How to Develop Standard WeBrain Tools

Li Dong: Lidong@uestc.edu.cn

Version: 1.1
Date: 3/26/2021

The Key Laboratory for NeuroInformation of Ministry of Education,
School of Life Science and Technology, University of Electronic Science and Technology of China, Chengdu, 610054, China

Content

1. WeBrain is powered by Docker

Considering the consistency and update of tools developed by different language (C/MATLAB/Python/JAVA etc) in different operating environments (Windows/Linux/Mac), in WeBrain system, it is suggested to package software into standardized units (container image) for development, shipment and deployment using container technique such as Docker (https://www.docker.com/). In WeBrain system, tools are standard container images that includes everything need to run; so that any tools can be integrated in WeBrain system, no matter what language or operating system used. Therefore, the developers can develop kinds of tools with WeBrain API using any experienced programming languages and environments, test it in the container using their own computers, and then release it (as well as tool instructions and examples) on the cloud so that it can be pulled by the WeBrain.

Docker is a technique designed to make it easier to create, deploy, and run applications by using containers. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. Docker containers that run on Docker Engine have following benefits:

Standard: Docker created the industry standard for containers, so they could be portable anywhere;

Lightweight: Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs;

Secure: Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry.

2. Application Programming Interface (API) Definition of WeBrain Tools

Furthermore, to largely decrease the requirements of IT skills and increase the friendliness of the WeBrain platform, a more flexible and conceptual application programming interface (API) of standardized units (container images) was designed and realized. This API is defined as a function with inputs and outputs, which is in line with most of tool usages on command window. The first 3 inputs (Input, Output and combs_project_id) of tools are fixed and required (The names should be 'Input', 'Output' and 'combs_project_id', respectively), and following any parameters can be defined by developers. The tool name is suggested to be 'wb_pipeline_*'.

The first input should be strings of data file directions, such as '~/zip_data/sub_01.zip,~zip_data/sub_03.zip'. It can be automatically generated in WeBrain system after selecting files. Noting that, data of each subject should be zipped as a separated zip file ONLY. For example, a zip file of EEG data (one subject) can be a 'Sub_01.zip', which contains all EEG files generated by EEG system (e.g. files of BrainProduct EEG system: sub_01.dat, sub_01.vhdr and sub_01.vmrk) or a folder consisting of the EEG files. **Surely, developers can define other kinds of data in the**

zip file.

The second input should be strings of output direction such as '~/EEG/results_temp'. It can be automatically generated in WeBrain system after selecting files.

The third input should be string of combs_project_id such as '1'. It is used to count the number of computing tasks created by developers in WeBrain. It is suggested to print/display the combs_project_id in the code ONLY.

The other inputs of parameters can be defined by developers, and they can be a file (supporting MATLAB *.mat and *.dat file; Excel *.xls and *.xlsx; ASCII *.txt file; and *.csv file ONLY) or a string (e.g. 'S12', '5','rest'…). **The maximal number of parameters allowed by WeBrain is 15!!!.**

The log could be printed on the command window, and then automatically captured by WeBrain system. **If the processing is completed or failed, the string of "********THE END********" must be printed, so that WeBrain system can capture the state of the processing of the created computing task.**

Warning:
DO NOT contain any blank spaces in all inputs or the input file names !!!

3. How to release tools on the WeBrain

For tool developers, there is no requirement of any IT skills and experiences such as k8s configuration, RESTful web service, dashboard implementation and computing resource scheduling etc., therefore, sharing and releasing tools on the WeBrain platform is easy (underlined)! There are 2 ways to share and release 2 kinds of tools, respectively:

- Offline tools

Offline tools are these toolkits which are developed by a language (e.g., C/MATLAB/Python/JAVA) in an operating environment (Windows/Linux/Mac). Users are always need to download them and implement in their own personal computers. Sharing these tools should:

[1] Prepare tools with example data, demos and tool instructions;

[2] Contact the WeBrain team (Lidong@uestc.edu.cn);

[3] Test and audit the tools by the WeBrain team;

[4] Release the tools on the WeBrain website (https://webrain.uestc.edu.cn/resources.html) and forum: (https://webrain.uestc.edu.cn/resources.html) by the WeBrain administrators;

[5] Users can download these tools from the WeBrain website.

- Tool images

Tool images are these toolkits/pipelines which have been standardized image containerized by the virtualization technique, and need to be added in the WeBrain system. Users can directly use these tools by logining their own accounts on the cloud. Sharing these tools should:

[1] Prepare the tool image with example data, demos and tool instructions;

[2] Release the tool image on a cloud repository (e.g. dockerhub) so that the

WeBrain administrator can pull the image from the repository;

[2] Contact the WeBrain team (Lidong@uestc.edu.cn);

[3] Test and audit the tool image by the WeBrain team;

[4] Add and release in the WeBrain system by the WeBrain administrators;

[5] Users can use these tools on the WeBrain website.

4. Tool Example

4.1 Docker example for API definition

Here is an example of Docker command of a tool (e.g. calculating power indices), which can be integrated in WeBrain system easily:

docker run -v /home/export:/export --net=host --privileged=true --rm=true -i docker.io/XXX/dockertest:latest /root/matlab_script/wb_pipeline_EEG_calcPower ~/export/data/sub_001.zip, ~/export/data/sub_002.zip, ~/export/result/ 1 10 255 [] [] 0 1:16 all []

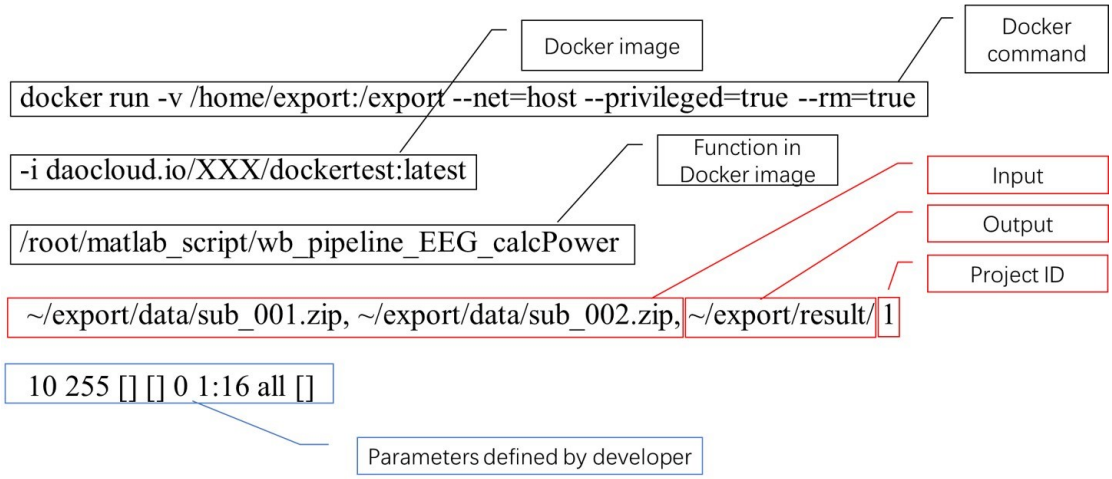

Fig. 1: A Docker example for API definition. The conceptual API is defined as a function with inputs and outputs, which is in line with most of tool usages on command window. An example of a Docker command of a WeBrain tool is showed. The first 3 inputs (Input, Output and combs_project_id) of tools are fixed and required (The names should be 'Input', 'Output' and 'combs_project_id', respectively, red box), and following any parameters (strings, blue box) can be defined by developers.

Noting that all inputs are strings in WeBrain system !!!

4.2 MATLAB example for API definition

As an example, a tool based on MATLAB in Linux OS is provided. The main MATLAB code in WeBrain is simple, and the inputs includes Input, Output, combs_project_id, parameter1,...parameterN (parameters). The first 3 inputs (Input, Output and combs_project_id) of tools are fixed and required, and following parameters could be defined by developers. Here is a demo of usage of MATLAB pipeline function:

```
% ----------------------
clear all;clc;
Input
='~\EEG\TEST_data\zip_data\sub_001.zip,~\EEG\TEST_data\zip_data\sub_002.zip';
Output = '~\EEG\results_temp';
combs_project_id = '1'; % project ID, print or display only.
epochLenth = '10'; % length of small epoch
eventlabel = 255; % event label in EEG data
bandLimit = '[]'; % specific frequency bands
bandName = '[]'; % names of frequency bands
proportion = '0'; % overlapped percentage for each segments.
seleChanns = '1:16'; % string with indices of the selected channels (re-referenced channels), or 'all'.
srate = '[]'; % sampling rate

wb_pipeline_EEG_calcPower(Input,Output,combs_project_id,epochLenth,eventlabel,
bandLimit,bandName,seleChanns,proportion,srate);
```

And then the MATLAB function 'wb_pipeline_EEG_calcPower.m' will be compiled into executable file on Linux system for deployment (mcc –m wb_pipeline_EEG_calcPower) and then dockerized.

4.3 MATLAB code template

Here I give an example of pipeline function based on MATLAB. And the key part is noted by red boxes. If have any questions, contact us as soon.

```
function wb_pipeline_template(Input,Output,combs_project_id,para1,para2,seleChanns,srate)
% ------------------------------------------------------------------------
% Instruction of pipeline tool
% ..........................
% step [1]
% step [2]
% ............

% References:
% Dong et al., XXXX, 2017
% ------------------------------------------------------------------------
% Input:
%     Input: zip paths of each subject (separate by commas).
%                     e.g. '*\sub_01.zip,*\sub_02.zip'. EEG data will be loaded as
%               EEG structure imported using EEGLAB. EEG.data should be channels
%               X time points OR channels X time points X epoches.
%       Output: output path.
%       combs_project_id: project ID (only print for WeBrain).
%       para1: XXXXXXXX
%       para2: XXXXXXXX.
%       ..............
```

```
%       seleChanns: string with indices of the selected channels
%                                   (e.g. '[1:4,7:30]'), or 'all'.
%       srate: sampling rate of EEG data. It can be automatically detected in
%              EEG data. But for ASCII/Float .txt File or MATLAB .mat File, user
%              should fill the sampling rate by hand. Default is '[]'.
%   Output:
%       EEG_results: results including power indices, mean indices across epoches
%              and parameters.
%                EEG_results.type: type of results;
%                EEG_results.XX: power acorss frequency bands;
%                EEG_results.XXX: relative power across frequency bands;

%                EEG_results.parameter.para1: XXXXX;
%                EEG_results.parameter.para2: XXXXX;
%                .....................................
% ------------------------------------------------------------------------
% Usage of function
% clear all;clc;
% Input = '~\zip_data\sub_001.zip,~\zip_data\sub_002.zip';
% Output = '~\EEG\results_temp';
% combs_project_id = '1';
% para1 = '5';
% para2 = '255';
% ........
% seleChanns = '[1:60]'; % string with indices of the selected channels
%                                       (re-referenced channels), or 'all'.
% srate = '[]';
% ------------------------------------------------------------------------
% Code Summary for working in School of Life Science and Technology,UESTC.
% Author: Li Dong, e-mail: Lidong@uestc.edu.cn
% This template is for non commercial use only.
% It is freeware but not in the public domain.

% Written by Li Dong (Lidong@uestc.edu.cn)
% $ 2019.9.18
% ------------------------------------------------------------------------
try
   wb_addpath();% add path
   disp('-------------------------------------------');
   disp(datestr(now));% computing date
   disp('Adding paths.....');
   % ---------------------
   % check all inputs
   if nargin < 3
      disp('********FAILED********');
      disp('3 inputs are required at least.');
      error('3 inputs are required at least.');
   elseif nargin == 3
      para1 = 5;
      para2 = [];
      seleChanns = 'all';
      srate = '[]';
   elseif nargin == 4
      para2 = [];
      seleChanns = 'all';
```

```
    srate = '[]';
elseif nargin == 5
    seleChanns = 'all';
    srate = '[]';
elseif nargin == 6
    srate = [];
end
% check inputs
if isequal(para2,'[]')
    para2 = [];
end

if isempty(para1) || isequal(para1,'[]')
    para1 = 5;
elseif ischar(para1)
    para1 = str2num(para1);    % string to number
end

if isempty(seleChanns)
    seleChanns = 'all';
end

% ----------
% display parameters
disp('Start Project......');
disp(['Project ID: ',num2str(combs_project_id)]);
% ------------------------------------------------------------------------
% loading data and calculating
% fprintf(fid,'loading data and calculating...,\r\n');
disp('loading data and calculating......');
files = regexp(Input,',','split'); % split strings by comma
Ns = length(files); % number of subjects
disp(['No. of subjects: ',num2str(Ns)]);
skipped_filename = [];
k = 1;
for s = 1:Ns
    [filepath,filename,~] = fileparts(files{s});
    disp('================');
    if ~isempty(filename)
        folders = regexp(filepath,filesep,'split');
        if length(filename)>33 && ~ismember('result',folders) % the files are not from the result folder.
            filename = filename(34:end); % if filename is too long, display last words only
        end
        disp(['calculating: ',filename]);
    else
        folders = regexp(files{s},filesep,'split');
        if isempty(folders{length(folders)})
            filename = folders{length(folders)-1};
        else
            filename = folders{length(folders)};
        end

        if length(filename)>33 && ~ismember('result',folders) % the files are not from the result folder.
            filename = filename(34:end); % if filename is too long, display last words only
        end
        disp(['calculating: ',filename]);
    end
    % --------------------
```

```
        try
            % loading data
            EEG = wb_loadEEG(files{s},Output); % load EEG data
            Nchanns = size(EEG.data,1); % No. of channels in EEG
            % -------------------
            % check sampling rate
            if ~isequal(srate,'[]')
                srate1 = str2num(srate);
                if isfinite(srate1) && srate1 > 0
                    EEG.srate = srate1;
                end
            end

            % check refchan
            if isequal(seleChanns,'all')
                selchan = 1:Nchanns;
            else
                selchan = str2num(seleChanns);
            end

            if isnumeric(selchan) && all(isfinite(selchan)) && ~isempty(selchan) && max(selchan) <=
Nchanns % is numeric array?
                disp(['No. of channels: ',num2str(Nchanns)]);
                disp(['No. of selected channels: ',num2str(length(selchan))]);
                disp(['Selected channels: ',num2str(selchan)]);
            else
                % disp('********FAILED********');
                disp('selchan is invalid.');
                error('selchan is invalid.');
            end
            % -------------------------
            try
                % calcualte XXXX indices
                EEG_results = wb_calculate_XXXX(EEG,para1,para2,selchan); % If the EEG data is
imported by EEGLAB, you could edit this function wb_calculate_XXXX.
                % now the webrain can load several format of EEG data, and
                % also save results or data.
                % -------------------
                % saving results
                disp('saving results...');
                output_temp = fullfile(Output,['power_',filename]);
                save(output_temp,'EEG_results');

            catch
                warning('Failed to calculate XXX indices');
                disp(['Skip: ',filename]);
                skipped_filename{k} = filename;
                k = k+1;
            end;
        catch
            warning('Failed to load EEG data');
            disp(['Skip: ',filename]);
            skipped_filename{k} = filename;
            k = k+1;
        end
    end

    disp('------------------------')
```

```
    if isempty(skipped_filename)
        disp('********SUCCESS********');
    else
        disp('********CalculatedSubjects********');
        disp(['No. of Calculated Subjects:',num2str(Ns-length(skipped_filename))]);
        disp('********SkippedSubjects********');
        disp(['No. of Skipped Subjects:',num2str(length(skipped_filename))]);
        for k1 = 1:length(skipped_filename)
            disp(skipped_filename{k1});
        end
        disp('********SUCCESS********');
    end
catch
    disp('********FAILED********');
end;
disp('********THE END********');
```

5. Resources

- How to use Docker (Chinese)

https://www.widuu.com/docker/installation/windows.html
https://www.runoob.com/docker/docker-install-nginx.html

- How to use Docker (English)

https://www.docker.com/products/docker-desktop
https://docs.docker.com/get-started/

6. Copyright: