Update for gazeNet

1st Lorenz Falcioni
Fakultät Elektro- und Informationstechnik
Ostbayerische Technische Hochschule Regensburg
Regensburg, Germany
lorenzfalcioni@gmail.com

2nd Timur Ezer
Software Engineering Laboratory for Safe and Secure Systems (LaS³)
Ostbayerische Technische Hochschule Regensburg
Regensburg, Germany
timur.ezer@oth-regensburg.de

3rd Jürgen Mottok
Software Engineering Laboratory for Safe and Secure Systems (LaS³)
Ostbayerische Technische Hochschule Regensburg
Regensburg, Germany
juergen.mottok@oth-regensburg.de

I. Abstract

The application of deep learning techniques in the field of eye-tracking data analysis has brought forth gazeNet, a "framework for creating event detectors that do not require hand-crafted signal features or signal thresholding". Zemblys et al. [2018] Despite the authors presenting the algorithm as a "proof of concept" Zemblys et al. [2018], the pretrained model spawned in the process is kindly provided by the authors free to use. The primary objective of this work is streamlining the usage of this model. Modifications of the original code have been made to meet the requirements both of the used programming language i.e. Python2 to Python3 as well as the used packages. The existing code has also been modified to be interpretable on Windows-machines. A conversion script for eye-tracking data serves as a complement to the original framework facilitating the application in future research.

II. Introduction

A core aspect of eye-tracking research is *event detection*, meaning the classification of sections of eye-tracking data into different types of eye movements such as fixations and saccades. For the longest time eye-tracking data analysis has relied mainly on "traditional" algorithms, which are based on hand-crafted features and thresholds. Zemblys et al. [2018]

Deep learning in the field of eye-tracking event detection has only recently appeared in the works of Hoppe and Bulling as means of training a neural network. While Hoppe and Bulling "use a Fast Fourier transform to extract a number of frequency components (features) first and then use these as input to their algorithm" Hoppe and Bulling [2016], Zemblys et al. [2018] have taken this approach one step further by creating an entire *framework* for the creation of event detectors that do not require hand-crafted signal features or signal thresholding.

Zemblys et al. [2018] created *gazeNet* "as a proof of concept [...] not to be understood as an off-the-shelf algorithm that can be employed instantly with no preparation and no understanding of how it works." Zemblys, Niehorster, and Holmqvist [2018] In fact the use of gazeNet can be cumbersome, because it lacks means of analyzing eye-tracking data from other sources than the one used by the authors. Furthermore the code is written in Python 2.7, which is no longer supported by the Python Software Foundation.

This work aims to take the first step towards a more user-friendly interface for the *application* of gazeNet. This is done by providing a template for a script, which converts eye-tracking-data provided by the user in the appropriate format for gazeNet and modifying the partly outdated code.

III. Update to gazeNet 0.1

A. *gazeNet*

Zemblys et al. [2018] have created gazeNet, a python framework for eye-movement event detection with deep neural networks. At the core of the presented framework lies a convolutional neural network (CNN) which can be trained to detect eye-movement events. Its architecture is inspired by Deep Speech 2, an end-to-end speech recognition neural network (Amodei et al. [2015], Zemblys et al. [2018]). Raw eye-tracking data in the form of a series of samples of x and y coordinates of the eye position is fed into the CNN. The CNN then annotates the data with event labels.

The training process of gazeNet relies heavily on the generation of synthetic training data using gazeGenNet, a recurrent neural network (RNN) with a fully connected layer at the end. The RNN is trained on a small amount of manually labeled *real* eye-tracking data and then used to generate large amounts of synthetic data. The synthetic data in turn is used to train gazeNet itself. This intermediary step is necessary, as the amount of real eye-tracking data is limited and would be insufficient for the training of gazeNet. Zemblys et al. [2018]

Zemblys et al. [2018] used manually annotated by expert coders eye-tracker data from the Humanities Lab, Lund University (hereafter called the *Lund2013* dataset) to prove the working of gazeNet. Parts of the dataset that included events other than fixations, saccades or post saccadic oscillations were discarded, as the scope of the paper was limited to the mentioned event types. From the remaining 20 trials six were used as training data. Thanks to the above described method of data augmentation the authors were able to train gazeNet on data extended by a factor of 450x resulting in 326 minutes of synthetic data.

Thankfully the authors provide the pretrained model stemming from the Lund2013 dataset free to use. Our work is centered around this pretrained model.

B. ETData

The ETData class defines a data type using the np.dtype function from the NumPy library. The data type contains fields for the time stamp, x and y coordinates of the eye position, a status flag indicating whether the data is valid or not, and an event code indicating the type of eye movement (e.g. fixation, saccade, etc.). The evt_color_map dictionary maps event codes to colors that can be used to plot the eye movements.

C. Update to Python 3.7.3

The original gazeNet was developed using Python 2.7 and PyTorch 0.2.0_4. Since then many changes have been made to the Python interpreter as well as the PyTorch framework. Modifications to the original code were necessary to make it compatible with the current versions of Python and PyTorch. The following changes were made to the original code:

- print is now a function; therefore it requires parentheses
- the division operator / now returns a float instead of an integer; the floor division operator // returns an integer
- for loops do not require an iterator variable anymore
- in-place-modifications of ordered dictionaries are not allowed anymore; instead a copy of the dictionary has to be created

Minor changes have been made in order to comply with PyTorch 2.1 regarding datatypes. Other packages used in the original code have not caused any problems. Nevertheless a requirement file has been provided to ensure that the user has the correct versions of the packages installed.

D. Platform Independence

Hardware acceleration can be used by gazeNet to speed up the training process. In this context worker threads are being used to load data in parallel to keep the GPU busy reducing time consumption even further. In order to achieve platform independence the default number of worker threads in data loading has been set to 0. While strictly speaking this change is not a necessity as the number of worker threads can be set via command line, it is worthwhile making, as the old default value of 1 causes problems on Windows machines due to collisions. For inexperienced users this change avoids a potential source of error.

E. Conversion script

In order to allow maximum flexibility for future users of gazeNet a script is provided which converts .csv/.tsv eye-tracking data files into the proprietary datatype ETData used by gazeNet. The script works on both Windows and UNIX-like operating systems thanks to pathlib which handles path and file names. Even though slight modification may be necessary, the given script should provide a useful template for the preparation of similarly formatted gaze recordings.

In this particular case the file to be converted stems from an eye tracker made by Tobii. Each sample of the recording is stored in a separate row. The gaze coordinates are stored in the columns ”Gaze point X” and ”Gaze point Y” in cartesian form as pixels and can therefore be used as they are. The custom datatype requires information about the geometry of the test setup in order to internally convert the cartesian coordinates to polar coordinates. As of now these parameters are hardcoded in the script. Finally invalid samples are filtered.

F. Validation of gazeNet 0.1

In Zemblys et al. [2018] the authors rely heavily on the lund2013-image-test dataset in the development of their framework, which makes it a good choice for validation of the updated gazeNet. Figure III-D shows the output of gazeNet 0.1 on the lund2013-image-test dataset. The results are comparable to the results presented in Zemblys et al. [2018].

The working of the conversion script has been shown on a recording of a participant looking at randomly appearing crosses on a screen. Figure III-F shows the output of gazeNet 0.1 on the recording. The results are comparable to the results presented in Zemblys et al. [2018].

IV. Future work

Even though the update to gazeNet 0.1 has increased useability there is still a lot of work that can be done in order to make this tool even more accessible. Especially considering the framework, as it is now, still only being a proof-of-concept already being so valuable, the potential cannot be underestimated. Areas of further development should either improve gazeNet in functionality or in ease of use.

A. Functionality

At the time of writing the pretrained neural network can only label fixations, saccades and post saccadic oscillations. Logically it would make sense to expand the types of events such as continuos pursuit that can be detected. One way to realize this would be to train a new network including data containing also said event types. The challenge hereby, apart from “obtaining reliably coded eye-tracking data” Zemblys et al. [2018], lies in training an algorithm, that “can perform as well as human expert coders” Zemblys et al. [2018]. It has yet to be seen if adding even only one new event type to be recognized has a negative impact on accuracy. If this should be the case, a possible solution might be to change the architecture of the neural network.

B. Ease of use

The addition of the conversion script has already opened up the possibility of using gazeNet with data from other eye trackers than the one used by the authors. However, at this point it is still necessary to clone the repository and work with the code more or less directly i.e. installing dependencies. A more user-friendly approach would be to provide a package

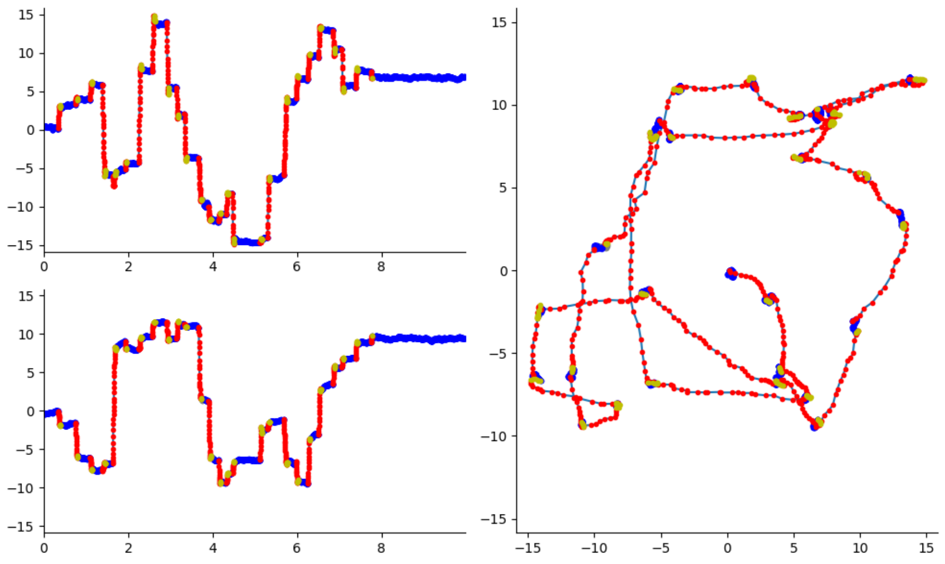

Fig. 1. The two graphs on the left show the x and y coordinates of the gaze position in degrees over time in seconds. The graph on the right resembles the gaze position in the x-y-plane. The red dots represent fixations, the blue dots represent saccades and the green dots represent post saccadic oscillations.

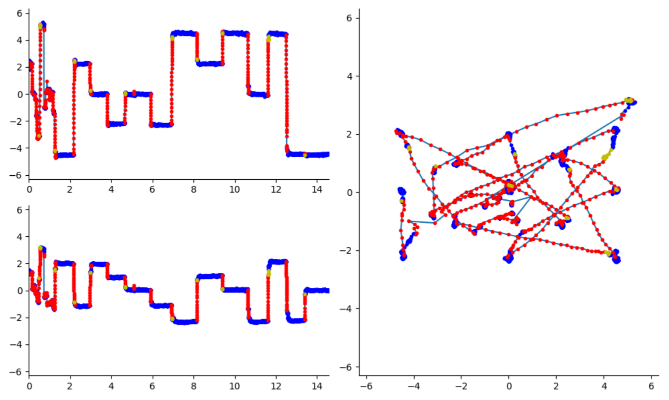

Fig. 2. The two graphs on the left show the x and y coordinates of the gaze position in degrees over time in seconds. The graph on the right resembles the gaze position in the x-y-plane. The red dots represent fixations, the blue dots represent saccades and the green dots represent post saccadic oscillations. It should be noted that the first two seconds of the recording contain the calibration points of the eye tracker. The calibration process contains events on which gazeNet is not trained. Therefore the labelling is insignificant.

that can be installed via pip. The user would then only have to import the package and call the appropriate functions.

Taking this thought even further, the user could be provided with a graphical user interface (GUI) to interact with gazeNet. This would allow the user to work with gazeNet without having to write a single line of code. The GUI could be implemented as a jupyter notebook, which would allow the user to work with gazeNet on any device with a web browser. The user would then only have to upload the data to be analyzed, define the geometry of the eye-tracking setup and download the results.

References

Dario Amodei, Rishita Anubhai, and Eric Battenberg. Deep speech 2: End-to-end speech recognition in english and mandarin. *Proceedings of Machine Learning Research*, 2015.

Sabrina Hoppe and Andreas Bulling. End-to-end eye movement detection using convolutional neural networks. *ArXiv*, abs/1609.02452, 2016. URL https://api.semanticscholar.org/CorpusID:5938282.

Raimondas Zemblys, Diederick C Niehorster, and Kenneth Holmqvist. gazenet: End-to-end eye-movement event detection with deep neural networks. *Behavior research methods*, 2018.