Project documentation

Categorical and binary classification in Visual object recognition task with dense neural network

JiHoon Kim[a]*

[a]Freie Universität Berlin, Berlin 14195, Germany

A B S T R A C T

As a programming project, this research focus on design a program for classification fMRI data. The Haxby dataset (2001) was used, and simple artificial neural network (ANN) was designed for categorical and binary classification. The module contained data acquisition, data preprocessing of fMRI data, neural network data preparation, grid search, cross validation, train and test categorical classification model, data visualization, binary classification model, and simple interface to interact with user. The result showed that preformed over chance level in both classifications. However, detecting the activation area remained as a further study.

Keywords: Nibabel, Nipype, Nilearn, Keras, fMRI

1. Background

This is project documentation for neurocognitive methods and programming project on summer semester 2020/2020 in freie Universität berlin. This project focused on classifier problem using neural network on fMRI data.

Multivariate pattern analysis (MVPA) of fMRI data analyze neural responses as patterns of activity. In a visual object recognition decoding task, machine learning has used as a classifier. Representatively, there are support vector machines (SVM), logistic regression, and ridge regression

Using artificial neural network (ANN), unrevealed information, which is hard to find, has been discovered nowadays. This research focused on the classification of face and 7 different object categories stimulus. In this time, I applied simple ANN than SVM for analyzing fMRI data.

2. Data and methods

2.1. Haxby et al. (2001) dataset

Haxby et al. (2001) dataset is a functional Magnetic Resonance Imaging (fMRI) dataset of a block-design fMRI study on face and objects representation in human ventral temporal cortex (VT). According to explanation of the open source website, “It consists of 6 subjects with 12 runs per subject. In each run, the subjects passively viewed greyscale images of eight object categories, grouped in 24s blocks separated by rest periods”.

This dataset contains anat.nii.gz, bold.nii.gz, mask.nii.gz and labels.txt per subject. anat.nii.gz has a high-resolution anatomical image. But for subject 6, there is no anatomical image. bold.nii.gz is a 4-dimension fMRI timeseries image. (1452 volumes with 40 x 64 x 64 voxels, corresponding to a voxel size of 3.5 x 3.75 x 3.75 mm). mask.nii.gz provide “VT”, “face” and “house” masks of research of interest (ROI) mask. And labels.txt have a stimulation task condition and the fMRI image volume order in the timeseries in a tow-column text file.

2.1.1. ROI based VT mask image

In this project, all subject’s fMRI image used. For label, I converted labels.txt file to .csv file. And I used the provided ROI mask.nii.gz based on VT. Compared to the entire brain masked image, this helps to reduce the dimensionality of data (Figure 1).

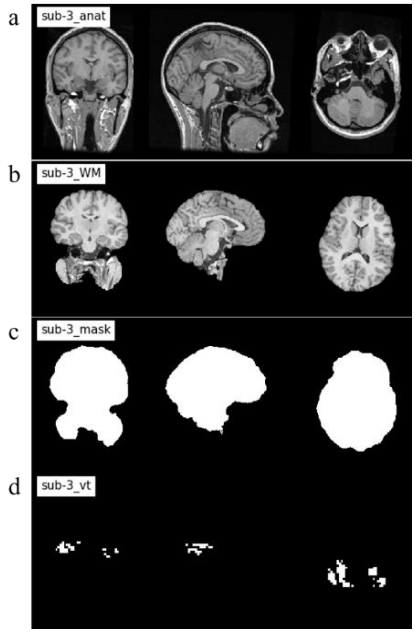

Fig. 1 – Subject 3 images. (a) anatomical image; (b) only for brain area image; (c) whole brain mask image; (d) ventral temporal mask image.

2.2. Dense neural network

A dense neural network (DNN) or fully connected neural network with three dense layers and two dropout layers were used to classify the object stimuli in the fMRI study. The structure of the DNN is shown in Fig 2.

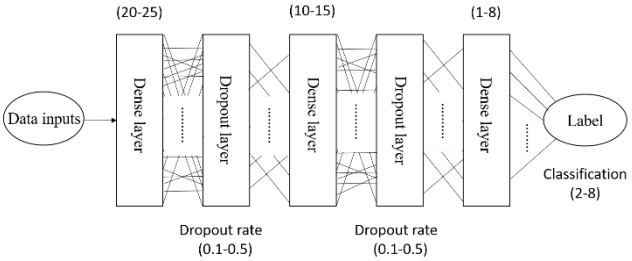

Fig. 2 – Schema of dense neural network model. The two layers, the dense layer and the dropout layer, appear alternately.

At the first dense layer, dimensionality of the output space is either 20 or 25. The next dense layer, it is either 10 or 15. At last dense layer is either 1 or 8 depends on binary classification or categorical classification, respectively. In the between dense layers, there are dropout layer which rate 0.1, 0.3, and 0.5. The appropriate learning model parameters for each participant's data were determined through grid search.

The accuracy of the DNN was determined by a cross validation method. Whole data divided training set, validation set, and test set as 4:1:1 ratio. K=5 cross-validation (CV) was performed to determine the appropriate epochs then train and test the model.

2.3. Libraries

All these modeling processes were provided by Python libraries (Abraham, A. et al. (2014), Chollet, F,et al. (2015), Gorgolewski K. et al. (2011)); nibabel, nipype, nilearn, numpy, pandas, matplotlib, scikit-learn, tensorflow, and keras. I used a laptop computer with CPU of Intel® Core™ i5-8265U CPU and 8GB RAM in Windows 10 Home. It was run in a docker environment which can access an image, "jihoonkim2100/nnpp:init" that can be loaded and used in the docker hub (Figure 3). It tested on the shell and jupyter notebook.

Environment	Version
Operating System	Window 10 Home
Virtual environment	Docker: jihoonkim2100/nnpp:init
Programming Language	Python 3.6.10
Libraries	Nibabel 3.1.1
	Nipype 1.6.0 dev0
	Nilearn 0.6.2
	Numpy 1.18.5
	Pandas 1.0.5
	Scikit-learn 0.23.1
	Tensoflow-cpu 1.13.1
	Keras 2.2.4
	Matplotlib 3.2.2
	Jupyter notebook 6.0.3

Fig. 3 – Implementation environment.

3. Research

3.1. System design

This model analysis the data of one subject in a whole run which shown in Figure 4. In the view of user perspective, user send an input, subject number among subject 1 to 6, to system. And, during categorical classification, user give inputs about hyper-parameter based on grid search result from system. At last, type the same subject for data visualization and binary classification of 'face' vs. 'house'. On the binary classification, there are no grid search and cross validation, and it trained with fixed parameter and evaluate the train set.

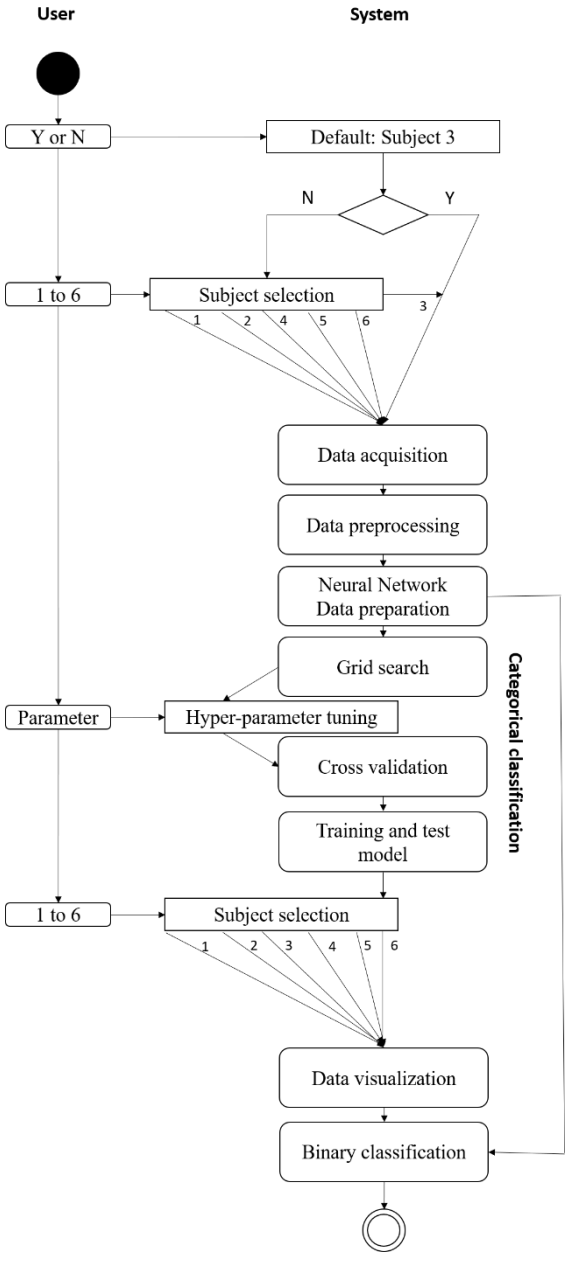

Fig. 4 – Workflow of classification with dense neural network model.

3.2. Implementation

There are five classes: DataAcquisition, DataPreprocessing, NN, Extra, and Interface which shown in Figure 5. First, DataAcquisition contain, newfolder, inputd, outputd, and reorder functions for data acquisition and reordering image as an example. Second, DataPreprocessing have mask_image, mcflirt, and nifti-masker functions for preprocessing of fMRI data. Third, NN get categorical_label, categorical_dataset, grid_search, cross_validation, and test_model functions to do categorical classification using neural network. Fourth, Extra do data visualization and binary classification using anat_mask, func_img, binary_class-ificaiton. Last, interface have preprocessing_interface ,hyperparmeter-_interface, and extra_image_inter-face for interacting with user.

Class	Function
DataAcquisition	setdata : initial setting newfolder : create a new folder inputd : load the data outputd : save the data reorder : reorder the data acquisition : do above functions
DataPreprocessing	mask_image : create a mask mcflirt : do motion correction niftimakser : preprocess smoothing, normalization, with VT mask.
NN	categorical_label : load the label categorical_dataset : data preparation grid_search : do grid search gridsearch_model : grid search model cross_validation : do cross validation cross_validation_model : CV model test_model : train and evaluate model build_model : Final model
Extra	anat_mask : show mask images func_img : show functional images bianry_classification : do binary classification on 'face' vs 'house'.
Interface	preprocessing_interface : get an input from user and proceed preprocessing hyperparameter_interface : receive inputs about the hyperparameter for CV and evaluate model from user extra_image_interface : generate images from input, subject number

Fig. 5 – Implementation of classes.

3.3. Results

Data preprocessing step, fMRI data return as 2d array data type. Each subject data type is different based on the VT mask image. Six fMRI data, fMRI size convert (40,64,64,1452) to (577,1452), (464,1452), (307,1452), (675,1452), (422, 1452), and (348, 1452) to each subject 1 to 6.

In neural network preparation, rest stimuli removed: (577,864), (464,864), (307,864), (675,864), (422, 792), and (348, 864). Except subject 5, all subject contains 108 images of eight object stimuli.

As described above, dataset was divided into two sub-datasets: as an example of subject 3, 720 for training data, and 144 for testing data. Among training data, 80% of the training set data were used for training stage, and the other 20% of the training data were used for validation using K=5 CV.

Before doing CV, grid search test the batch size -25, 50-, dropout rate -0.1, 0.3, and 0.5-, epochs -100, 200-, first dense layer dimension -20, 25-, and second dense layer dimension -10, 15-. And the best parameter results of each subjects are shown in Table 1.

Table 1 – Best grid search on each subject. ACC; Accuracy, N1; First dense layer, N2; Second dense layer.

Subject	ACC	Batch size	Dropout rate	Epochs	N1	N2
1	0.611111	50	0.3	200	20	10
2	0.511111	50	0.3	100	20	15
3	0.638889	25	0.3	200	20	15
4	0.533333	50	0.1	200	20	10
5	0.654545	25	0.1	200	25	10
6	0.748611	25	0.1	200	20	10

Figure 6. showed that CV of each fold and average training and validation of accuracy and loss of subject 3. In average accuracy indicated that overfitting happened around 25 to 50 epochs, however, minimize loss point is steady around 50 to 200 (Figure 6).

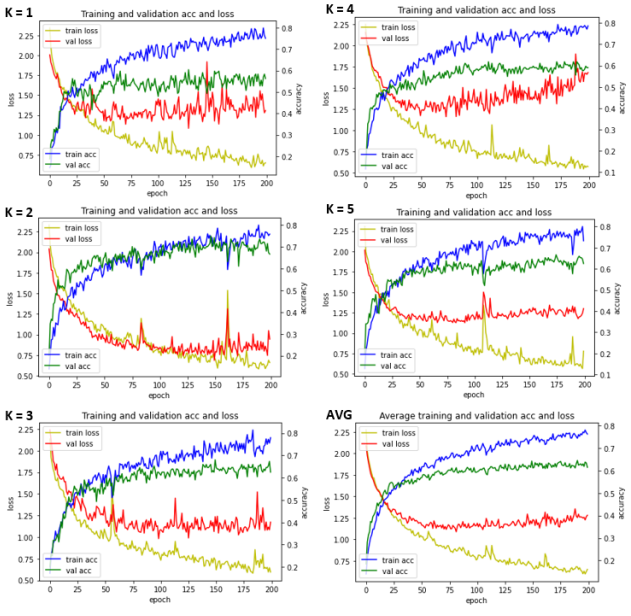

Fig. 6 – Cross validation training and validation of acc and loss results. It shows each fold and average training and validation of acc and loss.

I trained with train data and evaluated the model with test data in different epochs: 25, 50, 75, 100, 200 with same parameter of CV. As shown in Table 2, the result of average acc per subject shows 44.16 % to 68.05 % which exceed the chance level 12.5 %.

Table 2 – Average accuracy of evaluation of test data.

Subject	Average accuracy	Highest accuracy, epochs	Lowest accuracy, epochs
1	62.64 %	63.19 % 25-75	61.11 %, 100
2	55.83 %	65.97 %, 200	40.97 %, 25
3	68.05 %	73.61 %, 200	55.56 %, 25
4	44.16 %	50 %, 200	39.58 %, 25
5	53.73 %	58.33 %, 200	51.25 %, 25
6	59.58 %	64.28 %, 75	52.08 %, 25

In analysis of binary classification of face vs. house, the dataset divided by three subtests: train data, validation data, and test data as 144, 36, 36. And the dropout rate was 0,3 and dimensionality of output dense layers are 25, and 15. The model trained 18 batch size in 10 epochs, and evaluated accuracy showed the range of 77.78% to 100% on subjects (Figure 7).

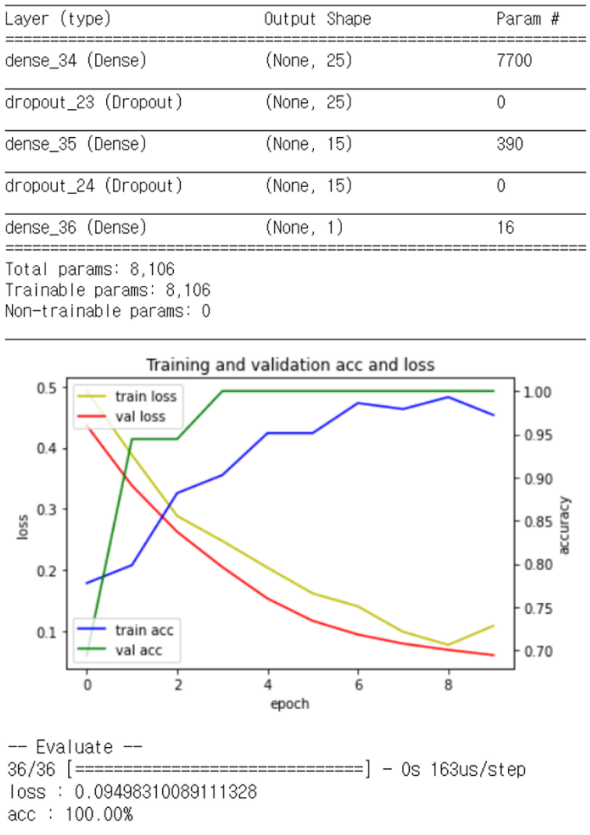

Fig. 7 – Architecture and result of binary classification of subject 3

4. Discussion

In this project, I tested the DNN method as a classifier of object recognition task stimuli fMRI data. The results suggest that it worked but could not exceed more than 80 % accuracy. For solving this problem, more dataset and more elaborate hyperparameter tuning is needed.

Nevertheless, compare with Nilearn examples, it shows better result on accuracy both categorical classification and binary classification. As an example, decoding with SVM: face vs house in the Haxby dataset showed the 70.37 % classification accuracy in subject 3. In my model, it showed at least 100 %.

However, this model could not show the activation map of the location to determine the critical point, so further research needed. And using convolutional neural network as an 3D voxel images analysis might be more effective than DNN. For that process, generating the integrated ROI based on the anatomy template of VT mask is needed.

In summary, I preprocessed the fMRI data, designed the model for classification using dense neural network, and test the model. The results showed higher accuracy than chance level in both categorical and binary classification.

5. Open Access

5.1. Copyright

The original authors of Haxby et al. (2001) hold the copyright of this dataset and made in available under the terms of the Creative Commons Attribution-Share Alike 3.0 license. The dataset available from the (http://www.pymvpa.org/datadb/haxby2001.html)

5.2. Code Availability Statement

The source code is available from the OSF (https://osf.io/qu3y7/), github (https://github.com/jihoonkim2100/NNPP), and docker hub website (https://hub.docker.com/repository/docker/jihoonkim2100/nnpp).

References

Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., Gramfort, A., Thirion, B., & Varoquaux, G. (2014). Machine learning for neuroimaging with scikit-learn. Frontiers in neuroinformatics, 8, 14. https://doi.org/10.3389/fninf.2014.00014

Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

Gorgolewski K, Burns CD, Madison C, Clark D, Halchenko YO, Waskom ML, Ghosh SS. (2011). Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in Python. Front. Neuroimform. 5:13.

Haxby, J., Gobbini, M., Furey, M., Ishai, A., Schouten, J., and Pietrini, P. (2001). Distributed and overlapping representations of faces and objects in ventral temporal cortex. Science 293, 2425–2430.