Code demonstration

Title: A neural network trained by 3D printed biomimetic cochleae provides clinical informatics of cochlear implant patients

Authors

Iek Man Lei [1,2], Chen Jiang[3,1], Chon Lok Lei[4], Simone Rosalie de Rijk[3], Yu Chuen Tam[5], Michael P. F. Sutcliffe[1], George G. Malliaras[1], Manohar Bance[3,*], Yan Yan Shery Huang[1,2,*,#]

Affiliations

[1]*Department of Engineering, University of Cambridge, United Kingdom*

[2]*The Nanoscience Centre, University of Cambridge, United Kingdom*

[3]*Department of Clinical Neurosciences, University of Cambridge, United Kingdom*

[4]*Department of Computer Science, University of Oxford, United Kingdom*

[5]*Emmeline Centre for Hearing Implants, Addenbrookes Hospital, Cambridge, United Kingdom*

*Corresponding authors: mlb59@cam.ac.uk, yysh2@cam.ac.uk

[#]Lead contact: yysh2@cam.ac.uk

1. Installation

This document provides instructions to use the code with windows 10. It contains instructions to install the code, train a neural network model, and perform forward and inverse predictions. The command could be slightly different on Linux or Mac or on a different python IDE. If python and git are not installed on your device, install [anaconda (Python 3.7.6 -64bit)](#) and [git](#). [Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019](#) is also required for running tensorflow.

Other versions of python might have compatibility issue with tensorflow, run the following to change the python version to 3.7.6 on Anaconda prompt if other versions are installed.

```
conda install python=3.7.6
```

The code requires Python (3.7.6) and the following dependencies: [PINTS](#), [scikit-learn](#), [joblib](#), [tensorflow (2.1.0)](#). It also needs [seaborn](#) for plotting. To install the code and the relevant dependencies, unzip the folder, launch anaconda prompt, navigate to the path of the folder, and run the following:

```
pip install --upgrade pip --user
pip install .
```


'Successfully installed….' will be printed when it is done. The total install time is around 20min on a normal computer. All the expected output in this demo can be found in 'demo_expected_output' folder.

2. Create NN model using experimental EFI data

Demo To generate the NN model using the experimental EFI data obtained in this study, launch Spyder (Spyder IDE is installed by default with the Anaconda package) or any python IDE, navigate to the path of the main folder, and run the following line.

```
%run fit-nn.py EFI_training_data.txt
```

```
In [3]: %run fit-nn.py EFI_training_data.txt
Fit seed:  542811797
Model: "sequential_1"

Layer (type)              Output Shape            Param #
=================================================================
dense_3 (Dense)           (None, 32)              256

dense_4 (Dense)           (None, 32)              1056

dense_5 (Dense)           (None, 1)               33
=================================================================
Total params: 1,345
Trainable params: 1,345
Non-trainable params: 0

Training the neural network for all stimuli...
Done. NN model is saved.
```

The run time is around 30min. A NN model file 'nn-EFI_training_data-stim_all.h5' will be saved in 'out-nn' folder after running.

Details Run fit-nn.py with argument [str:input_file.txt]. input_file.txt containing a list of file IDs as training data. The file IDs of our training dataset are listed in EFI_training_data.txt, which contains 82 experimental file IDs of our 3D printed biomimetic cochleae. All experimental EFI data are stored in 'data' folder, and the input parameters (model descriptors) of each experimental EFI data are stored in 'input' folder with the same file ID as the EFI data. The format of the inputs should be listed in a column format [p0; p1; p2; p3; p4], where p0=basal lumen diameter(mm), p1=infill density(%), p2=taper ratio, p3=cochlear width (mm) and p4=cochlear height (mm). For more instructions, see 'README.txt' in 'input' folder. 'available-electrodes.csv' in 'data' folder specifies the available electrodes in each EFI measurement, with '1'=available and '0'=unavailable. By default, position information of Advanced Bionics 1J electrode array is used here. Electrode position can be specified in line 76 in fit-nn.py if other cochlear implant is used;

3. Perform forward predictions with the trained NN model

Demo predict-nn.py performs predictions of EFI based on input parameters. The trained NN model created in the above step with the name 'EFI_training_data' is used in this demo. A txt file 'demo_forward_predict.txt' can be found in the main folder, which contains two predict_ids (101315 & 160340) for EFI predictions. Run the following line:

```
%run predict-nn.py EFI_training_data demo_forward_predict.txt
```

```
In [4]: %run predict-nn.py EFI_training_data demo_forward_predict.txt
Fit seed:  542811797
Loading trained Neural Network models...
Predicting 101315 ...
Running validation...
Predicting 160340 ...
Running validation...
```

The run time is around 5s for each prediction. Predicted EFIs with the file name ('id_101315-efi.csv' and 'id_160340-efi.csv') can be found in './out-nn/EFI-training_data-predict' folder. 'id_101315-simple-plot.png' and 'id_160340-simple-plot.png' compare the predicted EFI and the experimental EFI.

Run `predict-nn.py`, with argument `[str:nn_name]` and `[str:predict_ids.txt]`. `'nn_name'` is the name of the trained NN model, which is `EFI_training_data` in this demo. `'predict_ids.txt'` contains a list of file IDs (`predict_ids`) for prediction. The input parameters of the prediction file IDs should be stored in the `'input'` folder. Again, by default, position information of 1J electrode array is used here. Electrode position can be specified in line 82-85 in `predict-nn.py` if other cochlear implant is used. `'id_predict_id-efi.csv'`, which contains the EFI profile, will be saved in `'./out-nn/nn_name-predict'`

4. Perform inverse predictions with the trained NN model

Demo `invabc-nn.py` predicts a posterior distribution of the input parameters (model descriptors) that satisfies a threshold defined by the user best-fitting the EFI data. It allow predictions when all input parameters are unknown (4.1) or when some of the input parameters are known (4.2). Again, the trained NN model `'EFI_training_data'` is used here. `demo_forward_predict.txt` contains a slimJ patient file ID (`patient_slimj1`) for this inverse prediction demonstration.

4.1 Prediction of input parameters with unknown cochlear geometrical features and resistivity

Demo Run the following line. A final MAPE of 7% is used here (this is set in the code already).

```
%run invabc-nn.py EFI_training_data demo_invabc_predict.txt
```

```
In [32]: %run invabc-nn.py EFI_training_data demo_invabc_predict.txt
Fit seed:  542811797
Loading trained Neural Network models...
Predicting Patient_slimj1 ...
Summary statistics at guess input parameters:  0.4918711139638255
Summary statistics at true input parameters:  0.07534839396226625
Using ABC-SMC
Running in sequential mode.
Trying t=1, threshold=0.5
Trying t=2, threshold=0.2
Trying t=3, threshold=0.15
Trying t=4, threshold=0.1
Trying t=5, threshold=0.09
Trying t=6, threshold=0.08
Trying t=7, threshold=0.07
Iter. Eval. Acceptance rate Time m:s
1     5251   0.000190439916  28:50.9
2     5268   0.000379650721  28:56.3
3     5269   0.000569368002  28:56.6
```

'Done' will be printed to the console when it is finished. The run time is around 2hrs for this prediction. The outputs are saved in `'./out-nn/EFI_training_data-inv-predict'` folder. The 1000 combinations of the predicted input parameters are listed in `'id_Patient_slimj1-samples.csv'`, with p1 (infill density) converted to resistivity.

4.2 Prediction of resistivity (infill density) with known cochlear geometrical features

Demo Specify the fixed parameters in 'fix_param.py'. For instance, patient_slimj1 has cochlear geometric features of basal lumen diameter (p0) = 2.34mm, taper ratio (p2) = 0.88, cochlear width (p3) = 8.2mm and cochlear height (p4) = 4.5mm. The fixed values are specified as below in 'fix_param.py' with None in p1 because infill density (resistivity) is unknown.

```
fix_input = {#parameter index: fixed value,
    0: 2.34,   # Parameter 1 - Basal lumen diameter; None if not fixed.
    1: None,   # Parameter 2 - Infill density of the matrix; None if not fixed.
    2: 0.88,   # Parameter 3 - Taper ratio; None if not fixed.
    3: 8.2,    # Parameter 4 - Cochlear width; None if not fixed..
    4: 4.5,    # Parameter 5 - Cochlear height; None if not fixed.
}
```

Similar to 4.1, a MAPE of 7% is used (defined in the code already) and run the following line:

```
%run invabc-nn.py EFI_training_data demo_invabc_predict.txt
```

```
In [31]: %run invabc-nn.py EFI_training_data demo_invabc_predict.txt
Fit seed:  542811797
Loading trained Neural Network models...
Predicting Patient_slimj1 ...
Summary statistics at guess input parameters:  0.2683008888973653
Summary statistics at true input parameters:  0.07785445356587356
Using ABC-SMC
Running in sequential mode.
Trying t=1, threshold=0.5
Trying t=2, threshold=0.2
Trying t=3, threshold=0.15
Trying t=4, threshold=0.1
Trying t=5, threshold=0.09
Trying t=6, threshold=0.08
Trying t=7, threshold=0.07
Iter. Eval. Acceptance rate Time m:s
1      968    0.00103305785    5:18.1
2      972    0.00205761317    5:19.4
3      974    0.00308008214    5:20.0
```

The run time is around 1hr for this prediction. Again, the outputs are saved in './out-nn/EFI_training_data-inv-predict' folder. The 1000 predicted values of p1 (infill density) and the converted resistivity can be found in 'id_Patient_slimj1-samples.csv'.

Details Run invabc-nn.py, with argument [str:nn_name] and [str:predict_ids.txt]. 'nn_name' is the name of the trained NN model, which is EFI_training_data in this demo, and 'predict_ids.txt' contains a list of file IDs for prediction. The EFI profiles of the predict_ids are stored in the 'data' folder. MAPE (mean absolute percentage error) threshold is defined in line 240 in invabc-nn.py. By default, a final MAPE threshold of 7% (0.07) is used, and 1000 samples are drawn from the final posterior distribution to approximate the distribution.