# ‘Alcoholic’ Classification with multiple models on EEG[1] images and features

Peng Zhang

Research School of Computer Science, Australian National University, Australia
u6921163@anu.edu.au

**Abstract.** With the penetration of deep learning method in many data intensive fields, it helps solving multiple types of prediction or decision task, of which biomedicine is also a hot and important aspect. This paper describes deep learning solutions on ‘alcoholic’ binary classification task from EEG dataset that is collected from number of electroencephalography sensors around subjects’ brain. The specific solutions are implemented with technics including deep neural network, bidirectional network and convolutional network. This paper provides the whole observation of process from design, implementation, model tuning all the way to validation. Also, it gives an analysis and insights about the results based on comparison.

**Keywords:** EEG, BDNN, DNN, CNN, deep learning bidirectional network, convolutional network

## 1 Introduction

With the rapid development of User Interface and User Interaction, the deep learning in brain activity field has become a hot topic. Meanwhile, using deep learning method upon this area also provides an extra tool to biomedicine. Our EEG dataset in this paper is kind of signal showing the electrical activity detected on the surface of brain. Similar with the usage of deep learning in other science subjects, here deep learning is also used for doing some transformation and to dig some insights while training a network or building a model. Thus, we implemented three mainstream types of network structures to search on this task.

## 2 Dataset

*In this paper, we use both uci-eeg-features and uci-eeg-images two versions of eeg dataset.*

### 2.1 Observation on dataset

Firstly, let’s have a simple view at the EEG dataset. Here I am using the ‘load_mat’ method from ‘scipy’ package to read in the ‘ .mat’ file. There are two version of features given, one is compressed version, of dimension 192 per sample, another is the extended version containing features of 32*32*3 each.

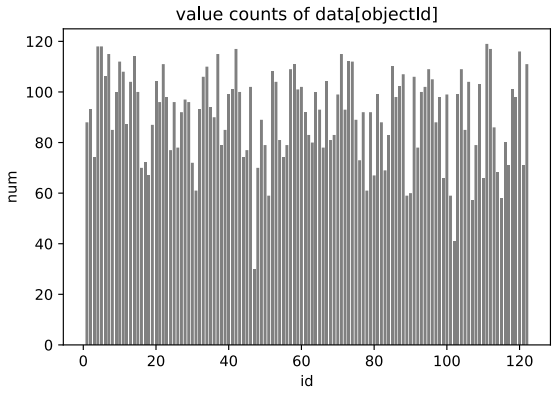
**Table 1.** Dictionary type content read from .mat file.

| ***Key*** | **Value** | **Shape** | **Range** |
|---|---|---|---|
| ***__header*** | MATLAB 5.0 MAT-file | | |
| ***__version*** | 1.0 | | |
| ***__globals*** | [ ] | | |
| ***Data*** | np.ndarray | (11057,192) | 0.0~18.83 |

| | | | |
|---|---|---|---|
| ***Extended data*** | np.ndarray | (11057,32,32,3) | 0.0~18.55 |
| ***y_alcoholic*** | np.ndarray | (1,11057) | 0/1 |
| ***y_stimulus*** | np.ndarray | (1,11057) | 1~5 |
| ***Subjected*** | np.ndarray | (1,11057) | 1~122 |
| ***trialnum*** | np.ndarray | (1,11057) | 1~119 |

As we could see, the data with shape (11057,192) is the 192 features of 11057 samples, 192 features include signals from 64 sensors of 3 frequency bands. And we got two y labels, one called 'y_alcoholic' is for the target of this task which means whether the object is alcoholic, another 'y_stimulus' is a multi-class label. Besides, there is also a extended version of feature containing a more informative data of 32*32 image in 3 frequency bands.

**Figure 1.** data distribution on 122 subjects




This **figure 1** shows another fact of this data set, which is that the 11057 samples are collected from 122 subjects, figure 1 shows the distribution of samples on 122 subjects.

### 2.2 Data Preprocess

1.Based on observation above, since we are not using multiple types of features, so a **normalization** operation is not necessary, and we tried do **data balancing** operation, the change is slight.
2. In the following process, we tried to build a **new-defined 10-classes label** to enhance the backward training process of BDNN:

**Formula 1.** 10-classes label

$$y\_data = (y1\_alcoholic * 5 + y2\_stimulus - 1)$$

This formula embeds two types of labels with separately 2 and 5 classes in to an integrated 10-classes label, which is going to be used in the enhanced version of BDNN.

## 3 Methodology

The big picture of methodology in this paper is based on deep learning, giving a research on deep learning solution upon alcoholic classification task.

*The three network structures or technics we use in this paper:*

1. Deep neural network
2. Bidirectional neural network
3. Convolutional neural network

## 3 Design and implement 1 (traditional BP neural network)

As going through the dataset, I decide to initially build a simple network using only two linear layers with sigmoid as activation function.

### 3.1 Neural network structure

**Figure 2.** DNN structure

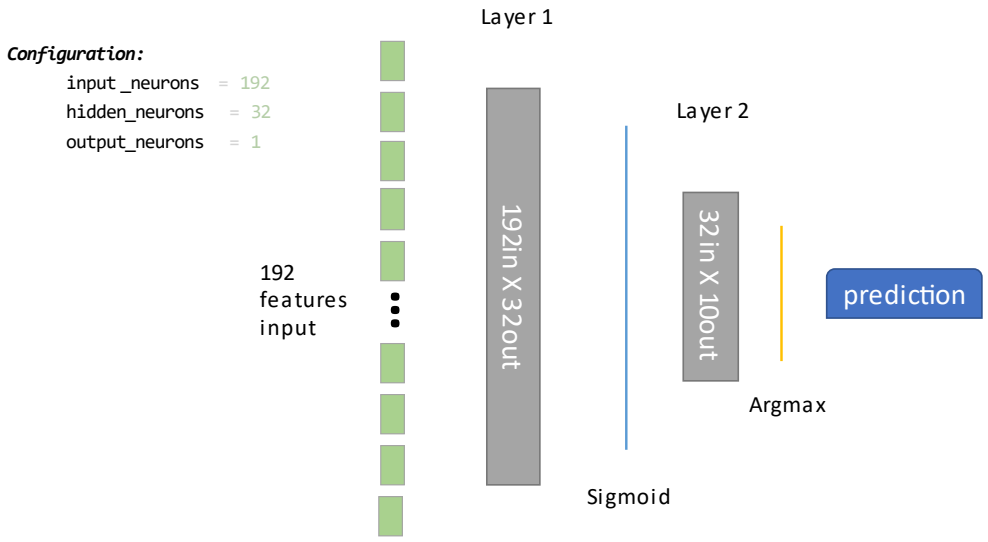



***Figure 1***

### 3.2 Neural network tuning

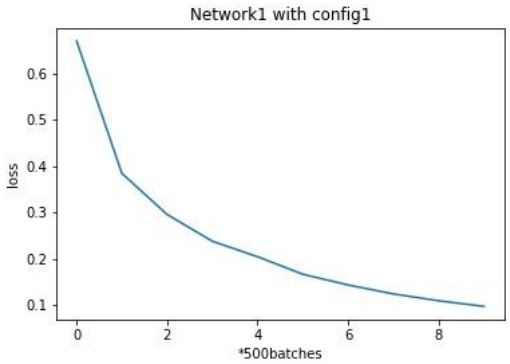
Before we start training, we firstly do train and test set split, we randomly take 20% of whole dataset of length 11057 as the test set, and to eliminate the effects caused by the randomness of data split while training, we set the random state of split function with a fixed seed value.

The parameters we have tuned include learning rate, number of epochs, batch size. Also, several loss functions and optimizers are tried to reach a considerable performance. Here is the final setting of the task.

**Table 2.** DNN setting

| num_epochs | batch size | learning rate | loss function | optimizer |
|---|---|---|---|---|
| **5000** | length of set | 0.01 | binary cross entropy | adam |

**Figure 3.** Loss descent




Finally, this model with configuration above finally reaches a testing accuracy at **0.92**, we can tell that it is a very high score, next step we are going to try a BDNN technic.

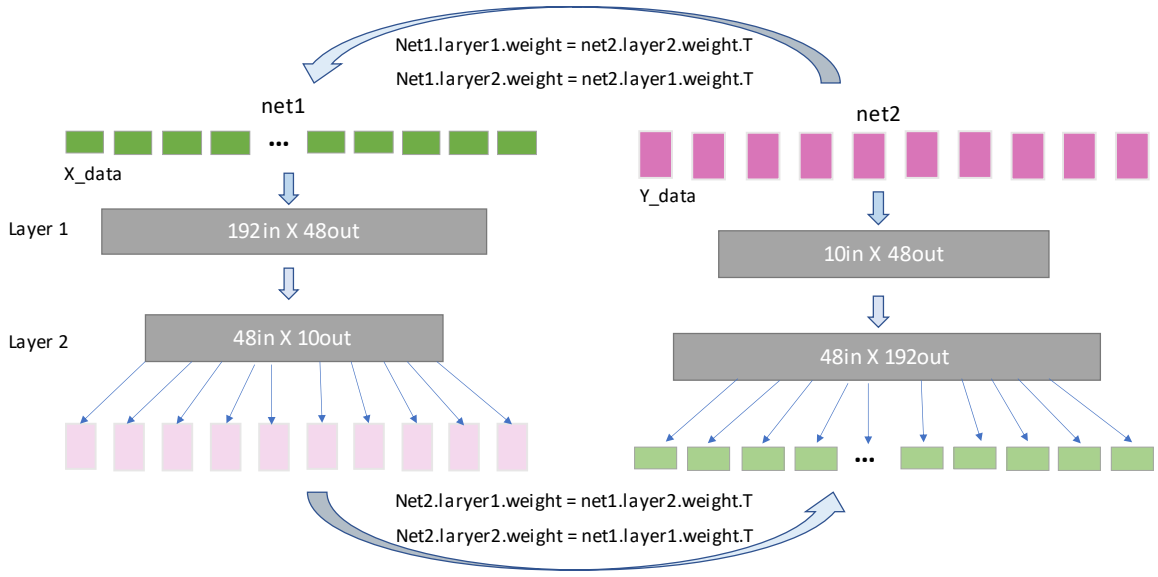
# 4 Design and implement 3 (bi-directional neural network[2])

## 4.1 Neural network structure and training process

This alcoholism dataset shows a classification work, binary one for the y label 'alcoholic', multi-class one for the y label 'stimulus'. So, before the training or model building, we could directly have a clear view that the backward result would be one representative 192 vector corresponding to a class. In this attempt, I tried BDNN on pure binary y label, the result is very low, it only reaches an accuracy at **0.53**, which is similar to the performance of blind guessing. This is because the binary output keeps too few features [6] for predicting the 192 features backward. So, I decided to involve the 'stimulus' y label to help the backward training process.

To implement the forward and backward training process, I created two networks with same weight [5] value, but the dimensions of each layer are inverse. Purpose of doing this is that we could control the interval of changing training direction, in other words, if interval value is x, then we do forward x times and backward x times circularly.

**Figure 4.** Training process of BDNN[2]

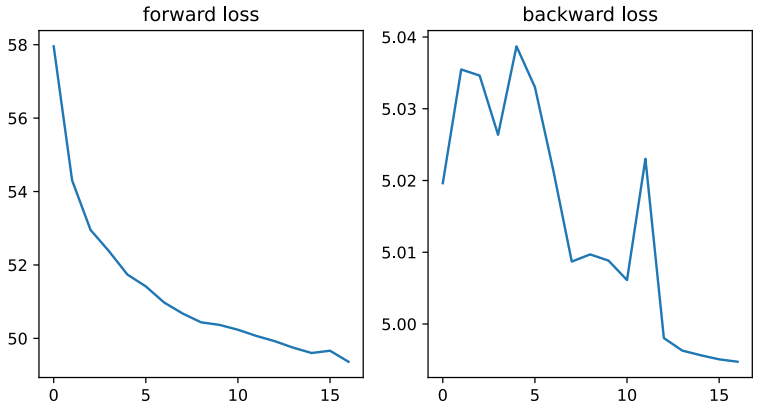



### 4.2 Model tuning

**Table 3.** Final BDNN setting

| Net | num_epochs | batch size | learning rate | loss function | optimizer |
|---|---|---|---|---|---|
| **forward** | 100/2 | 16 | 0.01 | cross entropy | Adam[7] |
| **backaward** | 100/2 | 16 | 0.003 | MSE | RMSprop |

**Figure 5.** Loss descent

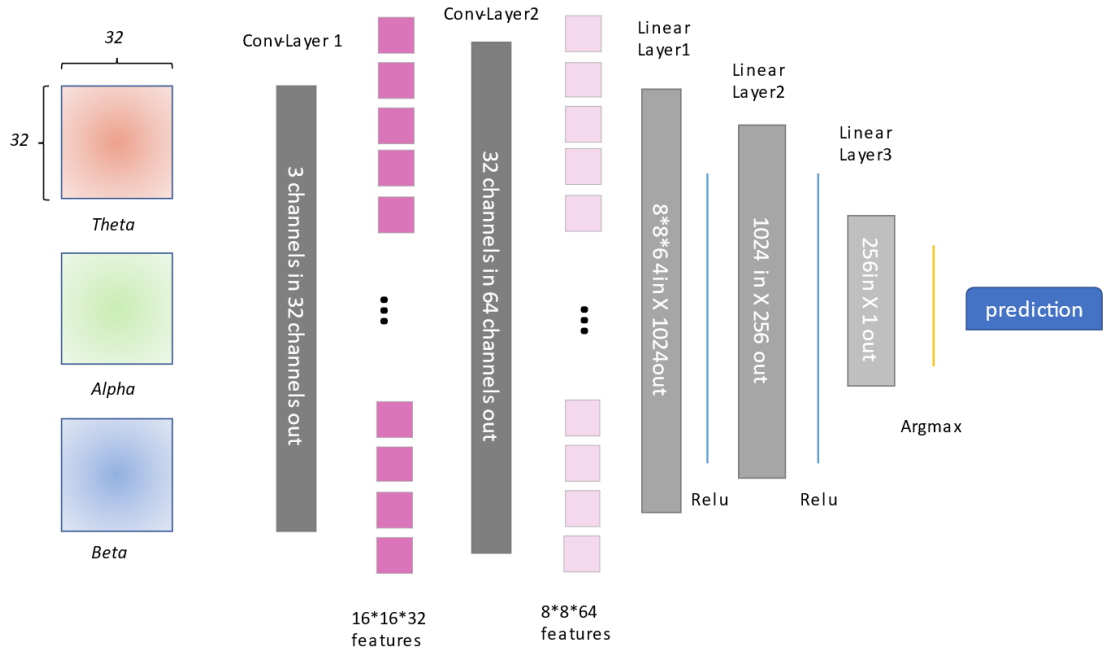



## 5 Design and implement 3 (convolutional neural network)

As introduced in abstract part and some analysis in [2], the extended version data is about images of 3channels generated from signals of 3 frequency bands. So, we also come up with a try using CNN on this task.

### 5.1 Network structure

**Figure 6**




### 4.2 Model tuning

**Table 4.** Final 5-layers CNN setting

| Net | num_epochs | batch size | learning rate | loss function | optimizer |
|---|---|---|---|---|---|
| **CNN** | 170 | 512 | 0.001 | cross entropy | Adam[7] |

## 6 Models testing and assessment.

### 6.1 Dataset distribution and partition

We firstly discuss several possible dataset division methods. Based on what we know about this dataset, these 11057 lines of data are collected from 122 subjects, there is a line of data called 'subjectId' records the subject of each line of features. So, **case 1**, which is also used in the tuning process of two models above, is to ignore the subject information and regard the dataset purely as a heap of features. **Case 2** [1]is to take data of a group of subjects as training data and the rest would be testing data. **Case 3** is to only generate training and testing data from features of one subject, of course, the data size would be relative smaller.

Before we take these 3 cases into model assessment work, we could have an initial interpretation of them. Case 1 is a general and sketchy way with no consideration on subject individuals, so I we could guess that model trained by this would be a generalization of rules across all individuals. However, with similar aim, case 2[1] helps build up the robust of the general model, because it separates train and test data with different group of subjects, so both the features and their corresponding subjects in test data are totally unknown to the model. On the contrary, case 3 focus on features within one subject. Figure 4 shows the features amount of each subjects, for testing with case 3 we may take data of several subjects with features as many as possible.

### 6.2 Partitions implement.

Train and test split ratio: 8/2

We set random state as 1 to remove the influence of randomness.

**Case 1 setting:**

```
x_train,x_test, y_train,y_test = train_test_split(X,Y,test_size = 0.2,random_
                                 state = 1)
```

**Case 2 setting:**

```
x_train, x_test, y_train,y_test = X[:8896],X[8896:],Y[:8896],Y[8896:]
```

Before 8896th data is from subject of 1 to 97, then the test set is from subjects of 98 to 122. These sizes are just around 8 to 2.

**Case 3 setting:**

Through detailed observation, we found almost all labels of one subject stay the same, so the learning process becomes very short and the model would directly guess one output out of 1 or 0 whatever the input is, so we decided to suspend this assessment method in following testing.

### 6.3 Testing.

| *Model* | Accuracy 1 (within-subject) | Accuracy 2(cross-subject) |
|---|---|---|
| *Two-layer DNN* | 0.92 | 0.63 |
| *BDNN* | 0.80 | 0.73 |
| *5-layer CNN* | 0.902 | 0.63 |

We could see that trained with train set randomly divided 80% out whole data would reach a relatively higher accuracy, that makes a lot of sense, because that model generates a relatively global understand of the whole dataset. By contrast, model trained in case two have generated the understanding of data from a part of subjects, so being unknow to the subjects in test set leads to a relatively low accuracy is also interpretable. This difference shows up distinctly when doing training on DNN. We do thousands of epochs over data of case 1, the testing accuracy could still grow slightly. But, under case 2, the model starts to be overfitting after 400 epochs of training.

Except comparison between assessment methods, we also could generate some insights from these 3 network structures. As we can see, a simple 2-layer DNN could finish this work with a relatively high accuracy (case 1). With applying BDNN, the model needs to pay attention on both forward and backward, so deservedly the forward

accuracy would be reduced with a consideration to the backward process. Besides, while training we found that we could slightly adjust the weights of forward and backward training by changing the interval of reversing and learning rates of two optimizers.

For the 5-layer CNN we built on extended data, with sight tuning the performance has easily reached a figure at 0.9 within-subject and 0.63 cross-subject, which is very close to the traditional DNN best performance. Technically, CNN could help us to extract more inner features such like positional information. I believe the CNN could reach a score obvious high that pure DNN, with help of a better embedded design on structure and a more precise tuning setting.

## 7 Conclusion and future work.

Considering this data is a reduced dataset, the time dimension with size 256 was reduced by averaging operation, DNN has done a considerable job in extracting rules from pure digital data. Besides, BDNN not only finish that work with an acceptable accuracy but also offers a backward use. In this context, a backward result from input 1 or 0 actually provides a pattern vector that is representative for a specific class, which may help researcher find more fresh insights [4] of this testing or process.

In future we may use the raw version of data to do more attempts with various views such like to extract positional or squencial features, or do some extra work on the time dimension.

## References


1. Yue Yao(&) , Jo Plested, and Tom Gedeon: Deep Feature Learning and Visualization for EEG Recording Using Autoencoders. Research School of Computer Science, The Australian National University, Canberra, Australia
2. A.F. Nejad and T.D. Gedeon：BiDirectional Neural Networks and Class Prototypes. Department of Artificial Intelligence School of Computer Science and Engineering The University of New South Wales Sydney 2052 AUSTRALIA
3. Pytorch. https://pytorch.org/
4. Pouya Bashivan，Irina Rish，Mohammed Yeasin，Noel Codella：LEARNING REPRESENTATIONS FROM EEG WITH DEEP RECURRENT-CONVOLUTIONAL NEURAL NETWORKS
5. Gedeon, T. D., Catalan, J. A., & Jin, J. Image Compression using Shared Weights and Bidirectional Networks. In Proceedings 2nd International ICSC Symposium on Soft Computing (SOCO'97) (pp. 374-381).
6. Turner, H and Gedeon, TD “Extracting Meaning from Neural Networks,” Proceedings 13th International Conference on AI, vol.1, pp. 243-252, Avignon, 1993.
7. Diederik, P., K., Jimmy, B.: Adam: A Method for Stochastic Optimization. In: The 3rd International Conference for Learning Representations, San Diego (2015)