also be seen by the population of red crosses in Figure 5, which are most likely around these 90%. Therefore, our model learned to compensate for this linearly with the bias term of the neurons.

C. Evaluation Generation

Since we cannot directly compare generated scanpaths with other scanpaths, we decided to use a classification experiment. The data we used is from the ETRA 2019 challenge [64], [65]. It consists of 960 trials with a recording length of 45 seconds each. The recorded task are visual fixation, visual search and visual exploration. Since the visual fixation does not hold much complexity for generation, we omitted the data from our experiments. In addition four different stimuli were used in the experiments, which are: Blank, natural, where is waldo, and picture puzzle. For the classification itself, we used the approach proposed in [3]. This approach consists of transforming the eye tracking data into images. These images contain the raw gaze data as dots in the red channel, the time is encoded into the blue channel and the green channel contains the path as lines between raw gaze points. As classifier, we used the same network as proposed in [3].

The classification experiment consists of two parts. The first part uses our VAE to generate one thousand new examples for each stimulus since the exploration and search task are both marked as free vieweing in the data set. Afterwards, we used the classification network to predict the stimuli on the generated data. For training, of the classifier we used 50% of the data set. The other 50% were used for training the generators, where each stimuli type was trained separately. For the generated scanpath, it has to be mentioned that they where centered on the image based on their mean value. This was done to avoid blank images and images where the scanpath is only partially drawn. For the second experiment, we used the VAE to generate data to improve the classification result. Therefore, we trained the generator on the same 50% and used the other 50% only for validation. This means that the generator and the classifier used the same real data for training where the classifier was also trained on additional 1,000 generated scanpath per stimuli. The generated scanpath was centered as mentioned before. For training of the classifier, we used the same parameters as in [3]. In addition, we

		Blank	Natural	Puzzle	Waldo	Accuracy
Real Data	Blank	17	12	0	3	0.531
	Natural	3	50	0	3	0.892
	Puzzle	1	1	58	0	0.966
	Waldo	0	6	2	52	0.866
Gen. Data	Blank	386	350	59	205	0.386
	Natural	267	422	149	162	0.422
	Puzzle	400	73	419	108	0.419
	Waldo	281	181	45	493	0.493
Gen. Train	Blank	21	11	0	0	0.656
	Natural	2	53	0	1	0.946
	Puzzle	0	0	60	0	1.0
	Waldo	0	2	1	57	0.95

used more advanced augmentation techniques. First, we added random noise by shifting a gaze point with a chance of 20% around 10% of its original location. The second augmentation technique was shifting the scanpath around 30% of its original central position (mean). Additionally, we used cropping of the input data which means that we used between 50-100% of a scanpath. Therefore, we selected the cropping length and starting index randomly. Table V shows the results for both experiments. The upper part (Real Data) is the evaluation of the classification on the test set. As can be seen, we achieved similar results as in [3]. For the first experiment, we want to evaluate or generated examples based on the classification. This is shown in the central part in Table V (Gen. Data). As can be seen, all stimuli achieved a classification accuracy above chance level (25%). This can be interpreted as our generated examples contain information about the gaze behavior from the specific stimuli. In addition, for each Stimuli, the second most classified target is Blank. Those two observations mean that, either our generated data can be mapped to random gaze behavior (Blank means the screen only contains the gray color), or that it contains useful information that could not be learned from the training data so far. Therefore, we conducted our second experiment where we used additionally 4,000 generated examples for training. The results can be seen in the lower part in Table V (Gen. Train). As you can see by the results, the generated data is helpful in improving the classification results. One reason for this is that the model has to learn different combinations of gaze behavior and thus rather learns important patterns.

V. CONCLUSION

We showed that, based on the input tensor construction, it is possible to use raw eye tracking data with fully convolutional neural networks for multiple tasks. They have the additional advantage that they can be used with any input size. In our results, we are improving the state of the art in the field of eye movement classification. Our main contribution in this area, however, is the construction of the input tensor as well as the pre-initialization of the first layer. This allows the use of raw data and makes this approach easy to use. In addition, the same approach can be used to improve data quality for experiments already performed.

REFERENCES

[1] W. Fuhl, N. Castner, and E. Kasneci, "Histogram of oriented velocities for eye movement detection," in *Proceedings of the Workshop on Modeling Cognitive Processes from Multimodal Data*. ACM, 2018, p. 5.

[2] ——, "Rule-based learning for eye movement type detection," in *Proceedings of the Workshop on Modeling Cognitive Processes from Multimodal Data*. ACM, 2018, p. 9.

[3] W. Fuhl, E. Bozkir, B. Hosp, N. Castner, D. Geisler, T. C. Santini, and E. Kasneci, "Encodji: Encoding gaze data into emoji space for an amusing scanpath classification approach ;)," in *Eye Tracking Research and Applications*, 2019.

[4] W. Fuhl, N. Castner, T. C. Kbler, A. Lotz, W. Rosenstiel, and E. Kasneci, "Ferns for area of interest free scanpath classification," in *Proceedings of the 2019 ACM Symposium on Eye Tracking Research & Applications (ETRA)*, 06 2019.

[5] S. Eivazi, A. Hafez, W. Fuhl, H. Afkari, E. Kasneci, M. Lehecka, and R. Bednarik, "Optimal eye movement strategies: a comparison of neurosurgeons gaze patterns when using a surgical microscope," *Acta Neurochirurgica*, 2017.

[6] H. Bahmani, W. Fuhl, E. Gutierrez, G. Kasneci, E. Kasneci, and S. Wahl, "Feature-based attentional influences on the accommodation response," in *Vision Sciences Society Annual Meeting Abstract*, 2016.

[7] S. Eivazi, M. Slupina, W. Fuhl, H. Afkari, A. Hafez, and E. Kasneci, "Towards automatic skill evaluation in microsurgery," in *Proceedings of the 22st International Conference on Intelligent User Interfaces, IUI 2017*. ACM, 03 2017.

[8] S. Eivazi, W. Fuhl, and E. Kasneci, "Towards intelligent surgical microscopes: Surgeons gaze and instrument tracking," in *Proceedings of the 22st International Conference on Intelligent User Interfaces, IUI 2017*. ACM, 03 2017.

[9] J. G. May, R. S. Kennedy, M. C. Williams, W. P. Dunlap, and J. R. Brannan, "Eye movement indices of mental workload," *Acta psychologica*, vol. 75, no. 1, pp. 75–89, 1990.

[10] D. Geisler, W. Fuhl, T. Santini, and E. Kasneci, "Saliency sandbox: Bottom-up saliency framework," in *12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, 02 2017.

[11] W. Fuhl, T. Kbler, T. Santini, and E. Kasneci, "Automatic generation of saliency-based areas of interest," in *Symposium on Vision, Modeling and Visualization (VMV)*, 09 2018.

[12] W. Fuhl, G. Kasneci, W. Rosenstiel, and E. Kasneci, "Training decision trees as replacement for convolution layers," in *Conference on Artificial Intelligence, AAAI*, 02 2020.

[13] W. Fuhl and E. Kasneci, "Learning to validate the quality of detected landmarks," in *International Conference on Machine Vision, ICMV*, 11 2019.

[14] W. Fuhl, T. C. Kbler, H. Brinkmann, R. Rosenberg, W. Rosenstiel, and E. Kasneci, "Region of interest generation algorithms for eye tracking data," in *Third Workshop on Eye Tracking and Visualization (ETVIS), in conjunction with ACM ETRA*, 06 2018.

[15] W. Fuhl, T. C. Kbler, K. Sippel, W. Rosenstiel, and E. Kasneci, "Arbitrarily shaped areas of interest based on gaze density gradient," in *European Conference on Eye Movements, ECEM 2015*, 08 2015.

[16] Q. Ji, Z. Zhu, and P. Lan, "Real-time nonintrusive monitoring and prediction of driver fatigue," *IEEE transactions on vehicular technology*, vol. 53, no. 4, pp. 1052–1068, 2004.

[17] R. J. Leigh and D. S. Zee, *The neurology of eye movements*. Oxford University Press, USA, 2015, vol. 90.

[18] A. Siekawa, M. Chwesiuk, R. Mantiuk, and R. Piórkowski, "Foveated ray tracing for vr headsets," in *International Conference on Multimedia Modeling*. Springer, 2019, pp. 106–117.

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

[20] R. Andersson, L. Larsson, K. Holmqvist, M. Stridh, and M. Nyström, "One algorithm to rule them all? an evaluation and discussion of ten eye movement event-detection algorithms," *Behavior Research Methods*, vol. 49, no. 2, pp. 616–637, 2017.

[21] W. Fuhl, T. Santini, C. Reichert, D. Claus, A. Herkommer, H. Bahmani, K. Rifai, S. Wahl, and E. Kasneci, "Non-intrusive practitioner pupil detection for unmodified microscope oculars," *Elsevier Computers in Biology and Medicine*, vol. 79, pp. 36–44, 12 2016.

[22] W. Fuhl, T. C. Kbler, D. Hospach, O. Bringmann, W. Rosenstiel, and E. Kasneci, "Ways of improving the precision of eye tracking data: Controlling the influence of dirt and dust on pupil detection," *Journal of Eye Movement Research*, vol. 10, no. 3, 05 2017.

[23] W. Fuhl, D. Geisler, T. Santini, T. Appel, W. Rosenstiel, and E. Kasneci, "Cbf:circular binary features for robust and real-time pupil center detection," in *ACM Symposium on Eye Tracking Research & Applications*, 06 2018.

[24] W. Fuhl, H. Gao, and E. Kasneci, "Tiny convolution, decision tree, and binary neuronal networks for robust and real time pupil outline estimation," in *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM, 01 2020.

[25] W. Fuhl, T. Santini, and E. Kasneci, "Fast camera focus estimation for gaze-based focus control," in *CoRR*, 2017.

[26] W. Fuhl, S. Eivazi, B. Hosp, A. Eivazi, W. Rosenstiel, and E. Kasneci, "Bore: Boosted-oriented edge optimization for robust, real time remote pupil center detection," in *Eye Tracking Research and Applications, ETRA*, 2018.

[27] W. Fuhl, D. Geisler, W. Rosenstiel, and E. Kasneci, "The applicability of cycle gans for pupil and eyelid segmentation, data generation and image refinement," in *International Conference on Computer Vision Workshops, ICCVW*, 11 2019.

[28] W. Fuhl, W. Rosenstiel, and E. Kasneci, "500,000 images closer to eyelid and pupil segmentation," in *Computer Analysis of Images and Patterns, CAIP*, 11 2019.

[29] W. Fuhl, N. Castner, L. Zhuang, M. Holzer, W. Rosenstiel, and E. Kasneci, "Mam: Transfer learning for fully automatic video annotation and specialized detector creation," in *International Conference on Computer Vision Workshops, ICCVW*, 2018.

[30] W. Fuhl, H. Gao, and E. Kasneci, "Neural networks for optical vector and eye ball parameter estimation," in *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM, 01 2020.

[31] W. Fuhl, T. Santini, T. Kuebler, N. Castner, W. Rosenstiel, and E. Kasneci, "Eye movement simulation and detector creation to reduce laborious parameter adjustments," *eprint arXiv:1804.00970*, 2018.

[32] W. Fuhl and E. Kasneci, "Eye movement velocity and gaze data generator for evaluation, robustness testing and assess of eye tracking software and visualization tools," in *Poster at Egocentric Perception, Interaction and Computing, EPIC*, 2018.

[33] D. D. Salvucci and J. H. Goldberg, "Identifying fixations and saccades in eye-tracking protocols," in *Eye Tracking Research and Applications*. ACM, 2000, pp. 71–78.

[34] H. Widdel, "Operational problems in analysing eye movements," *Advances in psychology*, vol. 22, pp. 21–29, 1984.

[35] R. Engbert and R. Kliegl, "Microsaccades uncover the orientation of covert attention," *Vision research*, vol. 43, no. 9, pp. 1035–1045, 2003.

[36] O. V. Komogortsev and J. I. Khan, "Eye movement prediction by oculomotor plant kalman filter with brainstem control," *Journal of Control Theory and Applications*, vol. 7, no. 1, pp. 14–22, 2009.

[37] O. V. Komogortsev, D. V. Gobert, S. Jayarathna, D. H. Koh, and S. M. Gowda, "Standardization of automated analyses of oculomotor fixation and saccadic behaviors," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 11, pp. 2635–2645, 2010.

[38] G. Veneri, P. Piu, P. Federighi, F. Rosini, A. Federico, and A. Rufa, "Eye fixations identification based on statistical analysis-case study," in *Cognitive Information Processing (CIP), 2010 2nd International Workshop on*. IEEE, 2010, pp. 446–451.

[39] G. Veneri, P. Piu, F. Rosini, P. Federighi, A. Federico, and A. Rufa, "Automatic eye fixations identification based on analysis of variance and covariance," *Pattern Recognition Letters*, vol. 32, no. 13, 2011.

[40] T. Santini, W. Fuhl, T. Kübler, and E. Kasneci, "Bayesian identification of fixations, saccades, and smooth pursuits," in *Eye Tracking Research and Applications*. ACM, 2016, pp. 163–170.

[41] M. Nyström and K. Holmqvist, "An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data," *Behavior research methods*, vol. 42, no. 1, pp. 188–204, 2010.

[42] R. van der Lans, M. Wedel, and R. Pieters, "Defining eye-fixation sequences across individuals and tasks: the binocular-individual threshold (bit) algorithm," *Behavior Research Methods*, vol. 43, no. 1, 2011.

[43] L. Larsson, M. Nyström, and M. Stridh, "Detection of saccades and postsaccadic oscillations in the presence of smooth pursuit," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 9, 2013.

[44] L. Larsson, M. Nyström, R. Andersson, and M. Stridh, "Detection of fixations and smooth pursuit movements in high-speed eye-tracking data," *Biomedical Signal Processing and Control*, vol. 18, pp. 145–152, 2015.

[45] S. Hoppe and A. Bulling, "End-to-end eye movement detection using convolutional neural networks," *arXiv preprint arXiv:1609.02452*, 2016.

[46] R. Zemblys, D. C. Niehorster, O. Komogortsev, and K. Holmqvist, "Using machine learning to detect events in eye-tracking data," *Behavior research methods*, vol. 50, no. 1, pp. 160–181, 2018.

[47] S. P. Lee, J. B. Badler, and N. I. Badler, "Eyes alive," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002.

[48] X. Ma and Z. Deng, "Natural eye motion synthesis by modeling gaze-head coupling," in *IEEE Virtual Reality Conference*. IEEE, 2009.

[49] H. Murphy and A. T. Duchowski, "Perceptual gaze extent & level of detail in vr: looking outside the box," in *ACM SIGGRAPH conference Abstracts and Applications*. ACM, 2002, pp. 228–228.

[50] B. H. Le, X. Ma, and Z. Deng, "Live speech driven head-and-eye motion generators," *IEEE transactions on Visualization and Computer Graphics*, vol. 18, no. 11, pp. 1902–1914, 2012.

[51] A. T. Duchowski and S. Jörg, "Modeling physiologically plausible eye rotations," *Proceedings of Computer Graphics International*, 2015.

[52] D. Tweed, W. Cadera, and T. Vilis, "Computing three-dimensional eye position quaternions and eye velocity from search coil signals," *Vision research*, vol. 30, no. 1, pp. 97–110, 1990.

[53] E. Wood, T. Baltrusaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling, "Rendering of eyes for eye-shape registration and gaze estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3756–3764.

[54] T. Pejsa, B. Mutlu, and M. Gleicher, "Stylized and performative gaze for character animation," in *Computer Graphics Forum*, vol. 32, no. 2pt2. Wiley Online Library, 2013, pp. 143–152.

[55] D. J. Campbell, J. Chang, K. Chawarska, and F. Shic, "Saliency-based bayesian modeling of dynamic viewing of static scenes," in *Eye Tracking Research and Applications*. ACM, 2014, pp. 51–58.

[56] A. Duchowski, S. Jörg, A. Lawson, T. Bolte, L. Świrski, and K. Krejtz, "Eye movement synthesis with 1/f pink noise," in *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*. ACM, 2015, pp. 47–56.

[57] A. T. Duchowski, S. Jörg, T. N. Allen, I. Giannopoulos, and K. Krejtz, "Eye movement synthesis," in *Eye Tracking Research and Applications*. ACM, 2016.

[58] D. Simon, S. Sridharan, S. Sah, R. Ptucha, C. Kanan, and R. Bailey, "Automatic scanpath generation with deep recurrent neural networks," in *Proceedings of the ACM Symposium on Applied Perception*. ACM, 2016, pp. 130–130.

[59] M. Assens, X. Giro-i Nieto, K. McGuinness, and N. E. O'Connor, "Pathgan: visual scanpath prediction with generative adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[60] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[62] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[63] R. S. Hessels, D. C. Niehorster, C. Kemner, and I. T. Hooge, "Noise-robust fixation detection in eye movement data: Identification by two-means clustering (i2mc)," *Behavior research methods*, vol. 49, no. 5, pp. 1802–1823, 2017.

[64] J. Otero-Millan, X. G. Troncoso, S. L. Macknik, I. Serrano-Pedraza, and S. Martinez-Conde, "Saccades and microsaccades during visual fixation, exploration, and search: foundations for a common saccadic generator," *Journal of vision*, vol. 8, no. 14, pp. 21–21, 2008.

[65] M. B. McCamy, J. Otero-Millan, L. L. Di Stasi, S. L. Macknik, and S. Martinez-Conde, "Highly informative natural scene regions increase microsaccade production during visual scanning," *Journal of neuroscience*, vol. 34, no. 8, pp. 2956–2966, 2014.