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

Deep learning in the field of eye tracking event detection has recently appeared as an advanced use of machine learning. The work of Zemblys, Niehorster, and Holmqvist [2018] "presents the algorithm as a proof of concept, and is 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] Moreover the use of gazeNet can be cumbersome, as it requires knowledge about the data structure being used.

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which converts eye-tracking-data provided by the user in the appropriate format for gazeNet and modifying the partly outdated code.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 1.1

A. gazeNet

The original gazeNet architecture by Zemblys et al. [2018] was inspired by Deep Speech 2 Amodei et al., an end-to-end speech recognition neural network. gazeNet was implemented using the pyTorch6 neural network framework (version 0.2.0 4) and the starter code from Sean Naren.7 Our network has two convolutional layers followed by three bidirectional recurrent layers with a fully connected layer on top. The convolutional layers use 2D filters with a size of 2 x 11 and are meant to extract deep features from raw input data, while the recurrent layers model event sequences and are responsible for detecting onsets and offsets of fixations, saccades and PSOs. Zemblys et al. [2018]

The presented framework gazeNet consists of various scripts written in python which allow the user to prepare raw data, augment data, train gazeNet and evaluate raw data.

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.

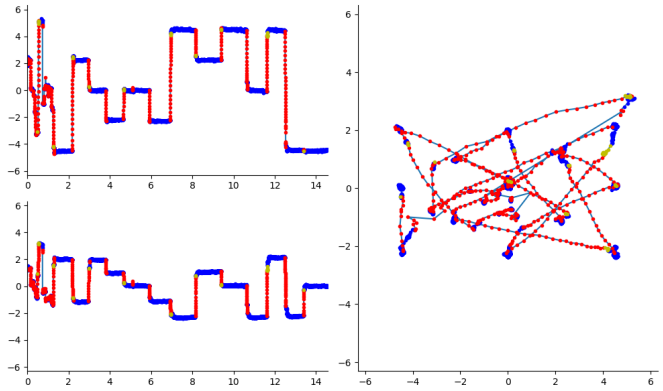

Fig. 1. The two graphs on the left show the x and y coordinates of the gaze position in degrees over time in seconds. The graph on the right resembles the gaze position in the x-y-plane. The red dots represent fixations, the blue dots represent saccades and the green dots represent post saccadic oscillations.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

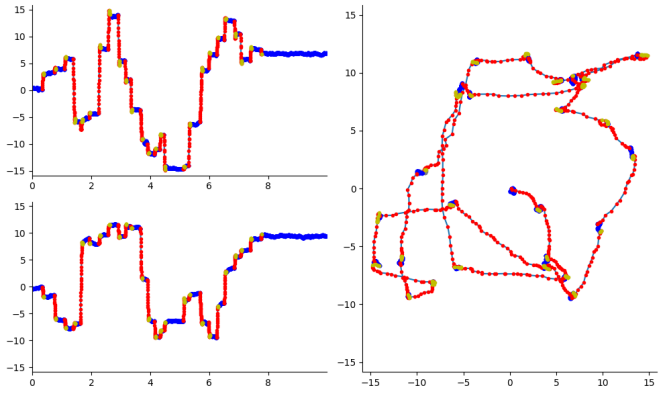

Fig. 2. The two graphs on the left show the x and y coordinates of the gaze position in degrees over time in seconds. The graph on the right resembles the gaze position in the x-y-plane. The red dots represent fixations, the blue dots represent saccades and the green dots represent post saccadic oscillations.

F. Validation of gazeNet 1.1

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.

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