Eye Movement Classification with Temporal Convolutional Networks[⋆]

Carlos Elmadjian, Candy Gonzales, and Carlos H. Morimoto

University of São Paulo, São Paulo, Brazil
{elmad,candytg,hitoshi}@ime.usp.br

Abstract. Recently, deep learning approaches have been proposed to detect eye movements such as fixations, saccades, and smooth pursuits from eye tracking data. These are *end-to-end* methods that have shown to surpass traditional ones, requiring no *ad hoc* parameters. In this work we propose the use of temporal convolutional networks (TCNs) for automated eye movement classification and investigate the influence of feature space, scale, and context window sizes on the classification results. We evaluated the performance of TCNs against a state-of-the-art 1D-CNN-BLSTM model using GazeCom, a public available dataset. Our results show that TCNs can outperform the 1D-CNN-BLSTM, achieving an F-score of 94.2% for fixations, 89.9% for saccades, and 73.7% for smooth pursuits on sample level, and 89.6%, 94.3%, and 60.2% on event level. We also state the advantages of TCNs over sequential networks for this problem, and how these scores can be further improved by feature space extension.

Keywords: Eye movement classification · Temporal convolutional networks · Feature selection.

1 Introduction

Eye movement classification is a fundamental task in many areas of research, including Psychology and Human-Computer Interaction. Many social, behavioral, and user experience studies depend on the correct identification of certain eye movement events [8]. Moreover, these raw events often make up more complex activity patterns (i.e., reading [5]), and thus are an elementary step in order to understand intricate cognitive states [21, 25].

Although we perform several different types of movements with our eyes [19], fixations, saccades, and smooth pursuits are the most common used for classification purposes [16]. While fixations come down as the act of staring at something, saccades and smooth pursuits are considered gaze shifts. Typically, saccades are very fast and have a short duration, while pursuits are slower and usually only triggered when following a moving target in the visual field [6] (see Figure 1).

The task of simultaneously classifying these three kinds of movements has been coined the "tertiary eye movement classification problem" (3EMCP), and

⋆ Supported by the São Paulo Research Foundation, grant 2015/26802-1.

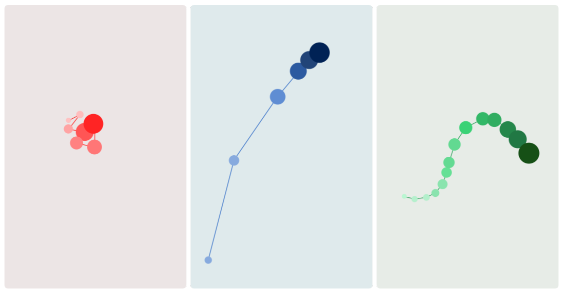

Fig. 1: Graphical representation of gaze scanpaths of fixations (left), saccades (middle), and smooth pursuit events (right). Fixations are markedly stationary, saccades present large ballistic gaze shifts, while pursuits show minor displacements, typically shaped by following a moving target.

it is still considered a challenging one [4]. Early solutions to this problem often relied on threshold levels for pattern segmentation [22], being later replaced by probabilistic-based models, which are considerably more robust to noise [17, 23]. Only recently data-driven approaches have been employed more extensively [12, 26], becoming the current state-of-the-art.

In the realm of deep neural networks, the 3EMCP can be considered a typical sequence-to-sequence (seq2seq) problem [28], in which we want a label for each individual sample that goes into the model. Although recurrent architectures such as LSTMs and GRUs have dominated the field of seq2seq problems in the last decade, recently it has been shown that Temporal Convolutional Networks (TCNs) can outperform these recurrent models in many applications [2].

Therefore, in this work we propose a TCN model for the tertiary eye movement classification problem, and we investigate its performance, architectural advantages, and shortcomings in comparison to the 1D-CNN-BLSTM model proposed by Startsev et al. [26], which is currently the best performing model in the GazeCom dataset [7]. Additionally, we investigate the role of different features, feature scale size, and temporal windows on general model performance, and we show how deep architectures in general can benefit from these findings in the domain of eye movement classification.

2 Related Work

Although not all related algorithms and models have focused specifically on the tertiary eye movement classification problem, they all fall into one of the following categories: threshold-based methods, probabilistic methods, or data-driven models, such as deep learning approaches.

2.1 Threshold-based methods

Threshold-based methods are the ones that make use of threshold levels to discriminate different classes of eye movements. They are often marked by their simplicity and low computational requirements, making them suitable for real-time applications.

Velocity threshold is a widely use criterion to cut apart saccades from other eye movements, as saccades are commonly characterized by their fast displacements. Salvucci and Goldberg [22] showed that it is possible to use a simple threshold to discriminate saccades from fixations by computing the point-to-point velocity in eye data stream.

However, because a static threshold can be severely affected by noise, equipment, or user attributes, Nyström and Holmqvist [20] proposed an adaptive velocity threshold to classify fixations, saccades, and post-saccadic oscillations (PSOs), a method that is particularly less sensitive to noise level fluctuations. Since their work focused on saccade and PSO detection, fixations were treated as a negative class.

Dispersion threshold is another form of criterion found in the literature. It provides a more robust approach to detect fixations compared to the velocity threshold [22]. The idea is that we can treat fixations as consecutive samples within maximum spatial separation under a minimum duration threshold.

Berg et al. [3] demonstrated that it is also possible to use Principal Component Analysis (PCA) to establish more generic dispersion thresholds that are able to discriminate fixations, saccades, and smooth pursuits. This approach, however, is very parameter-sensitive, which makes it not so adaptive.

2.2 Probabilistic methods

In the pursuit of methods that are less sensitive to noise and are also parameter-free, several authors proposed what we call *probabilistic* approaches.

Komogortsev and Khan [17] designed the Attention Focus Kalman Filter (AFKF) framework, which aimed to solve the 3EMCP in real-time. The AFKF framework was in fact an extension of a previous method [24], in which a Kalman filter with a χ^2-test is used to detect saccades. In this updated version, Komogortsev and Khan proposed that smooth pursuits could be treated as a negative event (i.e., neither saccades or fixations), provided they did not surpass the velocity of 140 deg/s.

A more reliable 3EMCP solution was presented by Santini et al. [23] through what they called Bayesian Decision Theory Identification (I-BDT) algorithm. The method consisted on defining priors and likelihoods for all events, and then calculating the posterior for each event, given eye velocity and movement ratio over windows according to the Bayes' rules.

2.3 Data-driven methods

Data-driven methods emerged with the increasing success and growth of machine learning techniques. Because models in this class are typically implicit, they tend to be more adaptive and robust than previous algorithms, often displaying a large capacity (i.e., the ability to fit a wide variety of functions).

Some of these methods resort to classical machine learning techniques, such as the one proposed by Vidal et al. [29]. They considered a set of shape features that characterize smooth pursuits as a statistical phenomenon, training a k-nearest neighbor classifier with them. Another example is the work of Zemblys et al. [31], in which they introduced a random forests-based model to detect fixations, saccades and PSOs.

With the prominence of deep learning, some authors introduced artificial neural models for this problem as well. Hoppe and Bulling [12] proposed a CNN-based end-to-end architecture to classify fixations, saccades, and smooth pursuits in a continuous gaze stream, from which frequency features are extracted to train their model. Later, Starsev et al. [26] introduced a 1D-CNN-BLSTM *seq2seq* network that also addresses the 3EMCP. Besides the three eye movement classes, they also added a "noise" class, so that the model was not forced to choose between one of the three. They showed that their 1D-CNN-BLSTM classifier outperforms 12 previous baseline models in the GazeCom dataset ([7, 1, 18, 3, 14, 15]), effectively becoming the current state-of-the-art.

It is also worth noting that since performance of neural networks is highly impacted by the amount of data available and its variability on training [10], some authors suggested gaze data augmentation techniques for this problem. Zemblys et al. [30] proposed gazeNet, a generative network that can create different kinds of eye movement events. More recently, Fuhl [9] introduced CNNs for raw eye tracking data segmentation, generation, and reconstruction.

Our TCN model is also part of this class. Although — to the best of our knowledge — there are no previous studies of TCNs on eye movement classification, this kind of network has already demonstrated that it can outperform recurrent architectures in a varied set of sequence modeling tasks [2]. We describe our architecture and its features in the following section.

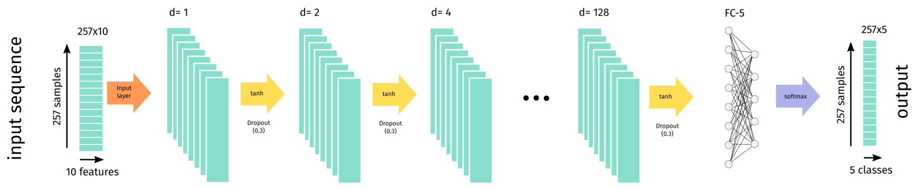

Fig. 2: Architecture of our TCN network. At each hidden state we increase dilations by a factor of 2. FC-5 represents a time-distributed fully-connected layer with 5 outputs.

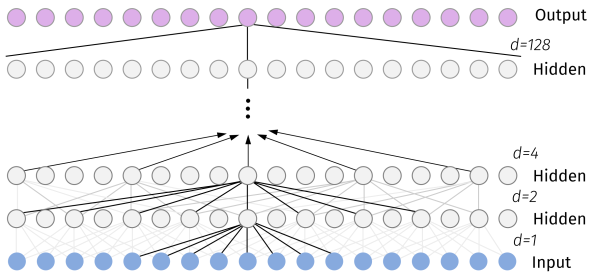

Fig. 3: Non-causal dilated convolutions using size 8 kernels, providing a large receptive field and the ability to look into the "future".

3 Model Architecture

Compared to LSTMs or GRUs, TCNs present some important advantages. TCNs are known to have longer memory than recurrent nets with the same capacity since they do not have gating mechanisms. They also have a flexible receptive field size, lower memory requirements for training, and can be trained with variable input length. Also, because TCNs have a convolutional structure, it can be more easily parallelized, meaning less training time than recurrent nets, that present a more sequential pipeline structure [2].

In broader terms, TCNs can be understood as 1D fully-convolutional networks with causal convolutions. This means that given a time instant t in a time series and a constraint y_t, then y_t can only be satisfied by $x_0, ..., x_t$, and not by $x_{t+1}, ..., x_T$ [2]. This makes them suitable for real-time applications, but also puts them in a disadvantageous position against bidirectional recurrent architectures, as they can process features in two directions.

To make up for this, we use non-causal TCNs instead, which gives our architecture the ability to look into the future at the expense of being offline. We also make use of several dilated convolutions $(1, 2, 4, 8, 16, 32, 64, 128)$, and a relatively sizeable kernel ($k = 8$), which effectively provides us with a large receptive field (see Figure 3). This convenience does have a cost though, as our network may require almost 2 million parameters to train. However, despite the amount of parameters required, training is comparatively fast due to its parallelized structure, roughly taking half of the time required to train the 1D-CNN-BLSTM on the same machine to reach an empirical optimum validation loss in the GazeCom dataset [7].

The schematic description of our architecture is depicted in Figure 2. All hyperparameters were defined according to a reduced grid search with a subset of the GazeCom dataset. The TCN has a total of 128 filters, an internal dropout rate of 0.3, and uses the hyperbolic tangent activation function ($tanh$) instead of the more commonly seen rectified linear units ($ReLU$). Weights are

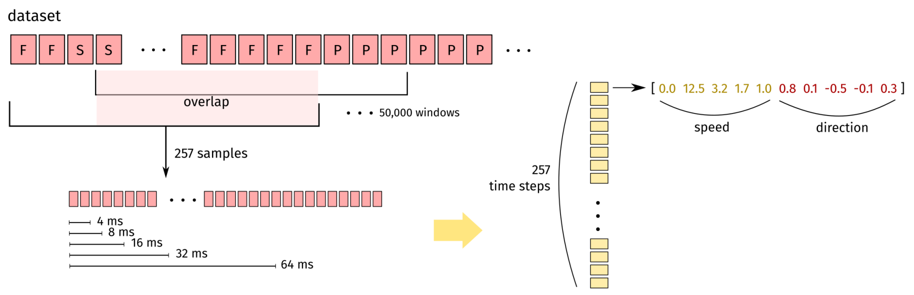

Fig. 4: Feature extraction procedure from the GazeCom dataset, according to Startsev et al. [26]. For each 257-sample window we extract speed and direction features related to each step of the sequence, using different scale sizes.

initialized following a random uniform distribution, and we do not employ batch normalization between layers.

At the end of the convolutional chain, we add a fully connected layer with a softmax activation function wrapped by a timed-distributed layer, so that we can have a certain probability distribution for each eye movement pattern at all time steps. Because of our one-hot encoded targets, we use a categorical cross-entropy loss function. We also opted for the Adam optimizer [13] with a initial learning rate of 0.0001, as this configuration demonstrated a faster convergence in our grid search.

4 Evaluation

4.1 Materials

Our network implementation was based on the publicly available code base from Startsev et al. [26], but we ported all their original training and evaluation routines to Python 3. We also created a customized Python 3 version of their feature extraction tool, which was originally written in MATLAB. The architecture itself was implemented and trained using the functional API from Keras (2.3.1), which is a built-in high-level API for Tensorflow (2.0.2).

For the TCN, in particular, we used the keras-tcn package (3.1.1), available at the Python Package Index (PyPI). All the training was done distributedly on two machines with an Intel Core i7-7700 CPU with 16GB RAM, both of them with a NVidia GeForce GTX 1070 GPU (8GB VRAM), running on the Ubuntu 18.04 operating system.

All our code base, data, and models trained for this work are publicly available at `https://github.com/elmadjian/3EMCP-with-TCNs`.

4.2 Dataset

The dataset employed in this study is the GazeCom dataset [7], the same one that was used by the 1D-CNN-BLSTM model proposed by Startsev et al. It contains 18 clips from roughly 47 viewers (the number slightly varies according to the video), with labels for fixations, saccades, smooth pursuits, and noise. The labels are given based on the judgment and agreement of two experts, after observing the patterns that each user performed when watching short videos. Clips are limited to a duration of 21 s for training.

The GazeCom data was collected using a 250 Hz remote eye tracker, and is constituted by 4.3 million samples, being 72.5% fixations, 10.5% saccades, 11% smooth pursuits, and 5.9% noise or blinks. The average confidence level given by the eye tracker is 99.4%. The average duration of each pattern is shown in Table 1.

Table 1: Average event duration in GazeCom dataset

	Duration (ms)	**Deviation (ms)**
Fixations	324.37	306.07
Saccades	46.38	22.17
Smooth Pursuits	410.98	323.42
Noise	291.91	294.34

4.3 Feature extraction

The temporal sequences that are fed to the model are formed by pre-computed multi-scale features extracted from the x and y gaze coordinates. In their original implementation, Startsev et al. considered five different temporal scales: 4, 8, 16, 32, and 64 ms. In ours, we included support to a larger set of scales (128, 256, and 512 ms) in order to investigate their role in model classification performance.

The feature extraction process is depicted in Figure 4. All these features are created by pre-processing the entire set of raw gaze coordinates before training. Each sample consists of a fixed-size context window of roughly 1 s (257 steps), and each timestep of this window presentes the associated multi-scale set of features, which could be either acceleration, speed, or direction, or a combination of them.

Statsev et al. have also demonstrated that classification improves when features such as speed and direction are combined in training, while acceleration deteriorates performance. Thus, we use this composite of speed and direction as our baseline, but we also investigated the addition of other two common features in the time series domain: standard deviation and displacement (not to be confused with distance).

4.4 Metrics

Because the GazeCom dataset has an unbalanced distribution of classes, general accuracy measurements might lead to biased interpretations of actual model performance. That is why we evaluate our model using the F-score and intersection over union (IoU) as our two primary metrics, since they can give more weight to misclassified samples and can be more informative in terms of event correlation.

Although other useful metrics could be applied as well, for comparison fairness we have limited our evaluation scope to the same metrics used by Startsev et al. [27].

4.5 Training and evaluation

In this work, we evaluate the general performance of our model in comparison with a state-of-the-art baseline using the same configuration and metrics; we investigate the role of increasing feature space and scale on model performance; and we take advantage of TCNs' lower memory requirements to examine model classification behavior of smooth pursuits when exposed to larger context windows of sequences.

To compare our model with the baseline, we use the same Leave-One-Video-Out (LOVO) cross-validation procedure reported by Startsev et al., in which each step of training is done with 17 clips and the model is evaluated on the remaining one. We also employ the same input setup, with a context window of 257 samples ($\sim$1 s), an overlap of 192 samples, with speed and direction features combined, and limiting the number of training sequences to 50,000, using a random permutation of sequences with the same seed. To get more stable and reproducible results, we also fix the random seed generators of the NumPy library and TensorFlow to 0.

While the baseline model was reportedly trained with a batch size of 5000 for 1000 epochs, the TCN model is trained with a batch size of 128 samples for 20 epochs. Another important difference is that our model uses the Adam optimizer, and not RMSprop. Although we did not re-train the baseline model, we ran the evaluation routines for its two trained versions available in a public repository from the authors and obtained the same reported results.

Assuming the standard feature space (SF) as the combination of speed and direction, we evaluated the TCN model performance in this aspect according to the following configurations: SF + standard deviation; SF + displacement; and SF + standard deviation + displacement. Regarding the feature scale size investigation, we evaluated the model with additional scales of 128, 256, and 512 ms. We used the same LOVO procedure while training using the same context window size, training sequences, batch size, and epochs, as reported before.

Since smooth pursuits have a typically longer duration than fixations and saccades, it has been hypothesized that larger context windows can benefit *seq2seq* models to improve its detection. To demonstrate this, we trained our model using two additional windows of 1.5 s and 2 s, using the same LOVO training procedure, but with the overlap having to be adjusted to keep the 50,000 sequences fixed. In other words, we used 385 samples with an overlap of 312 for the 1.5 s window, and 514 samples with an overlap of 470 for the 2 s window.

It is worth mentioning that larger contexts can be prohibitive for recurrent networks in terms of memory usage and computational complexity, but there

Table 2: TCN Model Evaluation

	Fixation				**Saccade**				**Smooth Pursuit**			
Model	**F1**	**Prec.**	**Recall**	**EF1**	**F1**	**Prec.**	**Recall**	**EF1**	**F1**	**Prec.**	**Recall**	**EF1**
TCN Model+	**0.945**	**0.929**	0.961	0.735	0.894	0.899	**0.889**	0.888	0.762	0.787	**0.739**	0.253
TCN Model+*	**0.945**	0.928	0.962	**0.900**	0.892	0.897	0.887	0.941	**0.764**	**0.791**	**0.739**	**0.608**
TCN Model	0.942	0.925	0.959	0.717	**0.899**	**0.903**	0.894	0.885	0.737	0.765	0.711	0.234
TCN Model*	0.942	0.924	0.960	0.896	0.896	0.901	0.890	**0.943**	0.739	0.769	0.711	0.602
Startsev et al. [26]	0.939	0.914	**0.967**	0.868	0.893	0.897	**0.889**	0.924	0.703	0.788	0.634	0.484
Startsev et al.*	0.939	0.914	**0.967**	0.883	0.893	0.897	**0.889**	0.935	0.703	0.789	0.634	0.537
Startsev et al. [27]	0.937	0.921	0.954	0.882	0.896	0.899	0.893	0.929	0.707	0.724	0.691	0.544
Startsev et al.*	0.937	0.921	0.954	0.892	0.896	0.899	0.892	0.939	0.708	0.725	0.692	0.576

Results for sample-level detection, except column **EF1**=Event F1 (IoU>= 0.5). Rows marked with * represent filtered output. **Prec.**= Precision. + = Our best model.

were no such issues training with the TCN model. Based on these findings, we ran a final LOVO evaluation combining the best configurations provided by the study on feature space, feature scale size, and temporal window width.

Finally, we also evaluated all models with and without a domain-based knowledge filtering procedure applied to the outputs. This heuristic assumes that any event that is shorter than 12 ms is a spurious output. The filter then replaces the class of the samples belonging to this event by the the next non-spurious neighbor.

5 Results

Compared to the reported results of the 1D-CNN-BLSTM model, and using the same input constraints and training regimen, our model achieved an improved performance in all three major patterns, with an F-score of 94.2% for fixations, 89.9% for saccades, and 73.4% for smooth pursuits. This represents a gain over the best published result in the GazeCom dataset of 0.3%, 0.3%, and 3%, respectively. If considered the models further improved by window width, feature space, and feature scale size, the highest gains were of 0.6%, 0.3%, and 5.7%, respectively. A complete assessment compiling the reported results in terms of sample F1 and event detection (episode F1 with IoU > 0.5) is shown in Table 2. The TCN model marked by a + sign is the one trained with the best features, while the ones marked with a * had their output filtered.

The use of a domain-based knowledge filtering for the outputs also increased event detection scores according to the metrics proposed by Hooge et al. [11]. With smooth pursuits, in particular, the TCN models roughly tripled their event F-scores, while the models from Startsev et al. had only modest improvements. Besides, the filter effect over general sample-level scores were barely noticeable.

The evaluation on feature space and feature scale size have shown mixed results. It is clear that increasing the scale size led to an overall marginal im-

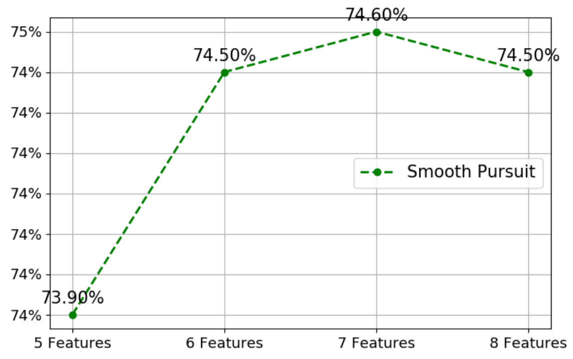

Fig. 5: Smooth pursuit F-score variation across different scale sizes.

Table 3: Feature space study

	Fixation		Saccade		SP	
	F1	**EF1**	**F1**	**EF1**	**F1**	**EF1**
win 257	0.942	0.717	0.899	0.885	0.737	0.234
win 257*	0.942	**0.896**	0.896	0.943	0.739	**0.602**
win 257+std	0.942	0.721	**0.898**	0.890	0.738	0.239
win 257+std*	0.942	**0.896**	0.895	**0.945**	0.739	0.601
win 257+disp	0.942	0.728	0.896	0.880	0.738	0.252
win 257+disp*	0.942	0.895	0.893	0.943	0.740	0.596
win 257+std+disp	0.942	0.730	0.896	0.883	0.740	0.253
win 257+std+disp*	0.942	**0.896**	0.894	0.944	**0.741**	0.600

EF1= Event F1 (IoU >= 0.5), *****=Filtered

provement on classification performance when compared to the default 5-scale features input. However these gains did not occur consistently across scale sizes, and smooth pursuits showed the largest fluctuations. Figure 5 exhibits how the F1 and IoU values for smooth pursuits oscillated with respect to the scale size. Tables 3 and 4 show the results related to the feature space study and feature scale size, respectively.

The investigation on the size of temporal window partially confirmed the hypothesis that smooth pursuits detection can benefit from larger contexts. With a window size of 1.5 s (385 samples), there was an improvement of 1.3% over the 1 s window (257 samples), but with a context of 2 s (514 samples), the gain dropped to 0.7%. These results can be seen in Table 5.

A final evaluation with the best configuration in terms of features and context windows gave us the highest F1 scores achieved for smooth pursuits in the

Table 4: Feature scale study

	Fixation		**Saccade**		**SP**	
	F1	**EF1**	**F1**	**EF1**	**F1**	**EF1**
5 Features	0.942	0.717	**0.899**	0.885	0.737	0.234
5 Features	0.942	0.896	0.896	**0.943**	0.739	0.602
6 Features	**0.943**	0.723	**0.899**	0.887	0.743	0.240
6 Features*	**0.943**	**0.897**	0.896	**0.943**	0.745	**0.613**
7 Features	0.942	0.725	0.898	0.884	0.745	0.244
7 Features*	0.942	0.896	0.895	**0.943**	**0.746**	0.597
8 Features	0.942	0.715	0.897	0.885	0.742	0.235
8 Features*	0.942	0.894	0.894	0.941	0.745	0.599

EF1= Event F1 (IoU >= 0.5), *=Filtered

GazeCom dataset. The results were 94.5% for fixations, 89.2% for saccades, and 76.4% for smooth pursuits.

Table 5: Temporal window width

	Fixation		**Saccade**		**SP**	
	F1	**EF1**	**F1**	**EF1**	**F1**	**EF1**
Win 257	0.942	0.717	0.899	0.885	0.737	0.234
Win 257*	0.942	0.896	0.896	**0.943**	0.739	0.602
Win 385	**0.943**	0.708	0.899	0.884	0.749	0.227
Win 385*	**0.943**	0.898	0.897	**0.943**	**0.751**	0.602
Win 514	0.941	0.694	**0.900**	0.886	0.744	0.213
Win 514*	0.942	**0.898**	0.897	**0.943**	0.747	**0.604**

Results for varying temporal window sizes (257, 358, and 514 samples) of the TCN model. Results marked by * indicate an identical trained model with filtered output.

6 Discussion

Our results show that the TCN architecture outperforms the state-of-the-art model with the GazeCom dataset. The most significant improvement was in the classification of smooth pursuits, with a 3% performance increase, or 5.7% increase over previous published results when considering the increments on context window width, feature space, and scale size (see Figure 6).

The investigation on feature space and scale size revealed that increasing both dimensions were somewhat beneficial to the TCN classification performance. The impact of adding novel features, although favorable, was comparatively less significant than the increase of feature scale size, as the latter provided comparatively higher scores on the GazeCom dataset, particularly with respect to smooth pursuits.

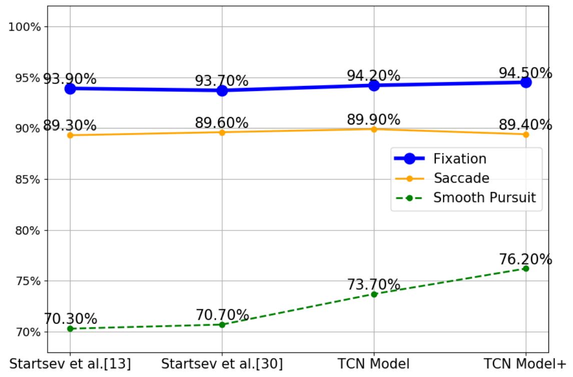

Fig. 6: Results of our TCN model for fixations, saccades, and smooth pursuits (SP), along with the reported results from the 1D-CNN-BLSTM model.

The improvement, though, was not consistent with size increments. It could be argued that saccade detection had no improvement at all, while smooth pursuits had the highest scores with a 7-scale size instead of the largest 8-scale one. Based on our experiments, it is still not clear whether the network capacity was not appropriate for a larger set of input scales using the same hyperparameters, or whether these inputs naturally have a detrimental effect on training when reaching a certain length.

A similar phenomenon was observed when examining the effect of larger context windows. Although training with both 1.5-second and 2-second temporal windows resulted in improved classification performance, the evaluation on the 1.5-second context presented the general highest scores. It is clear that smooth pursuits had an increased detection rate with larger contexts, but these results do not support the hypothesis that smooth pursuit classification scores increase proportionally with window size.

The filter heuristics also demonstrated to be fundamental when it comes to event detection with TCNs. This improvement could be explained by the fact that convolutional architectures are not favored by the sequential learning constraints of recurrent nets, thus often tainting an event block with one or other wrong predictions. This, of course, can be harmful depending on the metric chosen for event assessment, but it barely affects sample-level classification accuracy.

Compared to other architectures, the TCN is also clearly advantageous in terms of memory usage and training time, since TCNs do not make use of cell gates and have a convenient structure for parallel processing on GPUs. Contrary to LSTMs or GRUs, it could also learn from an arbitrary input length, but our current findings do not suggest that this would not necessarily result in a superior classification performance.

Another consideration is that our TCN model was not designed for online classification, but this would be possible by simply reverting the current configuration to *causal* convolutions. This would obviously result in more modest scores, since the network effectively ceases to be bidirectional. However, because evaluation on TCNs is potentially faster than on recurrent models with the similar memory requirements, TCNs could be a more appropriate choice for real-time applications with high frame rates.

7 Conclusion

In this work we designed a temporal convolutional network for the problem of automated eye movement classification. We demonstrated that our model was capable of surpassing the previous state-of-the-art architectures on the same dataset using the same metrics. Moreover, we highlighted several additional advantages of TCNs over preceding recurrent models, such as lower memory footprint, faster training, and the ability of learning sequences of arbitrary length.

Additionally, the research on feature space, feature scale size, and context window width revealed that such extrinsic factors can also play an important role on model performance, marginally improving predictions in all classes, though these improvements were not consistent with factor size increments. Overall, we witnessed the highest classification gain with smooth pursuits, which are the most underrepresented class in the GazeCom dataset.

References

1. Agtzidis, I., Startsev, M., Dorr, M.: In the pursuit of (ground) truth: a hand-labelling tool for eye movements recorded during dynamic scene viewing. In: 2016 IEEE Second Workshop on Eye Tracking and Visualization (ETVIS). pp. 65–68 (2016)
2. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR **abs/1803.01271** (2018), http://arxiv.org/abs/1803.01271
3. Berg, D.J., Boehnke, S.E., Marino, R.A., Munoz, D.P., Itti, L.: Free viewing of dynamic stimuli by humans and monkeys. Journal of Vision **9**(5), 19–19 (05 2009). https://doi.org/10.1167/9.5.19
4. Berndt, S., Kirkpatrick, D., Taviano, T., Komogortsev, O.: Tertiary eye movement classification by a hybrid algorithm. CoRR **abs/1904.10085** (2019), http://arxiv.org/abs/1904.10085
5. Campbell, C.S., Maglio, P.P.: A robust algorithm for reading detection. In: Proceedings of the 2001 workshop on Perceptive user interfaces. pp. 1–7 (2001)
6. Cassin, B., Rubin, M.L., Solomon, S.: Dictionary of eye terminology, vol. 10. Triad Publishing Company Gainsville (1984)

7. Dorr, M., Martinetz, T., Gegenfurtner, K.R., Barth, E.: Variability of eye movements when viewing dynamic natural scenes. Journal of Vision **10**(10), 28–28 (08 2010). https://doi.org/10.1167/10.10.28
8. Duchowski, A.T.: Gaze-based interaction: A 30 year retrospective. Comput. Graph. **73**, 59–69 (2018). https://doi.org/10.1016/j.cag.2018.04.002
9. Fuhl, W.: Fully convolutional neural networks for raw eye tracking data segmentation, generation, and reconstruction (2020)
10. Goodfellow, I.J., Bengio, Y., Courville, A.C.: Deep Learning. Adaptive computation and machine learning, MIT Press (2016), http://www.deeplearningbook.org/
11. Hooge, I., Niehorster, D., Nyström, M., Andersson, R., Hessels, R.: Is human classification by experienced untrained observers a gold standard in fixation detection? Behavior Research Methods **50**(5), 1864–1881 (1 2018). https://doi.org/10.3758/s13428-017-0955-x, copyright 2017. Published by Elsevier Ltd.
12. Hoppe, S., Bulling, A.: End-to-end eye movement detection using convolutional neural networks. CoRR **abs/1609.02452** (2016), http://arxiv.org/abs/1609.02452
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
14. Komogortsev, O., Gobert, D., Jayarathna, S., Koh, D., Gowda, S.: Standardization of automated analyses of oculomotor fixation and saccadic behaviors. Biomedical Engineering, IEEE Transactions on **57**, 2635 – 2645 (12 2010). https://doi.org/10.1109/TBME.2010.2057429
15. Komogortsev, O., Karpov, A.: Automated classification and scoring of smooth pursuit eye movements in the presence of fixations and saccades. Behavior research methods **45** (07 2012). https://doi.org/10.3758/s13428-012-0234-9
16. Komogortsev, O.V., Karpov, A.: Automated classification and scoring of smooth pursuit eye movements in the presence of fixations and saccades. Behavior Research Methods **45**(1), 203–215 (Mar 2013). https://doi.org/10.3758/s13428-012-0234-9, https://doi.org/10.3758/s13428-012-0234-9
17. Komogortsev, O.V., Khan, J.I.: Kalman filtering in the design of eye-gaze-guided computer interfaces. In: Proceedings of the 12th International Conference on Human-computer Interaction: Intelligent Multimodal Interaction Environments. pp. 679–689. HCI'07, Springer-Verlag, Berlin, Heidelberg (2007), http://dl.acm.org/citation.cfm?id=1769590.1769667
18. Larsson, L., Nyström, M., Andersson, R., Stridh, M.: Detection of fixations and smooth pursuit movements in high-speed eye-tracking data. Biomedical Signal Processing and Control **18**, 145 – 152 (2015). https://doi.org/10.1016/j.bspc.2014.12.008, http://www.sciencedirect.com/science/article/pii/S1746809414002031
19. Leigh, R.J., Zee, D.S.: The neurology of eye movements. OUP USA (2015)
20. Nyström, M., Holmqvist, K.: An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. Behavior Research Methods **42**(1), 188–204 (Feb 2010). https://doi.org/10.3758/BRM.42.1.188, https://doi.org/10.3758/BRM.42.1.188
21. Peters, C.E., Pelachaud, C., Bevacqua, E., Mancini, M., Poggi, I.: A model of attention and interest using gaze behavior. In: Panayiotopoulos, T., Gratch, J., Aylett, R., Ballin, D., Olivier, P., Rist, T. (eds.) Intelligent Virtual Agents, 5th International Working Conference, IVA 2005, Kos, Greece,

September 12-14, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3661, pp. 229–240. Springer (2005). https://doi.org/10.1007/11550617_20, https://doi.org/10.1007/11550617_20

22. Salvucci, D.D., Goldberg, J.H.: Identifying fixations and saccades in eye-tracking protocols. In: Proceedings of the 2000 Symposium on Eye Tracking Research & Applications. pp. 71–78. ETRA '00, ACM, New York, NY, USA (2000). https://doi.org/10.1145/355017.355028, http://doi.acm.org/10.1145/355017.355028
23. Santini, T., Fuhl, W., Kübler, T., Kasneci, E.: Bayesian identification of fixations, saccades, and smooth pursuits. In: Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications. pp. 163–170. ETRA '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2857491.2857512, http://doi.acm.org/10.1145/2857491.2857512
24. Sauter, D., Martin, B.J., Di Renzo, N., Vomscheid, C.: Analysis of eye tracking movements using innovations generated by a kalman filter. Medical and Biological Engineering and Computing **29**(1), 63–69 (Jan 1991). https://doi.org/10.1007/BF02446297, https://doi.org/10.1007/BF02446297
25. Shepherd, S.: Following gaze: Gaze-following behavior as a window into social cognition. Frontiers in Integrative Neuroscience **4**, 5 (2010). https://doi.org/10.3389/fnint.2010.00005, https://www.frontiersin.org/articles/10.3389/fnint.2010.00005
26. Startsev, M., Agtzidis, I., Dorr, M.: 1d cnn with blstm for automated classification of fixations, saccades, and smooth pursuits. Behavior Research Methods **51**(2), 556–572 (Apr 2019). https://doi.org/10.3758/s13428-018-1144-2, https://doi.org/10.3758/s13428-018-1144-2
27. Startsev, M., Agtzidis, I., Dorr, M.: Sequence-to-sequence deep learning for eye movement classification. In: Perception. vol. 48, pp. 200–200. Sage Publications LTD 1 Olivers Yard, 55 City Road, London EC1Y 1SP, England (2019)
28. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. pp. 3104–3112 (2014), http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks
29. Vidal, M., Bulling, A., Gellersen, H.: Detection of smooth pursuits using eye movement shape features. In: Proceedings of the Symposium on Eye Tracking Research and Applications. pp. 177–180. ETRA '12, ACM, New York, NY, USA (2012). https://doi.org/10.1145/2168556.2168586, http://doi.acm.org/10.1145/2168556.2168586
30. Zemblys, R., Niehorster, D.C., Holmqvist, K.: gazenet: End-to-end eye-movement event detection with deep neural networks. Behavior research methods (2018)
31. Zemblys, R., Niehorster, D.C., Komogortsev, O., Holmqvist, K.: Using machine learning to detect events in eye-tracking data. Behavior research methods **50**(1), 160–181 (2018)