# 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

gazeNet is a new framework for creating event detectors. The primary objective of this work is the update of the existing gazeNet by Zemblys et al. [2018] to current standards. In modifications this includes both in 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.

Furthermore an interface is provided to facilitate the application of the pretrained gazeNet. This is done by providiong a template for a script, which converts eye-tracking-data provided by the user in the appropriate format for gazeNet.

## II. Introduction

The use of neural networks presents a new way of event detection in eye-tracking-data. Zemblys et al. [2018] presents gazeNet as one of the first approaches in the application of the technology. As much as the work of Zemblys, Niehorster, and Holmqvist [2018] represents a milestone in the evaluation of eye-tracking-data, since usability was not a priority, the use of gazeNet can be cumbersome, as it requires operation via commandline and knowledge about the data structure being used.

## III. 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 onvolutional 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.

### *A. ETData*

### *B. Update to gazeNet 1.1*

### *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.

### *D. Platform Independence*

### *E. Validation of gazeNet 1.1*

## IV. 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 usefull 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

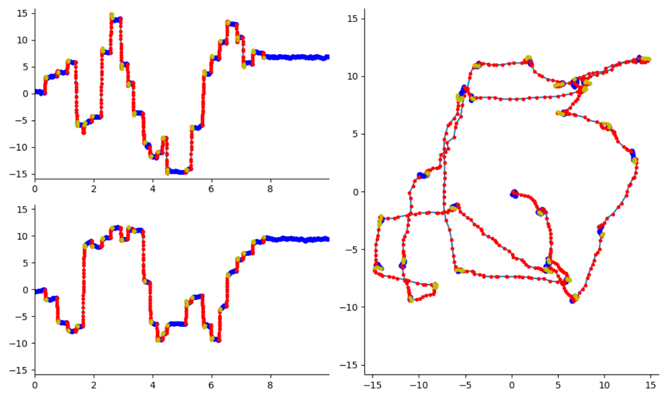

Fig. 1. The two graphs on the left show the x and y coordinates of the gaze position in degrees over time in seconds. The graph on the right resembles the gaze position in the x-y-plane. The red dots represent fixations, the blue dots represent saccades and the green dots represent post saccadic oscillations.

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.

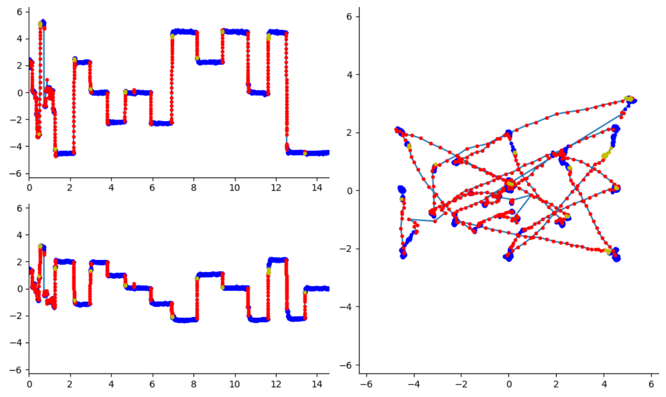

Fig. 2. The two graphs on the left show the x and y coordinates of the gaze position in degrees over time in seconds. The graph on the right resembles the gaze position in the x-y-plane. The red dots represent fixations, the blue dots represent saccades and the green dots represent post saccadic oscillations.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

## 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.