OpenNFT: An open-source Python/Matlab framework for real-time fMRI neurofeedback training based on activity, connectivity and multivariate pattern analysis

Yury Koush, John Ashburner, Evgeny Prilepin, Ronald Sladky, Peter Zeidman, Sergei Bibikov, Frank Scharnowski, Artem Nikonorov, Dimitri Van De Ville

PII: S1053-8119(17)30505-0
DOI: http://dx.doi.org/10.1016/j.neuroimage.2017.06.039
Reference: YNIMG14121

To appear in: *NeuroImage*

Received date: 11 May 2017
Accepted date: 16 June 2017

Cite this article as: Yury Koush, John Ashburner, Evgeny Prilepin, Ronald Sladky, Peter Zeidman, Sergei Bibikov, Frank Scharnowski, Artem Nikonorov and Dimitri Van De Ville, OpenNFT: An open-source Python/Matlab framework for real-time fMRI neurofeedback training based on activity, connectivity and multivariate pattern analysis, *NeuroImage* http://dx.doi.org/10.1016/j.neuroimage.2017.06.039

OpenNFT: An open-source Python/Matlab framework for real-time fMRI neurofeedback training based on activity, connectivity and multivariate pattern analysis

Yury Koush[1,2,3*], John Ashburner[4], Evgeny Prilepin[5], Ronald Sladky[6,7,8], Peter Zeidman[4], Sergei Bibikov[9], Frank Scharnowski[6,7,8], Artem Nikonorov[5,9], Dimitri Van De Ville[2,3]

[1] Department of Radiology and Medical Imaging, Yale University, New Haven, USA

[2] Institute of Bioengineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), Campus Biotech, Geneva, Switzerland

[3] Department of Radiology and Medical Informatics, University of Geneva, Geneva, Switzerland

[4] Wellcome Trust Centre for Neuroimaging, University College London, London, UK

[5] Aligned Research Group, 20 S Santa Cruz Ave 300, 95030 Los Gatos, California, USA

[6] Department of Psychiatric, Psychotherapy and Psychosomatics, Psychiatric Hospital, University of Zürich, Lenggstrasse 31, 8032 Zürich, Switzerland

[7] Neuroscience Center Zürich, University of Zürich and Swiss Federal Institute of Technology, Winterthurerstr. 190, 8057 Zürich, Switzerland

[8] Zürich Center for Integrative Human Physiology (ZIHP), University of Zürich, Winterthurerstr. 190, 8057 Zürich, Switzerland

[9]Supercomputers and Computer Science Department, Samara University, Moskovskoe shosse str., 34, 443086 Samara, Russia

yury.koush@yale.edu

yurykoush@gmail.com

opennft@gmail.com

[*]**Corresponding Author:** Department of Radiology and Medical Imaging, Yale University, New Haven, USA

Abstract

Neurofeedback based on real-time functional magnetic resonance imaging (rt-fMRI) is a novel and rapidly developing research field. It allows for training of voluntary control over localized brain activity and connectivity and has demonstrated promising clinical applications. Because of the rapid technical developments of MRI techniques and the availability of high-performance computing, new methodological advances in rt-fMRI neurofeedback become

possible. Here we outline the core components of a novel open-source neurofeedback framework, termed *Open NeuroFeedback Training* (*OpenNFT*), which efficiently integrates these new developments. This framework is implemented using Python and Matlab source code to allow for diverse functionality, high modularity, and rapid extendibility of the software depending on the user's needs. In addition, it provides an easy interface to the functionality of Statistical Parametric Mapping (SPM) that is also open-source and one of the most widely used fMRI data analysis software. We demonstrate the functionality of our new framework by describing case studies that include neurofeedback protocols based on brain activity levels, effective connectivity models, and pattern classification approaches. This open-source initiative provides a suitable framework to actively engage in the development of novel neurofeedback approaches, so that local methodological developments can be easily made accessible to a wider range of users.

Keywords: OpenNFT, neurofeedback, real-time fMRI, activity, connectivity, multivariate pattern analysis

Introduction

About two decades ago, real-time functional magnetic resonance imaging (rt-fMRI) was introduced (Cox et al., 1995) and turned into a rapidly developing discipline. Neurofeedback based on rt-fMRI provides an interactive protocol to let participants learn to voluntarily control their brain activity and connectivity (Caria et al., 2012; deCharms, 2008; Sitaram et al., 2017; Stoeckel et al., 2014; Sulzer et al., 2013; Weiskopf, 2012). Conventional measures of brain activity for fMRI-based neurofeedback continuously track the blood oxygen level dependent (BOLD) signal from a target region of interest (ROI). Most often, gradient-echo echo-planar imaging (GE-EPI) T2∗-sensitive protocols and their extensions are used to acquire brain volume data in real-time (Weiskopf, 2012; Weiskopf et al., 2007b). Each GE-EPI voxel value is proportional to the BOLD signal, which is an indirect measure of underlying neural activity (Logothetis et al., 2001). The neurofeedback loop is closed when brain activity induced by the participant's efforts for regulation, is presented as the neurofeedback signal (LaConte, 2011; Sitaram et al., 2017). With the help of neurofeedback, participants can learn voluntary control over the own brain activity. Such neurofeedback training has been shown to cause behavioral

consequences, thus providing a scientific tool for investigating the relationship between brain function and behavior (Sitaram et al., 2017; Sulzer et al., 2013). Likewise, neurofeedback allows neurological and psychiatric patients to normalize abnormal levels of brain activity that are associated with their disorder. It thus holds great promises as a drug-free and non-invasive experimental therapy that has been shown to be effective in depression, addiction, stroke, chronic pain, Parkinson's disease, and tinnitus (Haller et al., 2010; Hartwell et al., 2013; Li et al., 2013; Liew et al., 2016; Linden et al., 2012; Subramanian et al., 2011; Young et al., 2014) .

The conventional neurofeedback study involves defining the physiological target in terms of the brain ROI or network to be trained, assessing the participant's performance in terms of modulating brain activity, as well as the learning progression and the ability to transfer the trained regulation to an experimental situation without feedback. These experiments are challenging in terms of the evaluation of behavioral or therapeutic effects before and after learning (Sulzer et al., 2013). Therefore, these studies can consist of multiple runs accompanying the neurofeedback ones, such as pre- and post-training tests and transfer runs. Neurofeedback studies range from short single-day neurofeedback session (deCharms et al., 2005) to relatively long experiments with up to 10 sessions spanned over several days (Shibata et al., 2011). Neurofeedback training sessions usually last approximately 1 hour and consist of a few neurofeedback runs (Stoeckel et al., 2014; Sulzer et al., 2013). The runs are usually composed of alternating 10-30 s regulation and baseline blocks and last around 5-20 min each.

On the one hand, neurofeedback based on rt-fMRI is a rapidly developing and technically highly demanding approach. On the other hand, technical and methodological developments of the MRI hardware and software advance quickly, which, together with increasing availability of high-performance computing power, allows for more sophisticated real-time brain data processing approaches. The currently available software for rt-fMRI neurofeedback include the commercially available Turbo-BrainVoyager software (http://www.brainvoyager.com), and a few non-commercial software packages, such as FRIEND (https://www.nitrc.org/projects/friend/), which is written in C++ and based on the FSL libraries (https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/), the AFNI real-time plug-in written in C (Cox et al., 1995) (https://afni.nimh.nih.gov/), scanSTAT written in C/C++ (Cohen, 2001) (http://www.brainmapping.org/scanSTAT/), BART written in C/C++ (Hellrung et al., 2015), and the FieldTrip extension for rt-fMRI written in Matlab (http://www.fieldtriptoolbox.org/). All of

these software packages except for FieldTrip (which is non-GUI-based) are written in non-interpreted programming languages that require more sophisticated programming skills compared to interpreted languages. Thus, there is need for an open-source GUI-based multi-processing integrated framework written in more easy interpreted programming languages such as Python and Matlab, with a broad functionality asset for neurofeedback studies. However, despite the relative simplicity of programming in Matlab as compared to C/C++ or Java, it's GUI capabilities, parallel architecture implementation, and concurrent data processing performance is weaker. One solution to this problem is combining Matlab and Python, which allows for preserving the relative simplicity of programming in Matlab as well as an integration with other widely-used Matlab-based neuroimaging toolboxes, and for benefitting from the advanced GUI capabilities, parallel architecture and concurrent data processing of Python.

In this technical note, we provide a brief overview of the core neurofeedback data processing steps required to perform activity-, connectivity- and classification-based neurofeedback studies, and introduce an open-source framework, termed *OpenNFT*. This framework represents a parallel architecture and the set of core modules required to quickly design and test new neurofeedback experiments. The *OpenNFT* is written in an integrated Python/Matlab environment to facilitate concurrent functionality, high modularity, and the ability to extend the software in Python or Matlab into new directions depending on programming preferences, research questions, and clinical application. To demonstrate its capabilities, we describe three case studies covering neurofeedback based on coactivation patterns, advanced effective connectivity, and classification methods.

Methods

In this section, we provide a brief overview of all the methods used to support *OpenNFT* functionality. An interested reader is encouraged to check original publications for more details and outlined methods validation.

Neurofeedback data acquisition and transfer

The data flow of typical neurofeedback experiments consists of several major components including data acquisition, data transfer, data preprocessing, data processing, feedback

estimation and feedback presentation, which creates the core of the *OpenNFT* functionality (Fig. 1).

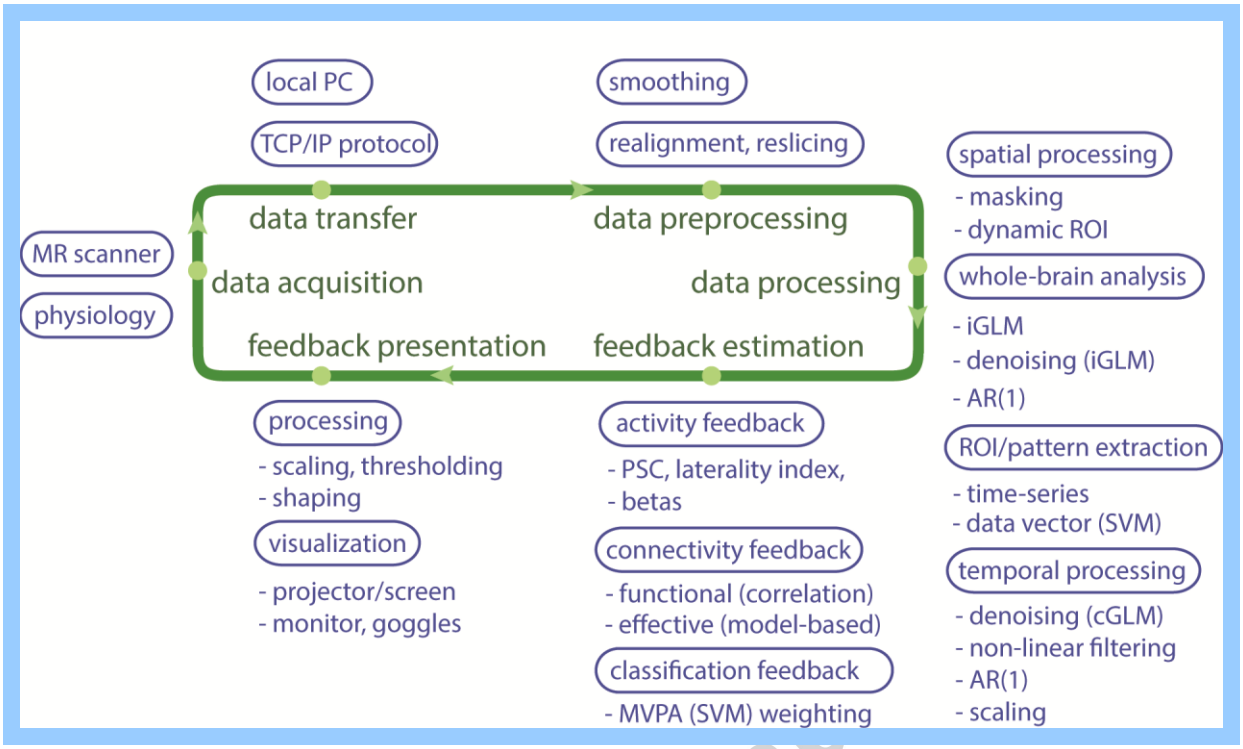

Fig. 1. An illustrative data flow for neurofeedback based on rt-fMRI that reflects the *OpenNFT* built-in functionality. MR – magnetic resonance, PC – personal computer, TCP/IP – transmission control protocol/ internet protocol, ROI – region of interest, PSC – percent signal change, GLM – general linear model, iGLM – incremental GLM, cGLM – cumulative GLM, AR(1) – autoregressive model of the first order, MVPA – multivariate pattern analysis, SVM – support vector machines.

Rt-fMRI data is acquired using standard MR scanners with parameters that are optimized for the specific experimental objectives. Rapid data acquisition, image reconstruction and export from the MR scanner, and availability of high-end computing power have made it possible to acquire and process a large amount of fMRI data in real-time. However, compared to most conventional fMRI experiments where analyses are based on data from the entire experiment (i.e., the complete time course) and where one can average over participants, rt-fMRI analyses are based on a small subset of the fMRI data and on a single participant level. They therefore require that MR acquisition parameters, such as spatial resolution, repetition time (TR) and echo-time (TE), are chosen in order to carefully balance the tradeoff between signal-to-noise ratio and BOLD sensitivity (Weiskopf, 2012). For example, relatively large in-plane voxel sizes in the range of 2-4 mm and matrix sizes of up to 120×120 voxels with 20-36 slices to cover the whole

brain (i.e., slice thickness 2-3 mm and slice gap 20-25% of the slice thickness) have been used in rt-fMRI in different neurofeedback studies at 3T (Stoeckel et al., 2014; Sulzer et al., 2013). The TR in rt-fMRI studies is usually between 2-3 s to cover the whole brain at 3T (Weiskopf, 2012; Weiskopf et al., 2007a), whereas a TR of 1 s been used to target specific brain slabs at 3T and higher magnetic fields (Koush et al., 2014; Koush et al., 2015; Koush et al., 2013b). The adaptation of recently developed parallel and multiband acquisition techniques for rt-fMRI purposes could substantially reduce the acquisition time without a significant loss of data quality, but puts higher demands on the computational power for image reconstruction and subsequent rt-fMRI processing (Todd et al., 2016).

To export and process the imaging data, a data analysis PC is added to the network that contains the manufacturer's MR scanner and reconstruction console. The acquired and reconstructed data are then exported by the MR hardware and software to a shared destination folder that can be assessed by the data analysis workstation via TCP/IP (Weiskopf et al., 2007b). In principle, researchers may export data from different acquisition-to-reconstruction data flow levels (Weiskopf et al., 2004a; Weiskopf et al., 2004b). For example, raw spectroscopic data were exported as soon as they were acquired at 3T and 7T MR scanners to provide an individual feedback signal based on the localized T2∗ changes (Koush et al., 2011; Koush et al., 2013a, 2014). Such customized export procedure typically requires an extension of the manufacturer's reconstruction and/or export prescriptions.

Timing and synchronization

With respect to the timing of rt-fMRI studies, there are substantial differences between continuous, intermittent and trial-based neurofeedback presentation, as well as differences between requirements regarding the synchronization of scanner and presentation hardware in rt-fMRI and conventional fMRI experiments. Technically, most fMRI echo-planar imaging (EPI) protocols allow the scanner to output a short (e.g., 1 msec) trigger pulse via a local serial/parallel/USB port, generated at or before the acquisition of the first scan (i.e., at the center of the k-space). Thus in conventional fMRI experiments, the time-scales of stimulus presentation and MR trigger pulse generation are synchronized, which implies a direct synchronization. In conventional neurofeedback studies, the neurofeedback estimation scheme is linked to the rt-fMRI scan arrivals so that the data is processed as soon as they arrive. This implies an indirect synchronization (in contrast to conventional fMRI studies).

In neurofeedback experiments, there are several interdependent time-scales and event onsets to be considered given the acquisition of the *scan(n)*, such as the time when the trigger pulse was generated by the scanner (t0), when the acquired volume was exported and the software starts reading the file (t1), when the volume was written and the software finishes reading the file (t2), when the data was preprocessed (t3) and processed (t4), when the feedback signal was estimated (t5) and, finally, when it was presented to the participant (t6)(Fig. 2).

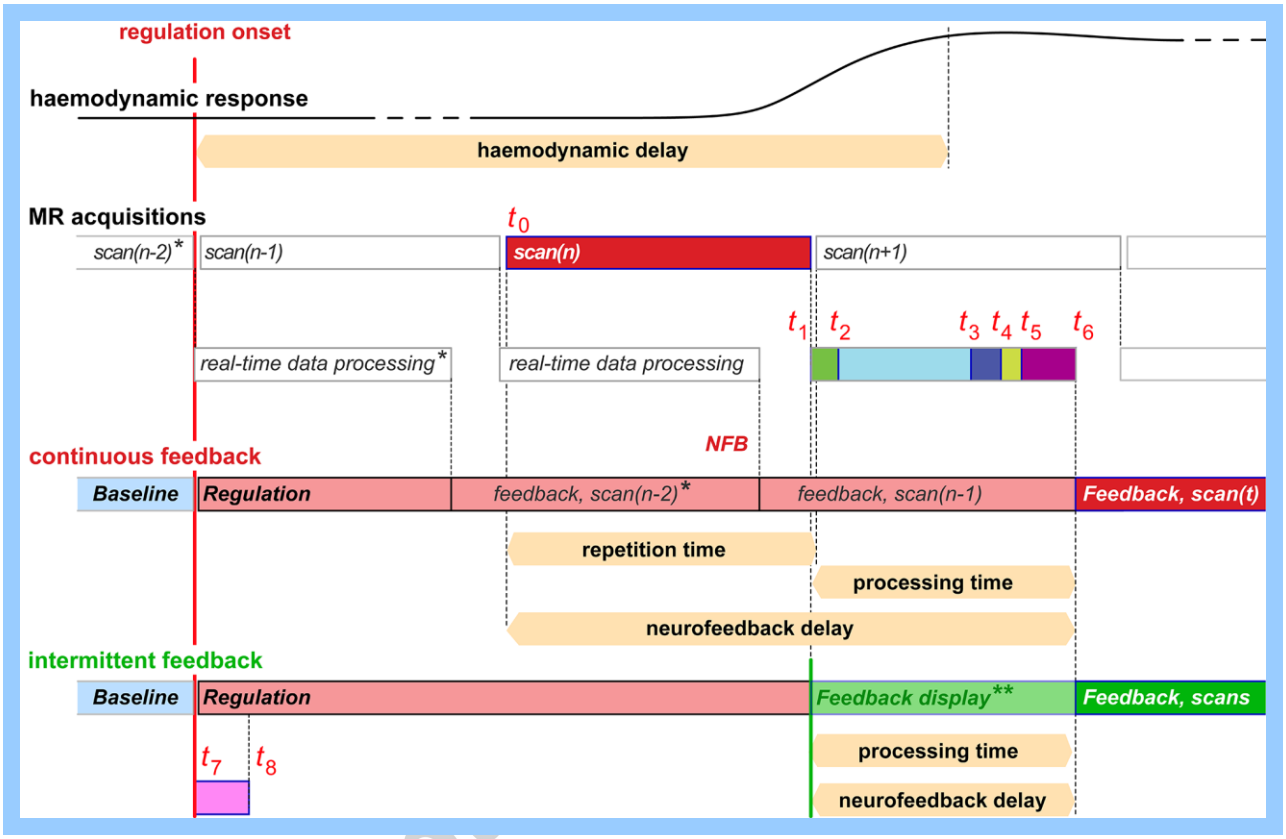

Fig. 2. Timeline of rt-fMRI data flow. The timing of data acquisition and analysis is shown in relation to the *scan(n)* and the BOLD signal delay. Time onsets are illustrative and may vary depending on MR scanner hardware and software properties, scanning parameters, selected data analysis modules and computational performance. Note that the haemodynamic delay of the BOLD signal (blue) is substantially longer (3-5 sec) than that of data acquisition (e.g. 1-2 sec) and that of data processing (e.g. 0.2-1 sec). ∗) denote that feedback from the scans of the baseline condition could be processed and displayed similarly to the regulation condition. ∗∗) denotes the first scan during which an intermittent feedback is required to be present, and that the visual instruction, e.g. a fixation dot, could alternatively appear instead of the regulation instruction to stop the regulation condition until the feedback is processed and displayed.

As soon as a volume of fMRI data is acquired, it is exported and transferred to a local workstation. While the scanner continues to acquire the next volume, the workstation processes

the current volume, estimates the feedback signal, prepares the visualization and finally presents it to the participant. Note that the time when MR scanner generates the trigger pulse for *scan(n+1)* is different from the time when the *scan(n)* first appears in the export directory and the software starts reading it, i.e. from (t1), which depends on MR hardware and software.

For intermittent feedback, the instruction can be displayed prior to the acquisition of the first regulation condition scan, i.e., similarly as for a conventional fMRI study, and time events (t7) and (t8) stand for the beginning and the end of instruction/stimuli display (Fig. 2). If *scan(n+1)* is the first scan during which the intermittent feedback is required to be presented, it is presented exactly the same way as for continuous feedback, i.e. after the processing of the last *scan(n)* of the regulation/task block (Fig. 2, denoted with **)). Also note that (t0) and (t7) will match if instruction/stimuli is displayed based on the trigger pulse event. Finally, intermittent feedback estimation differs from the continuous feedback so that it could be estimated once per regulation block or trial based on the number of the preceding scans in accordance with a specific estimation scheme (see 'Feedback estimation and presentation' sections).

OpenNFT is currently compatible with Siemens and Philips MR scanner exports and can be easily adapted to other MR scanner export formats. The data transfer between scanner hardware and data analysis workstation is achieved using the TCP/IP via a shared folder within the local network to which both are connected. Therefore, trigger pulse, neurofeedback protocol prescriptions, the new data arrivals, and the time when the feedback signal is calculated and is to be presented could be governed by the user and a study-specific estimation scheme. Despite a temporal shift of the first acquired file in continuous feedback to the first trigger pulse, the acquired time-series can be considered in the relative real-time scale of the displayed feedback signal. The offline data processing can compensate for the feedback display shift by accounting for the precisely recorded trigger pulse, or the TR used. In addition, the conventional haemodynamic delay is taken into account when modeling the experimental design for GLM convolving it with the haemodynamic response function (HRF), and when estimating the PSC in real-time and skipping the shifted scans.

Data preprocessing

Once the data has been transferred to the local workstation, data preprocessing commences. The optimal choice of preprocessing steps depends on the planned data processing scheme and the feedback type. For example, spatial realignment and reslicing to compensate for participant

head movements are highly recommended and should only be skipped if special precautions have been taken against movement (e.g., prospective motion correction (Maclaren et al., 2013)).

OpenNFT offers built-in realignment, reslicing, and spatial smoothing algorithms, which consist of standard SPM12 functions (www.fil.ion.ucl.ac.uk/spm) that were adapted for real-time purposes. Specifically, the SPM realignment and reslicing functions were adapted for recursive estimations, while preserving their original functionality. For example, it is possible to choose between different conventional methods, i.e., B-splines of any degree such as linear and cubic ones. The precision level and the number of iterations required to apply spatial realignment is set to the custom SPM settings, but we recommend that these parameters are optimized through a pilot fMRI run to reach the best trade-off between imaging parameters and TR, data complexity, accuracy, and computational needs (for details, see '*OpenNFT* performance' section). The reference EPI used for real-time realignment should have the same dimensions and orientation as those used for the real-time acquisition. This reference EPI, and also the ROIs from which the feedback will be provided, should be prepared offline before the neurofeedback training. The ROI definitions can be based on a functional localizer scan, and on anatomical masks (e.g. atlases, coordinates of other studies, or meta-analyses).

Data processing

In *OpenNFT*, whole-brain activation maps are estimated using incremental GLM (iGLM) statistical analysis (Bagarinao et al., 2003) and displayed in real-time. Incremental GLM algorithm is based on orthogonalization of the explanatory GLM regressors of which the parameter weights can then be estimated more efficiently (Bagarinao et al., 2003). For time-series data processing, cumulative GLM (cGLM) is used to correct for linear drift and head motions, i.e., when the time-series acquired up to the current value is used for a regular GLM estimation. Configurations of the whole-brain iGLM and time-series cGLM allow for creating regressors of the interest and residual forming matrix, as well as estimation of the statistics and contrasts of interest. Note that iGLM and cGLM estimations do not overlap in the current *OpenNFT* configuration unless enabled simultaneously. In the latter case, the extracted time-series will be additionally dependent on iGLM estimations; e.g., in connectivity-based feedback where dynamic ROIs are based on the thresholded iGLMs. For iGLM, residual forming matrix is typically used to remove confounds; e.g., six head motion, linear trend and high-pass filter with 1/128 sec cut-off frequency residuals (as implemented in SPM).

To deal with high-frequency noise and identify non-linear spikes, a configurable extension of a low-pass Kalman filter is applied (Koush et al., 2012). In brief, a low-pass Kalman filter is an adaptive recursive estimation algorithm that allows extraction of the desired signal through a filtering operation with approximated cut-off frequency by adjusting the filter parameters. The extension of the Kalman filter for spike identification is based on introducing the degree of control over a discrepancy between predicted and posteriori signal estimates based on the cumulated standard deviation estimate.

To account for serial correlations in fMRI data due to the aliased non-neurophysiological fluctuations and non-modeled neuronal activity, we used a first-order autoregressive model AR(1) that was implemented by applying recursive filtering to all voxels' time-series. This implementation of AR(1) differs from SPM, where it is performed during residual maximum likelihood estimation given all the available data. In brief, recursive filtering is performed prior to the estimation of the GLM by filtering the observation data vector (i.e., each voxel at time *t*) and the design matrix (i.e., each value at time *t*) with the first-order filter given default SPM AR(1) alpha-value: $\bar{y}_t' = \bar{y}_t - \alpha\bar{y}_{t-1}, \bar{x}_t' = \bar{x}_t - \alpha\bar{x}_{t-1}, \alpha = 0.2$.

In addition, iGLM can be used for different dynamic ROI estimation schemes. For example, dynamically adapting the ROI has been successfully used for ROI-based feedback (Linden et al., 2012), and also for connectivity-based feedback (Koush et al., 2015). For connectivity-based neurofeedback, *OpenNFT* dynamically adapts the ROIs based on the uncorrected statistical threshold (e.g., $p<.01$) applied to the iGLM activation map at the end of each neurofeedback trial. The voxels that survived thresholding within a predefined mask are selected as the new ROI for the next run. If fewer than 10 voxels survive the thresholding, the predefined mask is used as a fallback option. After two neurofeedback trials have been acquired, the ROI that contains most voxels is always selected, i.e., if 42 voxels survived thresholding after trial 1 but only 35 after trial 2, then the dynamic ROI definition based on trial 1 will be used for the third trial. This adaptation could be extended for other feedback estimation schemes. The ROI/mask could be based on pilot studies, or on anatomical definitions (Lancaster et al., 2000), or on both (Koush et al., 2015).

Feedback estimation

OpenNFT offers three different built-in feedback types to demonstrate its functionality: (1) constantly displayed (continuous) and periodically displayed (intermittent) activation-based feedbacks; (2) intermittent effective connectivity feedback; (3) continuous classification-based feedback.

For activation-based feedback, single or multiple ROI activity levels are often estimated in terms of percent signal change (PSC) after preprocessing the data. The PSC is typically computed as a percentage of the average signal during neurofeedback regulation blocks compared to the average during baseline blocks. Although very common, this approach suffers from the relative nature of the fMRI BOLD signal and thus depends on the data quality and baseline definition, i.e., zero-/reference-line identification. Practically, feedback estimation in terms of PSC requires basic volume data processing (extracting voxel activity using ROI/pattern masks), and, subsequently, calculating an average, weighted average or eigenvector estimate from the spatial-temporal data sample within the ROI. This basic processing can be extended to incorporate temporal signal processing of the extracted time-series, and dynamically adapting the size and shape of the ROI based on whole-brain iGLM. Before the PSC feedback is visualized, the time-series is adaptively scaled so that the signal changes are reflected in the displayed feedback signal. To define the dynamic range, *OpenNFT* uses the average of the 5% highest and lowest activity time points of the data acquired so far, i.e. to estimate the maximum and minimum limits of the scaling, respectively. This procedure ensures that the feedback signal stays within the dynamic range of acquired time series. At the same time it prevents rapid (and potentially artifactual) signal changes from causing sudden jumps of the scaling range and, thus, the feedback signal, which can be confusing for the participants (Koush et al., 2012).

Feedback estimates are not limited to single ROI activity in terms of PSC, but can also involve more advanced feedback estimation methods. It has been shown feasible to extract the feedback signal from more than one ROI. For example, feedback signals based on the difference (Robineau et al., 2014; Scharnowski et al., 2015) and the average (Chiew et al., 2012; Paret et al., 2014) between activity levels of two brain areas have been used successfully for neurofeedback training. For activation-based feedback that involves two ROIs, adaptive scaling could be used to combine them. For example, we implemented a scaling procedure that takes the independent dynamic ranges of the time-series into account by first calculating the limits of each ROI, and then scaling them based on the average of their limits (Fig. 3A).

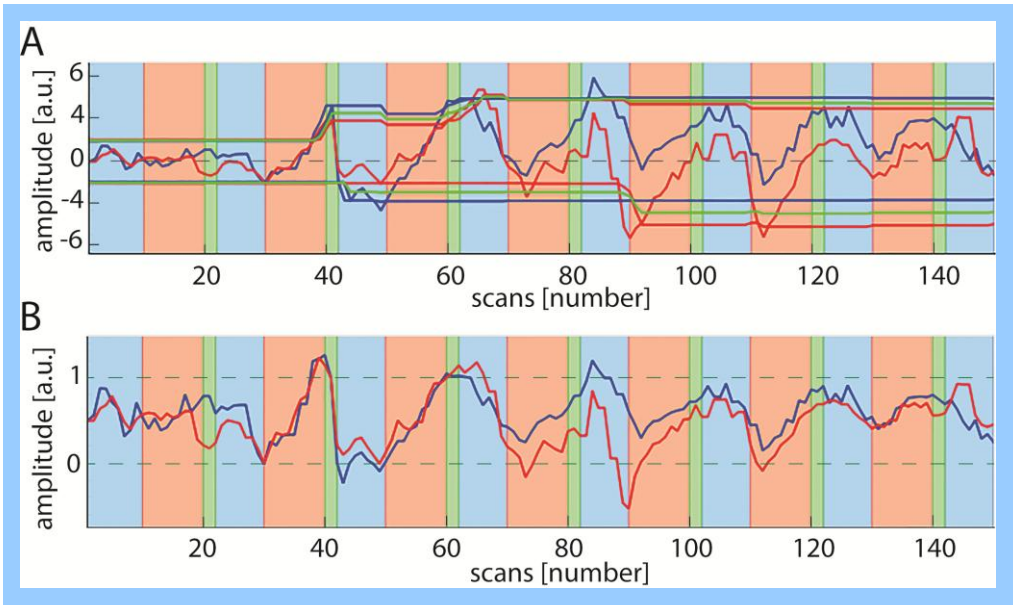

Fig. 3. Demonstration of the scaling procedure for intermittent activity-based neurofeedback. (A) Example time-series from two visual cortex ROIs (red and blue lines) during visual imagery (orange blocks) vs. fixation (blue blocks). Green blocks denote the feedback display. The adaptive scaling for each time series and their average is shown as piece-wise horizontal lines in the respective colors (the average in green). (B) The time-series scaled according to the average dynamic range.

Because of the haemodynamic delay, there might only be a few time points within each block that can be taken into account for calculating PSC. In our intermittent feedback example (Fig. 3, block length is 10 volumes), we therefore used the median of the past six data points of regulation and baseline to calculate the PSCs. The PSCs were estimated separately for each ROI at the end of the regulation block (for more details about the experimental parameters, see Data in Brief).

Several recent attempts have been made to replace real-time activity-based neurofeedback signals by real-time functional-connectivity estimates, such as temporal correlations using Pearson correlation coefficient estimation between two time-series (Kim et al., 2015; Megumi et al., 2015; Zilverstand et al., 2014), and estimation of functional connectivity networks using real-time Smooth Incremental Graphical Lasso (rt-SINGLE, (Monti et al., 2016)). The correlation coefficient could be estimated cumulatively or based on the sliding window, and is easy to extend in *OpenNFT* using available multiple ROI PSC estimation scheme. More computationally demanding estimates of effective connectivity, such as dynamic causal modeling (DCM, (Friston et al., 2003)), have also been successfully used to train participants exerting control over specific brain networks (Koush et al., 2015; Koush et al., 2013b). Neurofeedback training based on

effective connectivity is accomplished in several regulation runs that are each composed of a set of neurofeedback trials. Each of the neurofeedback trials consists of several baseline blocks interleaved with regulation blocks and followed by a rest epoch and neurofeedback display block. The connectivity-based feedback is estimated as the difference between the logarithmic evidences of two model alternatives, the target model and the alternative model, which is the logarithmic Bayes factor. What enters the estimation of the DCMs by the end of neurofeedback trial, is the unscaled preprocessed time-series extracted from the ROIs that constitute DCM models. The resulting logarithmic Bayes factor is the feedback value that can be presented to the participants (for more details about experimental parameters, see Data in Brief). *OpenNFT* provides an improved implementation of this trial-based estimation scheme. In particular, DCM models are now computed in parallel, and an extension is provided to integrate the linear trend and head motion residuals directly into the DCM model estimation.

In addition to the activity- and connectivity-based neurofeedback, multivariate pattern analysis (MVPA) based on the support-vector-machine (SVM) classifier was used to allow regulating one brain state versus another (Amano et al., 2016; deBettencourt et al., 2015; LaConte, 2011; Shibata et al., 2011). Note that in MVPA approaches for neurofeedback, there is an important distinction between the classifier/regression model training (i.e., the preparation of the feedback signal estimation), and the decoding of the brain states in real-time (i.e., the real-time feedback signal estimation). Thus, for the example provided (see Data in Brief) we first trained the linear SVM classifier to discriminate between two attentional states using the data from two fMRI functional localizer runs acquired prior to the neurofeedback run. Each run consisted of seven 20 sec regulation blocks that were interleaved with seven 20 sec baseline blocks (for more details about the experimental parameters, see Data in Brief). First- and second-level kernels were built based on the masked whole-brain data and the exemplary classification mask that entailed the parietal lobe (Talairach Daemon atlas (Lancaster et al., 2000)), respectively. The accuracy of the classification was determined using n-fold leave-one-out cross-validation technique applied across blocks of the fMRI runs. The SVM features were voxel-wise mean-centered using training data, the fMRI scans were averaged within a block, and 5 sec delay and overlap of the HRF were taken into account when estimating the classification model. The SVM estimations were performed using the PRONTO toolbox (Schrouff et al., 2013).

Next, the same participant performed a single neurofeedback run that consisted of 10 baseline blocks interleaved with 11 regulation blocks. The classification-based feedback was computed in

real-time using a conventional decision function, namely the dot product between the pre-trained classifier weight vector and the current data vector extracted from the classification mask (LaConte, 2011; LaConte et al., 2007). For the *OpenNFT* implementation of the classification-based feedback, we used similar whole-brain data preprocessing steps, provided an easy interface to set the pre-trained classifier weights and classification mask, and computed a classification-based feedback value per scan using the decision function. The resulting time-series undergoes similar signal processing procedures as for the PSC feedback to remove linear drift, high frequency noise and spikes. Z-scoring and logarithmic sigmoidal transfer function were applied to the processed time-series of the feedback estimates, which allows for a precise mapping between the classifier output and the relative proportion of two overlapping stimuli types that could constitute the feedback signal (deBettencourt et al., 2015).

Feedback presentation

The estimated feedback signal is typically represented in arbitrary units and needs to be converted into stimuli that are meaningful for the participant. This conversion often involves adaptive scaling and thresholding, as well as translating the achieved self-regulation into a (monetary) reward, which can be adapted according to shaping algorithms for operant conditioning (Bray et al., 2007; Skinner, 1953). Feedback is most commonly presented visually. Depending on the experimental needs and hardware availability, the visual feedback presentation can use anything from MR-compatible 2D screens or goggles to more sophisticated MR-compatible 3D displays or virtual-reality goggles. The feedback signal presentation can range from simple visual cues, such as a thermometer (Weiskopf et al., 2004a) or a moving bar (Koush et al., 2012), to avatar faces (Mathiak et al., 2010) and VR-scenes (Baecke et al., 2015).

OpenNFT implementation

OpenNFT (http://opennft.org/) is an open-source integrated software package designed for neurofeedback using rt-fMRI. It is implemented as a unified framework and is based on the platform-independent interpreted programming languages Python (https://www.python.org/) and Matlab (MathWorks, Natick, Massachusetts, United States). Our software is adapted, but not

limited, to use the functionality of the SPM (www.fil.ion.ucl.ac.uk/spm) and Psychtoolbox (psychtoolbox.org) open-source software suites and is distributed via the GitHub open-source platform (https://github.com/OpenNFT).

Architecture

OpenNFT's control core, which contains the parallel architecture, graphical-user-interface (GUI) and synchronization module, is implemented in Python, whilst analytical modules for data processing and neurofeedback estimation are implemented in Matlab. Python is chosen because it is an interpreted software language that provides an extensive support for the concurrent data processing, for flexible GUI design, and for integration of different software modules, e.g. machine learning (Pedregosa et al., 2011), while Matlab provides a comfortable way to program extensions for novel input data, data processing methods and feedback estimation approaches. Integration was done via the recently developed Matlab Engine for Python, which requires Matlab 2015b or higher. Python was used together with the NumPy package (Van der Walt et al., 2011).

OpenNFT is implemented using seven processes executed in parallel: the Python Core process and GUI subprocess, the Python Synchronization process, the Matlab Core process, and three Matlab Helper processes (Fig. 4). The Python Core process provides the control architecture over all the other processes, the inter-process communication, and watches the data from the MR scanner. The Python Core process also provides the control flow for sequential calls of the Matlab Core and Helper processes that are used as entry points for different principal flowchart modules. These calls of Matlab Core and Helper processes are maintained using the Matlab Engine and could be blocking, if a Matlab Engine function is called synchronously, and non-blocking, if an asynchronous Matlab function call is used (i.e., using the FutureResult object provided by the Matlab Engine for Python). Blocking implies that Python Core Process will wait until the Matlab Engine returns the result, and non-blocking implies that Python Core Process will continue running through sequential calls without waiting for the asynchronously launched procedure.

If a new image acquisition is started, the Python Synchronization process receives trigger pulse from the MR scanner and could either be recorded or used to initialize the visualization to the participant. The Python GUI subprocess is implemented as a part of the Python Core process for maintaining GUI interactions and data visualization for the experimenter. The GUI

subprocess implies its implementation as a part of the Python Core process, yet being a separate thread.

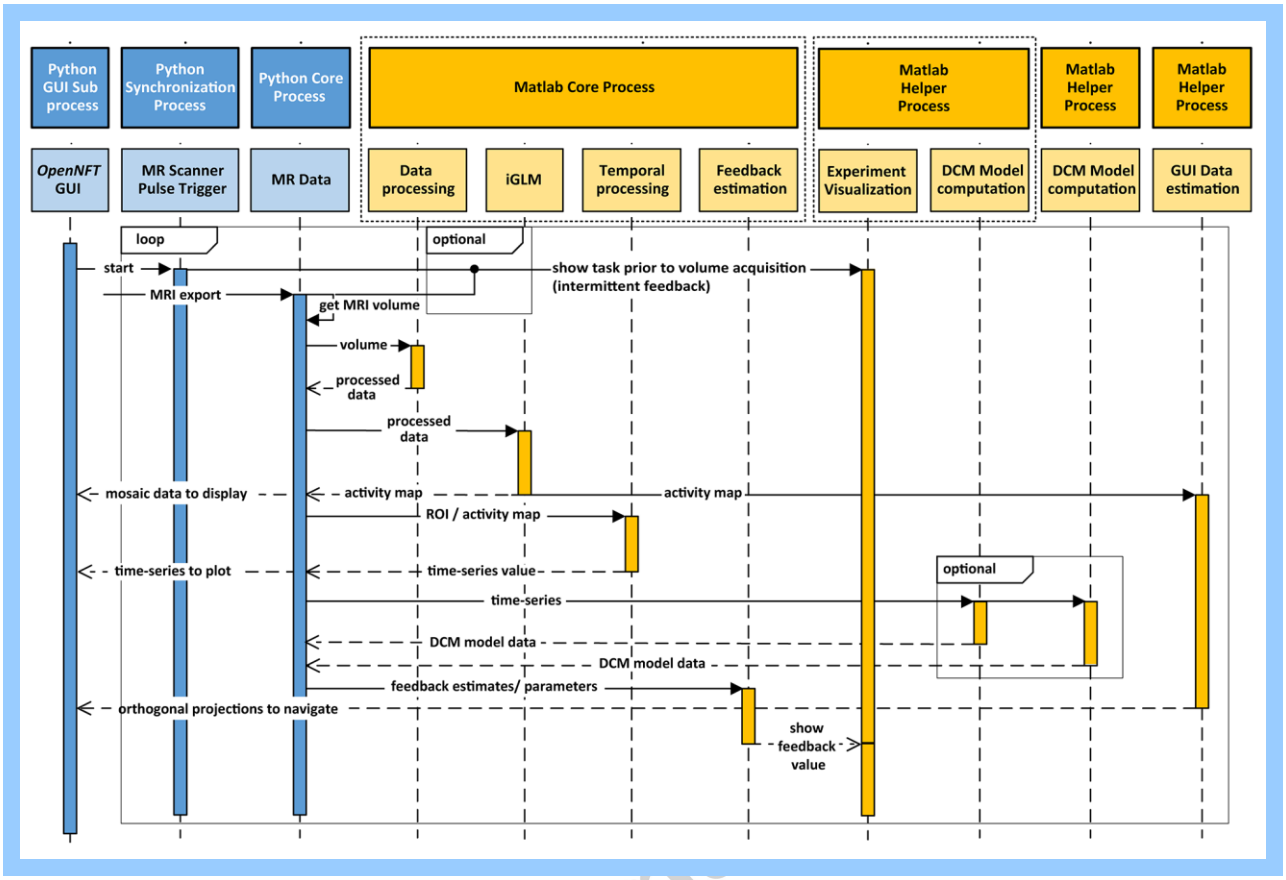

Fig. 4. The *OpenNFT* architecture. Every block at the top level indicates a separate process. Matlab processes are colored in yellow, Python processes in blue. Their functionality is briefly specified in lower more transparent blocks of the same color. Vertical lines under each block indicate calling of the sequence of processes that are within the specific time interval and required for the processing of one data portion from the MR scanner. Colored vertical bars denote active stages of the processes, and dashed lines denote their inactive states. Horizontal arrows denote data transfers between the processes. Note, DCM models are computed using Matlab Helper Processes, which is an optional configuration for DCM feedback type.

The Matlab Core process performs fMRI data (pre)processing, time-series processing, feedback signal estimation and processing. In addition to the Matlab Core process, the first of three Matlab Helper processes is reserved for experimental task and feedback visualization, i.e., Experiment Visualization (Fig. 4). This is particularly useful when feedback is presented as complex dynamic visual stimuli, animations or video sequences. This Matlab Helper can, for example, be used to present visual feedback using the Psychtoolbox (PTB) (Brainard, 1997; Pelli, 1997), which is a Matlab toolbox based on the OpenGL library. In current configuration,

this Matlab Helper process is also reserved for the concurrent estimation of a first DCM model, because during DCM computation the participant's screen is set constant (black/fixation), which also emphasizes how different modules/functions could efficiently share the provided Matlab Helper processes. The second Matlab Helper process is used to facilitate the presentation of the orthogonal views of the brain at a location indicated via a mouse-click, i.e., facilitating the GUI whole-brain navigation. These orthogonal projections are computed using adapted SPM scripts. For DCM feedback, the third Matlab Helper process, the Model Computation (Fig. 4), is used for the concurrent estimation of a second DCM model, i.e., when the feedback display is set to a fixation point (for details, see (Koush et al., 2015; Koush et al., 2013b)).

In current configuration, *OpenNFT* is indirectly triggered by the rt-fMRI scan arrivals (t1) (Fig. 2). In more details, when the new MR export data file arrives, the Python watchdog module catches the corresponding file-system change notification generated by the operation system and waits until the file is completely written (t2). As soon as the feedback signal is calculated by the Matlab Core process, the status of the feedback signal readiness is changed and it becomes available for display (t6). *OpenNFT* has an inbuilt capability to constantly record all the time events (Fig. 2) and the MR trigger pulse (t0) as a reference time vector for displaying the instructions to the participant, monitoring time events of the feedback estimation scheme and adjusting offline data analyses. The trigger pulse is monitored by the Python Synchronization process.

Real-time GUI and parameters

The *OpenNFT* GUI to define the experiment encompasses five main stages: Initialize – Parameter Panel – Setup – Start – Stop (Fig. 5). It is implemented as a Python subprocess, helps to initiate the key software parameters and features, illustrate the necessary data plots in real-time, and provides a real-time access to the whole-brain activation map.

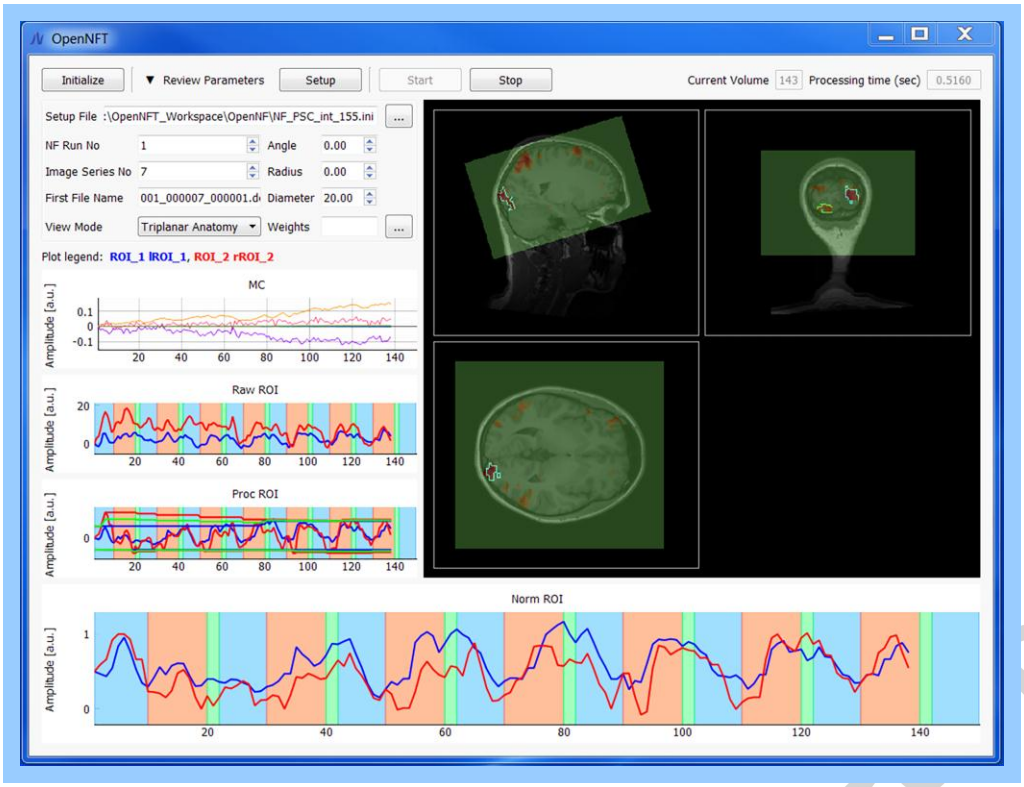

Fig. 5. Screenshot of the *OpenNFT* software in action showing the interface at the end of a neurofeedback run. The intermittent feedback signal was estimated as the percent signal change scaled average between two ROIs (for details, see *'OpenNFT* functionality*'* section). On the left panel, there is a quickset panel, the plot of head motion parameters, the plot of the raw time-series extracted from the two ROIs, the plot of the processed time-series and their dynamic scaling range, and the plot of the feedback signal in the scaled units ready to be converted into the feedback display. The blue, red and green backgrounds of the time-series plots denote baseline, neurofeedback regulation and neurofeedback presentation blocks. The right panel shows orthogonal views of the participant's brain structural scan, the fMRI volume (opaque green area), the ROI mask (green contour), and an activation map based on iGLM statistics.

Clicking "*Initialize"*, launches the initialization of the concurrent Matlab processes. Clicking *"Review Parameters"* opens the panel containing all settings for the operation mode and the data inputs, such as the feedback and data types, paths to the ROIs/masks, the template EPI, the structural scan, and the neurofeedback protocol file (Fig. 6).

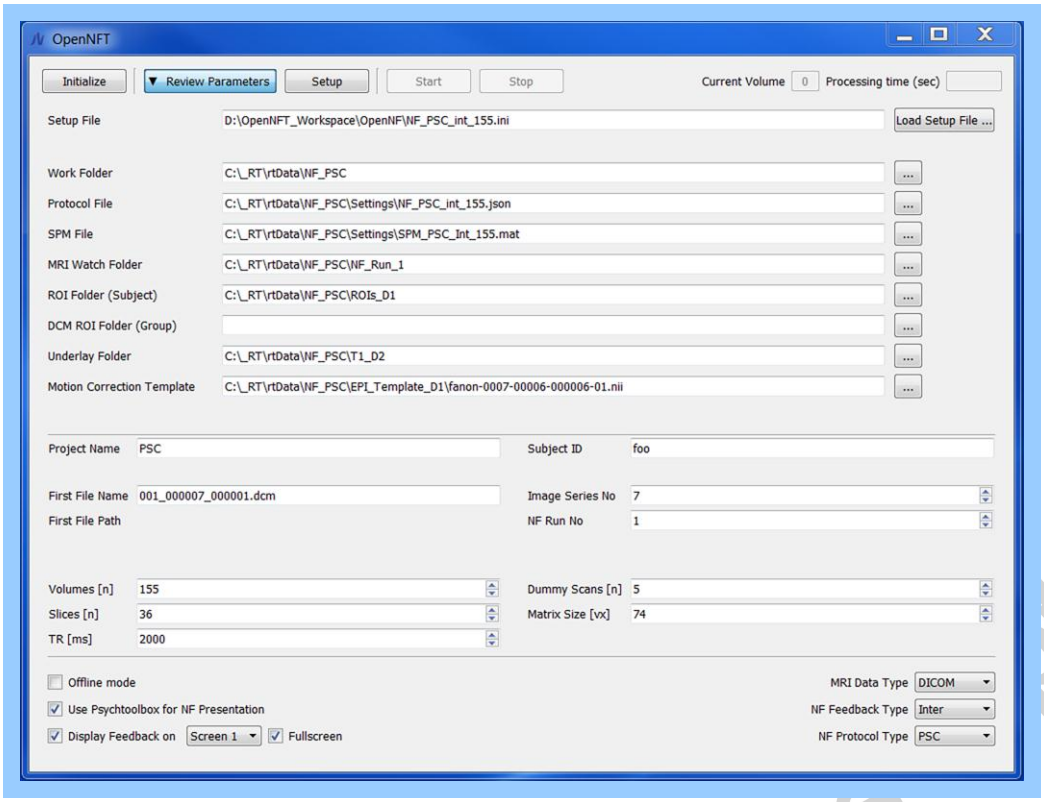

Fig. 6. Screenshot of the window with the main parameter settings.

The main control parameters include the number of volumes and their dimensions, TR, number and the volume name of the data series that will be exported next, and the next neurofeedback run. All the settings and parameters are stored in the initialization INI file that can be prepared prior to the experiment and loaded, rather than entering the parameters manually. The neurofeedback experimental protocol is stored in a JavaScript Object Notation (JSON) format file and contains the timeline of the conditions in terms of the start and end of each block. The experimental protocol can be edited in Matlab and stored in JSON format using the *jsonlab* Matlab toolbox (https://ch.mathworks.com/matlabcentral/fileexchange/33381-jsonlab--a-toolbox-to-encode-decode-json-files). Once the main parameters are set, *OpenNFT* can be switched into real-time operation mode (Fig. 5). In this mode, the quickset parameter panel is available, which allows for a quick set of some parameter settings between runs, e.g., changing the file name of the real-time exported MR images, and the run number (which is also used to make the run-specific data storage folder). To advance to the next run, the user has to change the feedback run number, the image series number, and then click *"Setup"* and *"Start"*. During Setup, all the necessary files are loaded. If necessary for the visual presentation of the selected feedback type, the Psychtoolbox (PTB) is configured and the PTB screen is prepared. The

execution of the principal feedback procedure is started by clicking *Start*, after which the framework idles until it receives real-time data input from the MR scanner (i.e., either the MR trigger pulse or the exported image file). Clicking *"Stop"* terminates the feedback workflow and saves the Matlab Core process workspace and all the user-specified data into the feedback run-specific data folder.

During the rt-fMRI run, head motion correction parameters, the raw and processed time series, an orthogonal view of the structural scan, the fMRI volume, the ROIs, and an activation map are plotted and updated for each volume (Fig. 5). The temporal data plots can be enlarged by double-clicking on the respective panel. The dynamically updated activation map and the ROI contours can be overlaid either onto the current EPI data volume or onto the structural scan, and can be displayed as a 2D mosaic or in an orthogonal interactive viewer. Using the mouse, it is possible to navigate through different views of the orthogonal projection without affecting the timing of data processing.

To guarantee high performance and flexible data flows, the core architecture of *OpenNFT* was motivated by a high degree of parallel processing and modularity. Parallel processing implies independent processes for user interface, synchronization with MRI pulse, data acquisition, data processing, and, among other options, the possibility to launch multiple Matlab processes. High modularity implies the possibility to enable/disable/extend parts of the estimation scheme independently from each other. Practically, this means that developers can combine modules, functions and their settings (e.g., optimized for speed or accuracy) according to the needs of the specific neurofeedback application, and, in addition, they can implement additional modules within the *OpenNFT* architecture, using Python, Matlab and even non-interpreted programing languages (C, C++, Java).

Data streaming

The use of shared data structures is common practice for efficient Python and Matlab implementations. To minimize the overhead that could be caused by data exchange, most of the fMRI data processing steps are performed in the Matlab Core process. The data is shared between the Matlab Core processes using parameter structures without direct sharing the fMRI data. Two synchronized copies of the structure are used to present these shared parameters to the Python Core and Matlab processes. However, currently, there is a limitation for transferring large data amounts using the Python application programming interfaces (API) of the Matlab

Engine; e.g., the time required by this API for transferring a 512×512 uint8 matrix might take up to a second. We thus use the Python API of the Matlab Engine only for sending and receiving scalars and vectors of moderate length; e.g., the parameter structure that is exchanged between all the processes and a reduced version of this structure for the helper processes. In addition, we use the API to send the time-series vectors for plotting from the Matlab core process to the Python GUI subprocess.

OpenNFT uses memory-mapped files to transfer large matrices from the Matlab Core and Helper processes to the Python Core process and GUI sub-process. For example, when an estimated activation map is updated dynamically in the Matlab Core process, it is directly transferred to the Python Core process for display as a mosaic (Fig. 4). However, for displaying the activation map as an overlay over the structural orthogonal views, we first overlay all the displayed items and then compute the orthogonal projections using the Matlab Helper process, to allow for interactive mouse navigation. These projections are then sent to the Python Core process using memory-mapped files.

Importantly, Python allows efficient communication with other software. Interfacing external software (e.g., for stimuli and feedback visualizations) can be performed using the UDP network socket. Using Python and Matlab functionality, *OpenNFT* can also be extended to handle input from a large range of standard hardware (e.g., via USB, parallel and serial data ports). The Python core also allows integrating other packages with *OpenNFT*, and implementing time-critical processing stages in a multithreading environment or on the GPU (graphical processing unit), which is relevant for heavy numerical implementations, e.g. DCM estimations (Aponte et al., 2016).

OpenNFT performance

To demonstrate the *OpenNFT* performance and feasibility, we briefly showcase selected quality measures for three neurofeedback data sets that use (1) continuously and periodically displayed (intermittent) activation-based feedback estimated as a PSC between the activation and baseline conditions; (2) intermittent effective connectivity feedback estimated based on the DCM estimations; (3) continuous classification-based feedback estimated based on the SVM decision function (for details, see Data in Brief). An interested reader can download the anonymized

experimental data that include a single participant for each paradigm and the necessary configuration files, and reproduce the experiments using *OpenNFT*. The real-time performance of the *OpenNFT* can be simulated without MR scanner using real-time simulation routine which copies fMRI scans from an arbitrary source folder to the destination folder specified in *OpenNFT* as 'MRI Watch folder' using a delay between successive copies (i.e., equal to the suggested repetition time (TR)). Similarly, *OpenNFT* operates in an offline mode reading acquired data as quickly as possible from 'MRI Watch folder'.

We performed a set of tests for conventional SPM functions that underwent some adaptations for computationally-sensitive real-time operations in *OpenNFT* and compared them to the conventional SPM analyses without this adaptation. The programming adaptations that do not affect the computations were systematically validated by all the performance checks and the ongoing neurofeedback studies. For example, only the spatial realignment routine underwent the computational modifications, whilst reslicing routine was adopted from the real-time coding perspective. All the adaptations are marked appropriately in code.

Data preprocessing

The SPM realignment procedure based on the B-splines interpolation of the 4^{th} order was adapted to limit the maximum number of iterations (10 iterations) and the accuracy of the approximations (< .01 mm), by this, ensuring the stable convergence of the interpolation given the limited time at TRs of ≤ 1 sec (Koush et al., 2013b; Koush et al., 2012). In addition, masking unwanted voxels that contribute the least to the determinant of the inverse covariance matrix of the parameters is implemented in SPM to speed up the realignment of the successive scans. This algorithm is preserved in *OpenNFT* and could be either optionally disabled for its real-time mode, because it might take several seconds to be computed in the beginning of the acquisitions, or it could be integrated into the first baseline condition when the feedback signal is typically not displayed.

Thus, we first tested two specific modes of the spatial realignment in *OpenNFT* comparing them to the conventional SPM realignment: the matched mode that approaches conventional SPM approach (i.e., the mode that underwent only programing adaptations), and the real-time mode that could be used for real-time applications (i.e., with realignment adaptations outlined above). For comparison purposes, we realigned all the scans in the particular neurofeedback run to the first scan of the run in *OpenNFT* and in SPM. For both modes of the spatial realignment,

we used the same reslicing (B-splines of the 4th order) and smoothing (5mm full-width-at-half-maximum (FWHM)). We found no differences between matched *OpenNFT* mode and the conventional SPM approach in terms of the correlation between head motion estimates, and negligible differences in terms of the correlation between time-series extracted from the preprocessed data given ROIs (Spearman correlation, all rho-values > .9995 and p-values < .0001). For real-time *OpenNFT* mode, the differences were somewhat larger for head-motion estimates (Spearman correlation, all rho-values > 0.9771 and p-values < .0001) and for extracted time-series (Spearman correlation, all rho-values > .9984 and p-values < .0001).

Incremental GLM

We tested the performance of the incremental GLM algorithm (Bagarinao et al., 2003) implemented in *OpenNFT*. For this purpose, we compared the activation maps estimated with the *reduced* and *extended* configuration modes of *OpenNFT* and SPM. Neurofeedback data for this comparison were preprocessed using *OpenNFT* that was configured the same way as the conventional SPM for spatial realignment, reslicing and smoothing operations.

First, for *reduced* GLMs in *OpenNFT* and SPM, we modelled experimental design regressors without regressors of no-interest (for details, see Data in Brief and SPM structures in Demo data). In addition, we disabled the orthogonalization of the modulations in SPM, the correction for serial correlation using autoregressive model of the first order (i.e., the AR(1)) and the high-pass filtering in *OpenNFT* and SPM. Note that the AR(1) is implemented differently in SPM and *OpenNFT*. Thus, for *reduced OpenNFT* and SPM configurations, we found high similarity between thresholded activation maps and their binary activation maps (p-values < .01 unc.) using t-statistics (Tables 1,2; Fig. 7; two-sample, two-tailed). The spatial similarity between corresponding binary activation maps was assessed using the Jaccard index, computed as the size of the intersection of binary maps over the size of their union. For comparison purposes, both, *OpenNFT* and SPM activation maps were masked using a mask created by the corresponding SPM estimation under comparison, because *OpenNFT* does not perform masking for statistical analysis. We estimated the neurofeedback condition-specific contrasts (for details, see Data in Brief).

Next, for *extended* SPM and *OpenNFT* configurations, we further explored the differences between their GLM implementations. Thus, we added six head motion regressors, the linear trend regressor, the 128 sec high-pass filter, the autoregressive voxel-wise time-series filtering in

OpenNFT and the similar AR(1) feature in SPM. Because the precision of the incremental GLM strongly depends on the number of modelled regressors (Misaki et al., 2015) and included data points, whilst SPM explores the whole data set at ones, we observed an expected reduction of the conventional SPM activation maps and their average t-values in SPM as compared to the incremental GLM in *OpenNFT* (Table 1). We also observed a reduction of the Jaccard similarity index between corresponding binary activation maps (Table 2; Fig.7).

Table 1. Average t-values (±sd) for reduced (red.) and extended (ext.) GLM estimations using SPM and *OpenNFT*.

Configuration/ Case study		PSC	SVM	DCM trials					
SPM	red.	3.6±1.1	5.0±2.5	3.5±1.3	3.7±1.4	3.3±1.0	3.5±1.7	3.4±1.0	3.5±1.1	3.6±1.3
	ext.	3.0±0.7	4.4±1.9	3.4±1.1	3.2±0.9	3.0±0.6	3.1±0.7	3.0±0.7	3.2±0.8	3.3±1.0
OpenNFT	red.	3.6±1.1	5.0±2.5	3.6±1.3	3.7±1.5	3.3±1.0	3.5±1.2	3.4±1.0	3.5±1.2	3.6±1.4
	ext.	3.1±0.7	4.7±2.2	3.4±1.1	3.3±0.9	3.1±0.8	3.5±1.1	3.2±0.8	3.5±1.2	3.6±1.3

Table 2. Jaccard indices / t-values (p-values) for comparisons between SPM and *OpenNFT* activation maps.

Configuration/ Case study	PSC	SVM	DCM trials					
reduced	.92 / 0.3 (.80)	.99 / 0.1 (.01)	.87 / 1.9 (.06)	.87 / 2.2 (.03)	.88 / 0.9 (.38)	.88 / 0.9 (.36)	.85 / 0.8 (.45)	.87 / 1.7 (.08)	.87 / 2.3 (.02)
extended	.43 / 2.7 (.01)	.78 / 19.1 (.00)	.78/ 0.8 (.43)	.41 / 1.7 (.09)	.38 / 8.1 (.00)	.51 / 20.0 (.00)	.37 / 5.2 (.00)	.31 / 12.2 (.00)	.52 / 7.3 (.00)

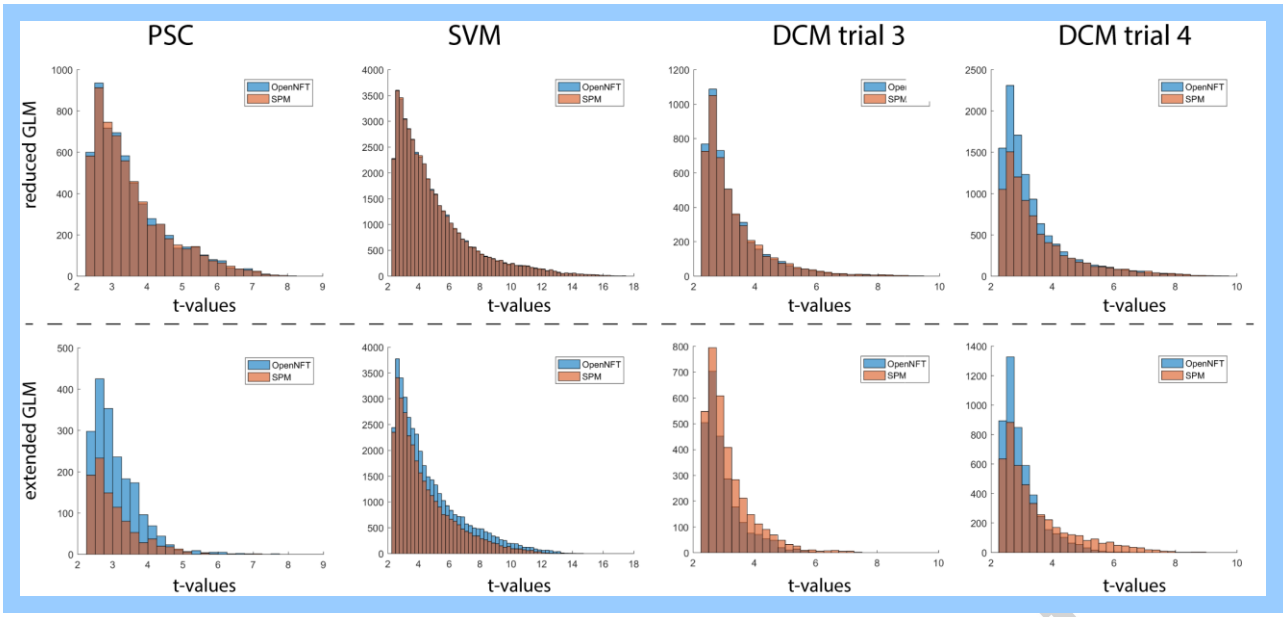

Fig. 7. Histograms of the activation map t-values for reduced and extended GLMs estimated using SPM and *OpenNFT*.

Timing

Finally, we evaluated *OpenNFT* time intervals to characterize the functional data flow modules in the Core Python process (Fig. 2) using real-time data export simulation, real-time data preprocessing and extended iGLM configuration (Table 3).

Table 3. Average data flow durations (±sd), msec.

Time/ Study	t2-t1	t3-t2	t4-t3	t5-t4	t6-t5	t8-t7	*elapsed*
PSC(cont.)	2.3±2.4	427.2±17.3	14.9±4.0	3.2±1.8	3.5±0.6	-	*481.5±26.1*
PSC(int.)	2.3±2.5	431.2±22.0	15.4±4.8	3.2±2.2	4.6±3.3	130.9±29.8	*484.0±31.7*
DCM	4.2±13.6	322.1±17.8	18.1±28.8	9.3±5.4	736.7±47.2	4.5±4.5	*398.3±42.5*
SVM	3.5±2.5	547.7±31.6	17.5±8.4	3.6±1.7	0.1±0.2	-	*610.9±45.0*

For PSC, DCM and SVM studies (74×74×36, 120x120x18 and 100×100×35 data sizes, respectively), t2-t1 – time to read the data volume, t3-t2 – time to spatially preprocess the data, t4-t3 – time to spatially-temporally process the data, t5-t4 – time to estimate and process the feedback signal, t6-t5 – time to

present the feedback signal to the participant (continuous feedback), and t8-t7 – time to present the instruction to the participant (intermittent feedback). int. – intermittent, cont. – continuous.

The time intervals were estimated separately to characterize the functional data flow modules in the Core Python process, which implies some supplementary time for inter-process communication and data transfer in addition to the time that was spent for computations, which could deviate depending on the operation system, workstation configuration and performance. Note that *elapsed* time represents time that the Core Python process spends for processing the single fMRI scan including updating the *OpenNFT* panels and some supplementary data transfers between the concurrent processes which specification was skipped. For example, *elapsed* time does not include time spent in asynchronous parallel processes to display the feedback, or calculate the orthogonal projections in GUI. For intermittent PSC feedback, the display of condition instruction time t7-t6 is somewhat longer, because a fixation dot was used to fix participants' gaze in the center of the screen, which was displayed with random 30-100 msec display period to facilitate flashing (Table 3). Similarly, t8-t5 time interval in DCM feedback implies the feedback flashing with 600-800 msec random display time. For DCM feedback, both DCM models converged in 13.7±2.7 seconds (for more details see (Koush et al., 2015; Koush et al., 2013b)). The SVM feedback was sent out using UDP, which is quick in *OpenNFT* and implies an additional time to account for to present the stimuli/feedback using an additional software.

The performance tests were conducted using a portable Dell Precision 7510 workstation with 32 GB RAM, 512 GB SSD drive and Intel Xeon CPU with 4 free cores. The minimum system requirements for *OpenNFT* are 8 GB RAM, i5 CPU with 2 or 4 free cores for two modes of the software using 3 or 4 Matlab processes, respectively, which needs to be compromised with the neurofeedback study design complexity, computational demands and the repetition time of the data acquisition. The recommended system configuration is 16 GB RAM and i7 CPU with 4 free cores. Note that time event metrics deviates on different workstations due to their hardware and software differences.

Discussion

Neurofeedback developments, applications and perspectives have been thoroughly investigated in recent reviews (Sitaram et al., 2017; Stoeckel et al., 2014; Sulzer et al., 2013). Here, we mostly focus on technical, computational and programming aspects of the neurofeedback studies. We performed an overview of the key methods required to build various feedback estimation schemes and provided a modular open-source software framework using conventional and more recent feedback signal estimates. This includes activation-based feedback, more sophisticated connectivity-based feedback and classification-based feedback.

Data (pre)processing

Necessary data (pre)processing modules are rationalized, on one side, by the required neurofeedback signal (e.g., activity-, connectivity- or classification-based), the neurofeedback paradigm (e.g., continuous, intermittent or trial-based), the form of the feedback signal presentation (e.g., visual, audio, tactile), and, on the other side, by the amount of data to be processed in real-time and the repetition time (TR) of the rt-fMRI data acquisitions. In particular, SPM remains one of the most commonly used open-source software tools in neuroimaging written in Matlab, from which we adopted conventional fMRI data realignment, reslicing, the estimation of the whole-brain orthogonal projections for GUI, and DCM estimations for connectivity-based feedback (Koush et al., 2015; Koush et al., 2013b).

Different artifacts considerably affect the quality of the whole-brain and extracted time-series data, especially if their signal amplitude exceeds twice the standard deviation of regular BOLD fluctuations (Koush et al., 2012; Weiskopf et al., 2004b). Such artifacts can be induced by the inhomogeneity of the static magnetic field, electrostatic discharges, motion-by-susceptibility interaction during head movements, eye-movement, irregular respiration, swallowing, and heart beats (Diedrichsen and Shadmehr, 2005; van Gelderen et al., 2007). Especially in patients, some of these non-linear artifacts can result in spikes (Koush et al., 2012). Depending on the feedback estimation scheme, the end-user can decide if whole-brain data (pre)processing is sufficient to provide enough quality of the extracted time-series, and apply additional signal processing to improve the feedback signal (Koush et al., 2012). In particular, *OpenNFT* provides a set of tools to preprocess the data, and remove residuals, noise and artifacts on the level of whole-brain and time-series data processing (Fig. 1).

The visualization of the whole-brain activity in real-time using incremental GLM (iGLM) (Bagarinao et al., 2003) provides access to the participant's performance during the

neurofeedback experiment. The iGLM allows for modeling the regressors of interest, estimation of the contrasts of interests, removing low- and high-frequency noise and confounds, such as those related to head motion, respiration and cardiac rhythm (Misaki et al., 2015). In addition, the iGLM can be applied in specific dynamic feedback estimation schemes; e.g., for dynamic ROI definitions based on the moderately thresholded statistical maps provided at the end of each neurofeedback trial. Such a dynamic ROI adaptation has been shown to be beneficial for connectivity-based feedback (Koush et al., 2015). If needed for future experiments, the whole-brain iGLM can also be easily extended for sliding-window GLMs (Nakai et al., 2006). However, the calculation of the whole-brain iGLM can be blocked if it is not needed (Koush et al., 2012). To process just the extracted time-series, we provide an option to compute the cumulative GLM which can be further extended to be calculated incrementally or based on a sliding-window. Overall, estimating whole-brain and time-series GLMs is computationally not demanding and can thus be done in real-time, but the limited data available in a real-time setting poses difficulties that are inherent to any neurofeedback experiment. In particular, partial GLM estimations suffer from high sensitivity to the provided data length and the number of modeled regressors (Misaki et al., 2015; Nakai et al., 2006). Indeed, a comparison between the available GLM configurations in *OpenNFT* and that of SPM showed reduced t-map activations and a reduction of the Jaccard similarity index for binary activation maps (Table 1,2; Fig. 7).

Feedback estimation

In addition to conventional activity feedback estimation using percent signal change, we implemented more advanced connectivity-based and classification-based feedback estimations. It has previously been shown that a connectivity-based feedback signal based on DCM can be controlled in a differential visual-spatial attention paradigm (Koush et al., 2013b), and can be trained in an emotion-regulation paradigm (Koush et al., 2015). We provided an open-source implementation of this approach to facilitate its application and improved its estimation scheme. In particular, we parallelized the estimation of the target and the opposed DCM models, and added a possibility to include regressors of no interest into the DCM model estimations at the end of the neurofeedback trial. This new approach shortened the DCM estimation by a factor of two compared to the previously used sequential estimation of the models. The number of DCM models processed simultaneously can now be easily extended depending on how many Matlab processes can be simultaneously processed by the local workstation, and this without reduction of the computational efficiency. In the future, massive DCM estimations can substantially be

sped up by using the GPU (Aponte et al., 2016), and integrating them with multiple Matlab processes. Importantly, a special attention needs to be paid to the differences between default settings for DCM estimation parameters in SPM8 and SPM12. We addressed these differences and provided recommended configurations in offline DCM test functions available under GitHub repository.

In addition to providing an easily usable tool to implement computationally challenging connectivity-based neurofeedback, *OpenNFT* also provides complex MVPA feedback. Although MVPA approaches have been successfully used for real-time fMRI and neurofeedback (Cortese et al., 2016; Shibata et al., 2016) (Amano et al., 2016; deBettencourt et al., 2015; LaConte, 2011), a thorough investigation of the classification/regression update schemes, their influence on feedback learning and further advances towards incremental classification/regression approaches remain to be addressed in the future studies (e.g., compare (deBettencourt et al., 2015; LaConte et al., 2007; Shibata et al., 2011)). For example, an incremental SVM might allow for incremental training of the classifier and estimation of the decision function after each rt-fMRI scan (Cauwenberghs and Poggio, 2001), something which be easily implemented in future versions of *OpenNFT*.

OpenNFT implementation, set up and timing considerations

Our goal was to balance best practices in Python and Matlab programming to implement conventional and advanced neurofeedback measures based on rt-fMRI. The core programing engine is Python, which provides larger functionality and flexibility than Matlab. Based on this core, we integrate Matlab processes to add specific functions. Together, Python and Matlab allow for maximal flexibility and an efficient integration (Table 3). *OpenNFT* modularity is implied by its architecture, which encompasses seven processes written in two different programming practices, so that its multi-processing, GUI, and synchronization are written in Python and can be easily distinguished from the computational part written mostly in Matlab (Fig. 4). Such modularity allows for rapid implementation of different rt-fMRI paradigms and can easily be extended, despite somewhat complex software architecture and interdependent programming logics. The choice for Matlab to accomplish the most of the necessary computations is motivated by its popularity, availability, strong mathematical potential and relatively easy scripting language, as well as by the availability and the potential of Matlab-based toolboxes as such SPM, PRONTO and PTB. *OpenNFT* is not limited to the use of Matlab and

Matlab-based toolboxes, as different parts or even all of the Matlab code can be replaced with Python code. For example, *OpenNFT* uses PTB toolbox functions for feedback presentation, but the feedback values could also be provided using UDP commands (as it is implemented for classification-based feedback). For the moment, the computational part is implemented in Matlab to facilitate the quick and efficient integration of the novel and existing computational approaches so that experienced developers can use a flexible set of implemented features and the parallel architecture both, to extend already implemented data processing schemes and to develop their own. Depending on the growing popularity of Python vs. Matlab, future versions of *OpenNFT* can flexibly adapt to such developments and might focus more on one or the other.

The *OpenNFT* architecture is based on multiple concurrent processes: the Core Python process, the Python Synchronization process, the Python GUI subprocess, the Matlab Core process, and three Matlab Helper processes (Fig. 4). The Core Python process provides the control architecture over all the other processes and the inter-process communication. The Matlab Core process performs the major computations including whole-brain and time-series data (pre)processing and feedback estimation. The total number of implemented Python and Matlab processes can be flexibly reduced or extended, depending on the required software functionality and on the workstation capabilities. For example, to reserve more time for a precise spatial realignment, to perform heavy computations in real-time, to facilitate an interactive brain activity navigation and feedback visualization, we used a multi-processing approach and implemented an estimation of the DCM models, orthogonal projections and feedback visualization functions separately from the Matlab Core process. Matlab Helper processes can be used to support heavy computations and their number can be further extended, as well as GPUs could be used (Aponte et al., 2016). In addition, *OpenNFT* records specific time events that correspond to the presentation of stimuli, data acquisition, data processing, and feedback presentation (Table 3), which can then be included in post-hoc analyses.

An integrated Python/Matlab framework requires multiple components to work together, which includes the Python environment that interacts with Matlab using the Matlab Engine API. Both programming practices are written in interpreted programming languages, hence, *OpenNFT* represents itself a set of packages and functions running under the roof of an integrated framework. We provide a detailed installation manual accompanied with exemplary datasets, simulations, and testing routines to ease the installation and use of the *OpenNFT* framework (see

Data in Brief and GitHub project repository). Once the framework is settled following these guidelines, *OpenNFT* has been shown to work as a stable and reliable tool on different operational systems and workstations. Further information can be obtained through direct contact with the developers via the webpage http://opennft.org/ and GitHub repository, which also provides the communication channel for feedback on potential problems, bugs or possible extensions. We anticipate that *OpenNFT* will greatly benefit from improvements of future SPM, PRONTO and PTB versions, as well as from contributions of its users.

Neurofeedback based on rt-fMRI is typically presented to the participant once per repetition time (TR) of the data acquisition (i.e., with 1 - 3 sec period) (Fig. 2). Further increases of the temporal resolution of the rt-fMRI feedback signal is mainly limited by the intrinsic haemodynamic delay of the BOLD signal (3 – 5 sec). However, increasing the temporal resolution (i.e., TR ≤ 1 sec) might increase the signal-to-noise ratio and the reliability of the computationally demanding methods, i.e., connectivity estimates (Koush et al., 2013b). Like in conventional SPM analyses, the haemodynamic delay is taken into account when modeling the regressors of interest as a convolution of the box-car function with the haemodynamic response function. In *OpenNFT*, this applies to the whole-brain iGLM and the time-series cumulative GLM. During experiments, participants are always informed about the intrinsic haemodynamic delay to their brain activity in continuous neurofeedback paradigms. Some paradigms, such as intermittent and trial-based designs, do not suffer from this disadvantage so much (Koush et al., 2015, Johnson et al., 2012).

In our implementations, real-time data preprocessing takes approx. 80-90% of the total processing time (Table 3), mainly due to the high precision level of the real-time realignment that comes close to the quality of conventional offline realignment in SPM. If faster feedback estimations are needed, this can be easily achieved by reducing the complexity of the applied algorithms (e.g., using linear instead of 4^{th} order B-spline interpolation), and by blocking estimation modules that are not needed for the current experiment. For example, the real-time preprocessing time can be reduced by masking out the voxels that non-significantly contribute to the spatial realignment as implemented in SPM, which is an optional feature in *OpenNFT*. Also, the rt-fMRI brain volumes acquired during the current neurofeedback session can be realigned to the current session realignment template, which typically results in a reduction of the initial displacement and speeds up the convergence rate of the spatial realignment algorithms. In this

case and if otherwise required by the feedback estimation scheme, previously prepared ROIs/masks/weights, including those from the previous sessions, have to be coregistered to the same realignment template. However, for conventional scanning parameters, and using conventional computer hardware, measures for improving the speed of computations won't be necessary because the standard pipeline can usually be performed in time.

Software overview for neurofeedback based on rt-fMRI

As it was outlined above in detail, *OpenNFT* is an open-source GUI-based multi-processing integrated Python/Matlab framework with a broad functionality asset for neurofeedback studies. The combination of the outlined features makes it the first software of this type. Due to the high heterogeneity of the available software and because not all of their components are freely accessible or publicly documented, we are not able to provide a systematic comparison between them. Instead, we provide criteria that can be used to evaluate neurofeedback software, and discuss *OpenNFT* in that context. Although this list is not exhaustive, we consider availability, programming language, performance, interface, extendibility, and functionality. In contrast to other software packages that are commercial (e.g., TurboBrainVoyager), non-commercial but available only upon request (e.g., scanSTAT), or freely available but not open-source (e.g., AFNI real-time plug-in), *OpenNFT* is non-commercial, freely available under the GNU GPL license, and open-source. Thus, it is most easily accessible and its development will benefit from contributions of the user community. *OpenNFT* is implemented in interpreted Python/Matlab that does not require source-code compilation as needed for example for C or C++ (e.g. as in FRIEND, BART, AFNI). This has the advantages that programming in these languages is relatively simple, adaptable to an extensive array of different computer platforms and operating systems (currently tested for Windows 7 to 10, and macOS), and that it allows for integrating available libraries to easily extend functionality or adapt to specific needs. On the other hand, interpreted languages require the installation of a framework to interpret and run the code (i.e., Matlab), and GUI programming, parallel architecture and concurrent data processing is weaker compared to compiled code. To partly overcome these drawbacks while maintaining the advantages of using Matlab (simplicity of programming, extendibility using existing toolboxes), the core process is implemented in Python, which offers extensive GUI capabilities, parallel architecture and concurrent data processing that is similarly performant as that of compiled code. While compiled languages typically possess the higher performance characteristics, extensive test of *OpenNFT* have shown that conventional computer hardware is sufficient to perform all

computations that are necessary for real-time fMRI based neurofeedback. The GUIs of neurofeedback software range from advanced 3D visualizations to no GUI at all. Although the optimal GUI varies depending on the neurofeedback study design, it is evident that a GUI can ease the use of the software, avoids mistakes of the users, and benefits monitoring the progress of the experiment. For this reason, *OpenNFT* provides a GUI that can be used to insert, modify, and review the experimental parameters before the run has started, including the specification of quick-set parameters that repeat between neurofeedback runs and sessions. The GUI also allows for observing the ROI localizations, whole-brain and ROI-specific activations, head motion parameters and the extracted time-series in real-time during the experiment. Furthermore, the GUI can also easily be modified to adapt it to the specific needs of each experiment or to the preferences of the user. In terms of functionality, OpenNFT offers conventional ROI-based feedback (including the use of multiple and adaptively changed ROIs), advanced effective connectivity feedback, as well as MVPA feedback.

Outlook

The field of rt-fMRI is advancing rapidly. It started with neurofeedback from BOLD measures in a single ROI, but now it became possible to provide feedback based on sophisticated measures of brain connectivity and patterns of brain activity. New developments in fMRI, such as the use of VR and hyperscanning (where two participants can interact while being scanned in two different MR scanners at the same time) allow for highly immersive and interactive paradigms that can be adapted for rt-fMRI purposes (Baecke et al., 2015; Montague et al., 2002; Mueller et al., 2012). Even simultaneous EEG-fMRI has recently been accomplished in real-time for neurofeedback purposes (Zich et al., 2015; Zotev et al., 2014). These novel developments will benefit from a modular open-source rt-fMRI framework that facilitates interfacing and integration of new approaches to support the progress in this highly dynamic field of research. The interested researcher is encouraged to actively engage in the development of novel neurofeedback approaches using the open-source opportunities provided through GitHub that will make it easy to follow the software updates, make your own fork releases, and request the software team to integrate your own development into the next core release.

Author contributions:

YK – conceived and coordinated the study, defined software architecture and functionality, implemented the Matlab functionality, and conducted the software performance check. AN, YK – designed the parallel architecture. AN, EP, YK, RS, SB – implemented the Python architecture and functionality, checked and optimized the Matlab implementations. JA – justified and optimized the real-time modifications of the SPM spatial preprocessing; PZ – justified and optimized the SPM DCM estimations; DVDV – justified and optimized GLM analysis and computational processing. YK, DVDV, FS – designed the case studies. YK – acquired and processed the case studies data and demonstrated the software performance. YK, FS, AN, DVDV – wrote the manuscript. RS, YK – designed the website and *OpenNFT* logo. All co-authors critically checked and edited the manuscript, the website content and the supporting documentation.

Acknowledgements

This study was supported in part by the Swiss National Science Foundation (YK: P300PB_161083, FS: BSSG10_155915), in part by the Wyss Center at the Campus Biotech Geneva. FS is supported by the Foundation for Research in Science and the Humanities at the University of Zurich (STWF-17-012). AN, EP and SB are supported by the Samara University and Aligned Research Group. AN is supported by the Federal Grant MD-2531.2017.9. The authors declare no competing financial interests.

References

Amano, K., Shibata, K., Kawato, M., Sasaki, Y., Watanabe, T., 2016. Learning to Associate Orientation with Color in Early Visual Areas by Associative Decoded fMRI Neurofeedback. Curr Biol 26, 1861-1866.

Aponte, E.A., Raman, S., Sengupta, B., Penny, W.D., Stephan, K.E., Heinzle, J., 2016. mpdcm: A toolbox for massively parallel dynamic causal modeling. J Neurosci Methods 257, 7-16.

Baecke, S., Lutzkendorf, R., Mallow, J., Luchtmann, M., Tempelmann, C., Stadler, J., Bernarding, J., 2015. A proof-of-principle study of multi-site real-time functional imaging at 3T and 7T: Implementation and validation. Scientific reports 5, 8413.

Bagarinao, E., Matsuo, K., Nakai, T., Sato, S., 2003. Estimation of general linear model coefficients for real-time application. NeuroImage 19, 422-429.

Brainard, D.H., 1997. The Psychophysics Toolbox. Spat Vis 10, 433-436.

Bray, S., Shimojo, S., O'Doherty, J.P., 2007. Direct instrumental conditioning of neural activity using functional magnetic resonance imaging-derived reward feedback. The Journal of neuroscience : the official journal of the Society for Neuroscience 27, 7498-7507.

Caria, A., Sitaram, R., Birbaumer, N., 2012. Real-time fMRI: a tool for local brain regulation. Neuroscientist 18, 487-501.

Cauwenberghs, G., Poggio, T., 2001. Incremental and decremental support vector machine learning. Neural information processing systems., 409–415.

Chiew, M., LaConte, S.M., Graham, S.J., 2012. Investigation of fMRI neurofeedback of differential primary motor cortex activity using kinesthetic motor imagery. Neuroimage 61, 21-31.

Cohen, M.S., 2001. Real-time functional magnetic resonance imaging. Methods 25, 201-220.

Cortese, A., Amano, K., Koizumi, A., Kawato, M., Lau, H., 2016. Multivoxel neurofeedback selectively modulates confidence without changing perceptual performance. Nature Communications 7.

Cox, R.W., Jesmanowicz, A., Hyde, J.S., 1995. Real-Time Functional Magnetic-Resonance-Imaging. Magnetic Resonance in Medicine 33, 230-236.

deBettencourt, M.T., Cohen, J.D., Lee, R.F., Norman, K.A., Turk-Browne, N.B., 2015. Closed-loop training of attention with real-time brain imaging. Nat Neurosci 18, 470-475.

deCharms, R.C., 2008. Applications of real-time fMRI. Nat Rev Neurosci 9, 720-729.

deCharms, R.C., Maeda, F., Glover, G.H., Ludlow, D., Pauly, J.M., Soneji, D., Gabrieli, J.D., Mackey, S.C., 2005. Control over brain activation and pain learned by using real-time functional MRI. Proc Natl Acad Sci U S A 102, 18626-18631.

Diedrichsen, J., Shadmehr, R., 2005. Detecting and adjusting for artifacts in fMRI time series data. NeuroImage 27, 624-634.

Friston, K.J., Harrison, L., Penny, W., 2003. Dynamic causal modelling. NeuroImage 19, 1273-1302.

Haller, S., Birbaumer, N., Veit, R., 2010. Real-time fMRI feedback training may improve chronic tinnitus. European radiology 20, 696-703.

Hartwell, K.J., Prisciandaro, J.J., Borckardt, J., Li, X.B., George, M.S., Brady, K.T., 2013. Real-Time fMRI in the Treatment of Nicotine Dependence: A Conceptual Review and Pilot Studies. Psychology of Addictive Behaviors 27, 501-509.

Hellrung, L., Hollmann, M., Zscheyge, O., Schlumm, T., Kalberlah, C., Roggenhofer, E., Okon-Singer, H., Villringer, A., Horstmann, A., 2015. Flexible Adaptive Paradigms for fMRI Using a Novel Software Package 'Brain Analysis in Real-Time' (BART). PLoS One 10.

Kim, D.Y., Yoo, S.S., Tegethoff, M., Meinlschmidt, G., Lee, J.H., 2015. The inclusion of functional connectivity information into fMRI-based neurofeedback improves its efficacy in the reduction of cigarette cravings. J Cogn Neurosci 27, 1552-1572.

Koush, Y., Elliott, M.A., Mathiak, K., 2011. Single Voxel Proton Spectroscopy for Neurofeedback at 7 Tesla. Materials (Basel) 4.

Koush, Y., Elliott, M.A., Scharnowski, F., Mathiak, K., 2013a. Real-time automated spectral assessment of the BOLD response for neurofeedback at 3 and 7T. J Neurosci Methods 218, 148-160.

Koush, Y., Elliott, M.A., Scharnowski, F., Mathiak, K., 2014. Comparison of real-time water proton spectroscopy and echo-planar imaging sensitivity to the BOLD effect at 3 T and at 7 T. PLoS One 9, e91620.

Koush, Y., Meskaldji, D.E., Pichon, S., Rey, G., Rieger, S.W., Linden, D.E., Van De Ville, D., Vuilleumier, P., Scharnowski, F., 2015. Learning Control Over Emotion Networks Through Connectivity-Based Neurofeedback. Cereb Cortex.

Koush, Y., Rosa, M.J., Robineau, F., Heinen, K., S, W.R., Weiskopf, N., Vuilleumier, P., Van De Ville, D., Scharnowski, F., 2013b. Connectivity-based neurofeedback: dynamic causal modeling for real-time fMRI. NeuroImage 81, 422-430.

Koush, Y., Zvyagintsev, M., Dyck, M., Mathiak, K.A., Mathiak, K., 2012. Signal quality and Bayesian signal processing in neurofeedback based on real-time fMRI. NeuroImage 59, 478-489.

LaConte, S.M., 2011. Decoding fMRI brain states in real-time. NeuroImage 56, 440-454.

LaConte, S.M., Peltier, S.J., Hu, X.P.P., 2007. Real-time fMRI using brain-state classification. Human Brain Mapping 28, 1033-1044.

Lancaster, J.L., Woldorff, M.G., Parsons, L.M., Liotti, M., Freitas, E.S., Rainey, L., Kochunov, P.V., Nickerson, D., Mikiten, S.A., Fox, P.T., 2000. Automated Talairach Atlas labels for functional brain mapping. Human Brain Mapping 10, 120-131.

Li, X.B., Hartwell, K.J., Borckardt, J., Prisciandaro, J.J., Saladin, M.E., Morgan, P.S., Johnson, K.A., LeMatty, T., Brady, K.T., George, M.S., 2013. Volitional reduction of anterior cingulate cortex activity produces decreased cue craving in smoking cessation: a preliminary real-time fMRI study. Addiction Biology 18, 739-748.

Liew, S.L., Rana, M., Cornelsen, S., de Barros, M.F., Birbaumer, N., Sitaram, R., Cohen, L.G., Soekadar, S.R., 2016. Improving Motor Corticothalamic Communication After Stroke Using Real-Time fMRI Connectivity-Based Neurofeedback. Neurorehabilitation and neural repair 30, 671-675.

Linden, D.E., Habes, I., Johnston, S.J., Linden, S., Tatineni, R., Subramanian, L., Sorger, B., Healy, D., Goebel, R., 2012. Real-time self-regulation of emotion networks in patients with depression. PLoS One 7, e38115.

Logothetis, N.K., Pauls, J., Augath, M., Trinath, T., Oeltermann, A., 2001. Neurophysiological investigation of the basis of the fMRI signal. Nature 412, 150-157.

Maclaren, J., Herbst, M., Speck, O., Zaitsev, M., 2013. Prospective motion correction in brain imaging: a review. Magn Reson Med 69, 621-636.

Mathiak, K.A., Koush, Y., Dyck, M., Gaber, T.J., Alawi, E., Zepf, F.D., Zvyagintsev, M., Mathiak, K., 2010. Social reinforcement can regulate localized brain activity. European archives of psychiatry and clinical neuroscience 260 Suppl 2, S132-136.

Megumi, F., Yamashita, A., Kawato, M., Imamizu, H., 2015. Functional MRI neurofeedback training on connectivity between two regions induces long-lasting changes in intrinsic functional network. Front Hum Neurosci 9, 160.

Misaki, M., Barzigar, N., Zotev, V., Phillips, R., Cheng, S., Bodurka, J., 2015. Real-time fMRI processing with physiological noise correction - Comparison with off-line analysis. J Neurosci Methods 256, 117-121.

Montague, P.R., Berns, G.S., Cohen, J.D., McClure, S.M., Pagnoni, G., Dhamala, M., Wiest, M.C., Karpov, I., King, R.D., Apple, N., Fisher, R.E., 2002. Hyperscanning: simultaneous fMRI during linked social interactions. NeuroImage 16, 1159-1164.

Monti, R.P., Lorenz, R., Braga, R.M., Anagnostopoulos, C., Leech, R., Montana, G., 2016. Real-time estimation of dynamic functional connectivity networks. Human Brain Mapping.

Mueller, C., Luehrs, M., Baecke, S., Adolf, D., Luetzkendorf, R., Luchtmann, M., Bernarding, J., 2012. Building virtual reality fMRI paradigms: a framework for presenting immersive virtual environments. J Neurosci Methods 209, 290-298.

Nakai, T., Bagarinao, E., Matsuo, K., Ohgami, Y., Kato, C., 2006. Dynamic monitoring of brain activation under visual stimulation using fMRI--the advantage of real-time fMRI with sliding window GLM analysis. J Neurosci Methods 157, 158-167.

Paret, C., Kluetsch, R., Ruf, M., Demirakca, T., Hoesterey, S., Ende, G., Schmahl, C., 2014. Down-regulation of amygdala activation with real-time fMRI neurofeedback in a healthy female sample. Front Behav Neurosci 8, 299.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, 2825-2830.

Pelli, D.G., 1997. The VideoToolbox software for visual psychophysics: transforming numbers into movies. Spat Vis 10, 437-442.

Robineau, F., Rieger, S.W., Mermoud, C., Pichon, S., Koush, Y., Van De Ville, D., Vuilleumier, P., Scharnowski, F., 2014. Self-regulation of inter-hemispheric visual cortex balance through real-time fMRI neurofeedback training. Neuroimage 100, 1-14.

Scharnowski, F., Veit, R., Zopf, R., Studer, P., Bock, S., Diedrichsen, J., Goebel, R., Mathiak, K., Birbaumer, N., Weiskopf, N., 2015. Manipulating motor performance and memory through real-time fMRI neurofeedback. Biol Psychol 108, 85-97.

Shibata, K., Watanabe, T., Kawato, M., Sasaki, Y., 2016. Differential Activation Patterns in the Same Brain Region Led to Opposite Emotional States. PLoS Biol 14, e1002546.

Shibata, K., Watanabe, T., Sasaki, Y., Kawato, M., 2011. Perceptual learning incepted by decoded fMRI neurofeedback without stimulus presentation. Science 334, 1413-1415.

Sitaram, R., Ros, T., Stoeckel, L., Haller, S., Scharnowski, F., Lewis-Peacock, J., Weiskopf, N., Blefari, M.L., Rana, M., Oblak, E., Birbaumer, N., Sulzer, J., 2017. Closed-loop brain training: the science of neurofeedback. Nature Neuroscience Reviews.

Skinner, B.F., 1953. Science and human behavior. Oxford: Macmillan.

Stoeckel, L.E., Garrison, K.A., Ghosh, S., Wighton, P., Hanlon, C.A., Gilman, J.M., Greer, S., Turk-Browne, N.B., deBettencourt, M.T., Scheinost, D., Craddock, C., Thompson, T., Calderon, V., Bauer, C.C., George, M., Breiter, H.C., Whitfield-Gabrieli, S., Gabrieli, J.D., LaConte, S.M., Hirshberg, L., Brewer, J.A., Hampson, M., Van Der Kouwe, A., Mackey, S., Evins, A.E., 2014. Optimizing real time fMRI neurofeedback for therapeutic discovery and development. Neuroimage Clin 5, 245-255.

Subramanian, L., Hindle, J.V., Johnston, S., Roberts, M.V., Husain, M., Goebel, R., Linden, D., 2011. Real-time functional magnetic resonance imaging neurofeedback for treatment of Parkinson's disease. J Neurosci 31, 16309-16317.

Sulzer, J., Haller, S., Scharnowski, F., Weiskopf, N., Birbaumer, N., Blefari, M.L., Bruehl, A.B., Cohen, L.G., DeCharms, R.C., Gassert, R., Goebel, R., Herwig, U., LaConte, S., Linden, D., Luft, A., Seifritz, E., Sitaram, R., 2013. Real-time fMRI neurofeedback: progress and challenges. Neuroimage 76, 386-399.

Todd, N., Moeller, S., Auerbach, E.J., Yacoub, E., Flandin, G., Weiskopf, N., 2016. Evaluation of 2D multiband EPI imaging for high-resolution, whole-brain, task-based fMRI studies at 3T: Sensitivity and slice leakage artifacts. NeuroImage 124, 32-42.

Van der Walt, S., Colbert, S.C., Var, G., 2011. The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering, 22-30.

van Gelderen, P., de Zwart, J.A., Starewicz, P., Hinks, R.S., Duyn, J.H., 2007. Real-time shimming to compensate for respiration-induced B0 fluctuations. Magn Reson Med 57, 362-368.

Weiskopf, N., 2012. Real-time fMRI and its application to neurofeedback. NeuroImage 62, 682-692.

Weiskopf, N., Hutton, C., Josephs, O., Turner, R., Deichmann, R., 2007a. Optimized EPI for fMRI studies of the orbitofrontal cortex: compensation of susceptibility-induced gradients in the readout direction. Magnetic Resonance Materials in Physics Biology and Medicine 20, 39-49.

Weiskopf, N., Mathiak, K., Bock, S.W., Scharnowski, F., Veit, R., Grodd, W., Goebel, R., Birbaumer, N., 2004a. Principles of a brain-computer interface (BCI) based on real-time functional magnetic resonance imaging (fMRI). IEEE Trans Biomed Eng 51, 966-970.

Weiskopf, N., Scharnowski, F., Veit, R., Goebel, R., Birbaumer, N., Mathiak, K., 2004b. Self-regulation of local brain activity using real-time functional magnetic resonance imaging (fMRI). J Physiol Paris 98, 357-373.

Weiskopf, N., Sitaram, R., Josephs, O., Veit, R., Scharnowski, F., Goebel, R., Birbaumer, N., Deichmann, R., Mathiak, K., 2007b. Real-time functional magnetic resonance imaging: methods and applications. Magn Reson Imaging 25, 989-1003.

Young, K.D., Zotev, V., Phillips, R., Misaki, M., Yuan, H., Drevets, W.C., Bodurka, J., 2014. Real-Time fMRI Neurofeedback Training of Amygdala Activity in Patients with Major Depressive Disorder. PLoS One 9.

Zich, C., Debener, S., Kranczioch, C., Bleichner, M.G., Gutberlet, I., De Vos, M., 2015. Real-time EEG feedback during simultaneous EEG-fMRI identifies the cortical signature of motor imagery. NeuroImage 114, 438-447.

Zilverstand, A., Sorger, B., Zimmermann, J., Kaas, A., Goebel, R., 2014. Windowed correlation: a suitable tool for providing dynamic fMRI-based functional connectivity neurofeedback on task difficulty. PLoS One 9, e85929.

Zotev, V., Phillips, R., Yuan, H., Misaki, M., Bodurka, J., 2014. Self-regulation of human brain activity using simultaneous real-time fMRI and EEG neurofeedback. NeuroImage 85 Pt 3, 985-995.

Highlights:

- Development of an open-source Python/Matlab framework for real-time fMRI neurofeedback.
- Support of neurofeedback based on activity, connectivity and multivariate pattern analysis.
- Broad functionality, high modularity, and extendibility.
- Easy interfacing with functions of Statistical Parametric Mapping (SPM).
- Demonstration of functionality for three case studies.