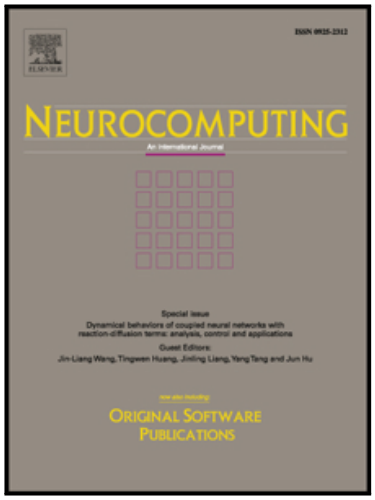
Communicated by Shiliang Sun

Accepted Manuscript



Deep Learning Based on Batch Normalization for P300 Signal Detection

Mingfei Liu, Wei Wu, Zhenghui Gu, Zhuliang Yu, FeiFei Qi, Yuanqing Li

PII: S0925-2312(17)31460-1
DOI: 10.1016/j.neucom.2017.08.039
Reference: NEUCOM 18831

To appear in: *Neurocomputing*

Received date: 30 April 2017
Revised date: 8 August 2017
Accepted date: 18 August 2017

Please cite this article as: Mingfei Liu, Wei Wu, Zhenghui Gu, Zhuliang Yu, FeiFei Qi, Yuanqing Li, Deep Learning Based on Batch Normalization for P300 Signal Detection, *Neurocomputing* (2017), doi: 10.1016/j.neucom.2017.08.039

Neurocomputing, Vol. 0, No. 0 (2017) 1–18

Deep Learning Based on Batch Normalization for P300 Signal Detection

Mingfei Liu.
School of Automation Science and Engineering,
South China University of Technology,
Guangzhou, 510641, China
E-mail: auliubanghoudai24@mail.scut.edu.cn

Wei Wu
School of Automation Science and Engineering,
South China University of Technology,
Guangzhou, 510641, China
Corresponding author. Email: auweiwu@scut.edu.cn

Zhenghui Gu
School of Automation Science and Engineering,
South China University of Technology,
Guangzhou, 510641, China
Corresponding author. Email: zhgu@scut.edu.cn

Zhuliang Yu
School of Automation Science and Engineering,
South China University of Technology,
Guangzhou, 510641, China
Email: zlyu@scut.edu.cn

FeiFei Qi
School of Automation Science and Engineering,
South China University of Technology,
Guangzhou, 510641, China
Email: jieyu_qiangwei@163.com

Yuanqing Li
School of Automation Science and Engineering,
South China University of Technology,
Guangzhou, 510641, China
Email: auyqli@scut.edu.cn

Detecting P300 signals from electroencephalography (EEG) is the key to establishing a P300 speller, which is a type of brain-computer interface (BCI) system based on the oddball paradigm that allows users to type messages simply by controlling eye-gazes. The convolutional neural network (CNN) is an approach that has achieved good P300 detection performances. However, the standard CNN may be prone to overfitting and the convergence may be slow. To address these issues, we develop a novel CNN, termed BN^3, for detecting P300 signals, where Batch Normalization is introduced in the input and convolutional layers to alleviate over-fitting, and the rectified linear unit (ReLU) is employed in the convolutional layers to accelerate training. Since our model is fully data-driven, it is capable of

automatically capturing the discriminative spatio-temporal features of the P300 signal. The results obtained on previous BCI competition P300 data sets show that BN^3 both achieves the state-of-the-art character recognition performance and that it outperforms existing detection approaches with small flashing epoch numbers. BN^3 can be used to improve the character recognition performance in P300 speller systems.

Keywords: Brain-Computer Interface (BCI); Deep Learning; Convolutional Neural Network (CNN); P300

1. Introduction

Patients who suffer from amyotrophic lateral sclerosis (ALS), Parkinson's disease or other motor disabilities usually require a direct and easily accessible communication approach. A brain-computer interface (BCI) bridges one's brain signals directly to the external world without needing any muscular activities from the subject.[1] Among different brain signals, EEG has relatively higher temporal resolution and properties like non-invasiveness, inexpensiveness and safety. The P300 event-related potential (ERP) is one of the EEG signals that is commonly used in building speller systems. Due to the non-invasive signal acquisition method, the collected P300 ERPs often have very low signal-to-noise ratios (SNRs). Thus, stimuli must be repeated to improve the robustness of the spelling process. However, increasing the number of repetitions also lowers the typing speed. Such issues can be mitigated by adaptively choosing the flashing times,[2] but more fundamentally, correct classification of P300 responses under a small repetition number is critical for building faster speller systems.[3]

Previous works on P300 classification mainly utilized traditional machine learning techniques such as support vector machine(SVM) and linear discriminant analysis (LDA). Alain et al.[4] ensembled 17 SVM classifiers, reaching an average accuracy of 96.5% in character predictions, ranking the top in the BCI competition 2004 data set II.[5] The proposed method filtered the signal into 0.1-10 Hz and down-sampled and averaged them to reduce noise and variability. Then, it used a recursive channel elimination algorithm to select the best number of channels for different subjects. Li at.al.[6] implemented ICA for ocular artifact correction before the SVM classification. Lin et. al.[7] used bagging linear discriminant analysis (LDA) to achieve a comparable result to SVMs in BCI competition II.[8]

Over the past decade, deep learning, as a subfield of machine learning, has made impressive advances in solving real-world problems such as computer vision and natural language processing. Novel deep architectures such as VGG net[9] and ResNet[10] have been proposed, empowered with the ability to capture sophisticated and hierarchical features of high-dimensional data. However, applications of deep neural nets in EEG signal detection are still at the begining stage.[11] Cecotti[12] developed a 4-layer convolutional neural network (CNN) to achieve comparable results to the ensembled SVMs. The spatial and temporal features of the input P300 signals were extracted respectively by the first and second layer of the CNN and then were sent to a fully connected layer for classification. More recently, Martin Langkvist et al[13] used deep belief networks (DBN) to perform automatic feature extractions on raw sleep data. Wulsin et al[14] modeled single channel EEG waveforms with DBNs for classification and anomaly detection in a semi-supervised paradigm. Stober et al[15] combined CNN with DLSVM output layers to achieve rhythm classification among individuals. However, there is a lack of efficient methods proposed for fast P300 classification in real-world speller systems.

Fig. 1. The P300 speller paradigm

In this paper, we propose a novel deep neural network to achieve state-of-the-art P300 signal classification and character recognition. EEG signals often have a low SNR and high subject variability, for which hand-designed features perform suboptimally. Deep convolutional neural networks, as proved in the field of computer vision, may be capable of capturing the intrinsic features of EEG signals. We introduce Batch Normalization and Dropout to improve the generalization of our model and use ReLU activations in convolutional layers to accelerate training. Our model combines the feature extraction and classification steps into one process, requiring less signal preprocessing work. The paper is organized as follows: In Section 2 we describe the P300 speller paradigm . In Section 3 we provide information regarding our experimental data, model architecture and training details. Experimental results are presented in Section 4 and discussions are provided in Section 5.

2. Background

The P300 speller paradigm, also called the Oddball Paradigm was first proposed by Farwell and-Dochin.[16] As shown in Figure 1, a 6×6 matrix of letters with its rows and columns flashing in random sequences is presented to the user during the experiment. When the user gaze at a letter in the matrix, an ERP will be elicited if the target letter is contained in the flashed row or column. Under this condition, the ERP is detected as a positive EEG peak of voltage 300 ms after the stimulus and is therefore named P300. After repetitions of flashes, we can accumulate the detection of P300 signals and take the intersection of the row and column with most detections as the intent character. In this way, the task can be translated into a P300 and non-P300 binary classification problem and the user can type messages simply by controlling his eye-gaze. Therefore, to build a fast and robust speller system, accurate classifications of the P300 responses are critical. In our experiments, we use Data set II of BCI Competition III[17] and Data Set IIb of BCI Competition II,[8] for 3 total subjects of data, which we abbreviate as III A, III B and II. All data sets were collected using the BCI2000 system under the Oddball Paradigm. Rows and columns of the matrix are randomly intensified at 5.7 Hz and repeated 15 times for each character. For the train and test data set, there are 85 and 100 characters in III A and III B, and 42 and 31 in II, respectively. Original signals are sampled at 240 Hz in 64 channels and bandpass filtered into 0.1-60 Hz. The topographic map of the 64 channels is shown in Figure 2.

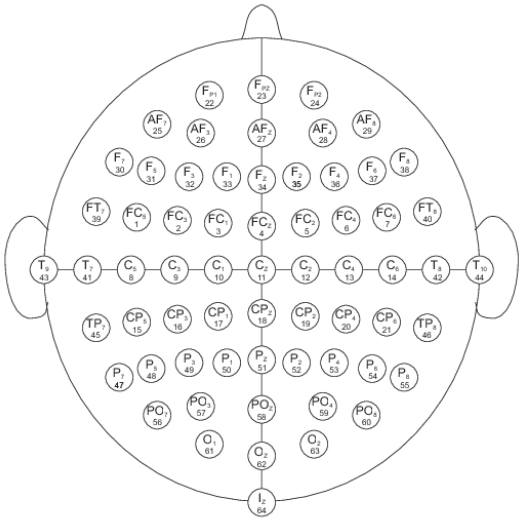


Fig. 2. The layout of the EEG montage for the P300 data set

3. Methods

3.1. *Data preprocessing*

According to the prior knowledge that the P300 ERP occurs approximately 300 ms relative to the stimulus onset, we use a wide time window of 0 to 667 ms after the stimulus in our data analysis. This results in an EEG data matrix of 64×160. We then filter the EEG signals with an 8th-order bandpass Butterworth filter with cut-off frequencies of 0.1 and 20 Hz.

Due to the design of the spelling paradigm, only 1/6 of the collected samples contain P300, which leads to highly unbalanced sample sizes of the P300 and non-P300 data. Under this setting, the binary classification is prone to biasing towards the non-P300 samples. To address this issue, we artificially make the numbers of the positive and negative training samples equal to each other by replicating the P300 training samples 4 times. Note that we do not normalize or downsample the signals during the preprocessing state since they are included in our proposed model, which will be described in the following sections. The sample size information of our data sets is shown in Table 1.

Table 1. Number of samples in the training and test data sets

Data set	Train		Test	
	P300	non-P300	P300	non-P300
III A	12750	12750	3000	15000
III B	12750	12750	3000	15000
II	6300	6300	930	4650

3.2. *Motivation and general architecture*

We develop a novel deep convolutional neural network for P300 classification. CNN is a special type of the multilayer perceptron (MLP) that was originally inspired by the topology of the visual cortex. It has been widely used in visual tasks such as object detection and hand-written digit recognition, and has experienced considerable progress in recent years.[9,10,18,19] In traditional computer vision problems, raw input data often contains high dimensions. This "curse of dimensionality" makes the use of traditional MLPs challenging due to computation amounts. To overcome this issue, CNN introduces the convolutional layers, whose properties of weight-sharing and sparse connectivity significantly reduce the computation burden and improve the model's robustness against signal variances.[20] Another feature of CNN is its ability to extract hierarchical features of raw data with little preprocessing required. With this advantage, a CNN could combine feature extraction and discrimination into one process. Automatic feature extraction has been proved to achieve better results than hand-designed features in many tasks. These advantages motivate our choice of using CNN as the building block for a pragmatic end-to-end speller.

Figure 3 depicts the architecture of our network, which is termed Batch Normalized Neural Network (BN^3). BN^3 consists of 6 layers, denoted by $L_{0\sim5}$. The central column of each layer indicates the dimensions of that layer's input and output. For example, the input layer with batch normalization keeps the original size of the data, and layer L_5 transforms a 128-dimensional vector into a scalar. Details of the model configurations are provided in the following subsections.

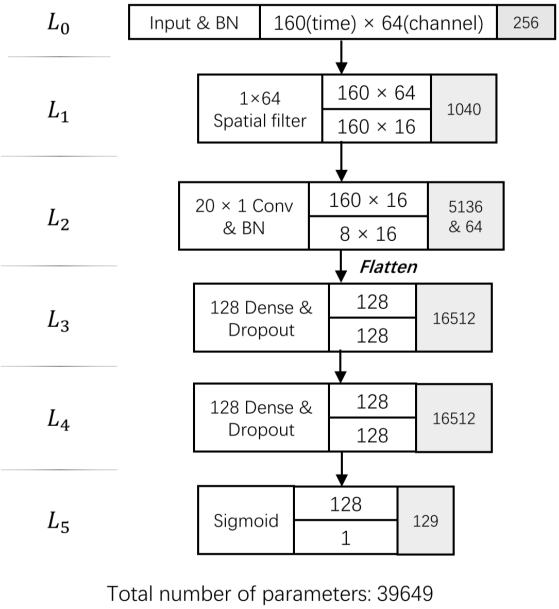


Fig. 3. Overview of the network architecture. Numbers in the grey boxes indicate the number of parameters in each layer.

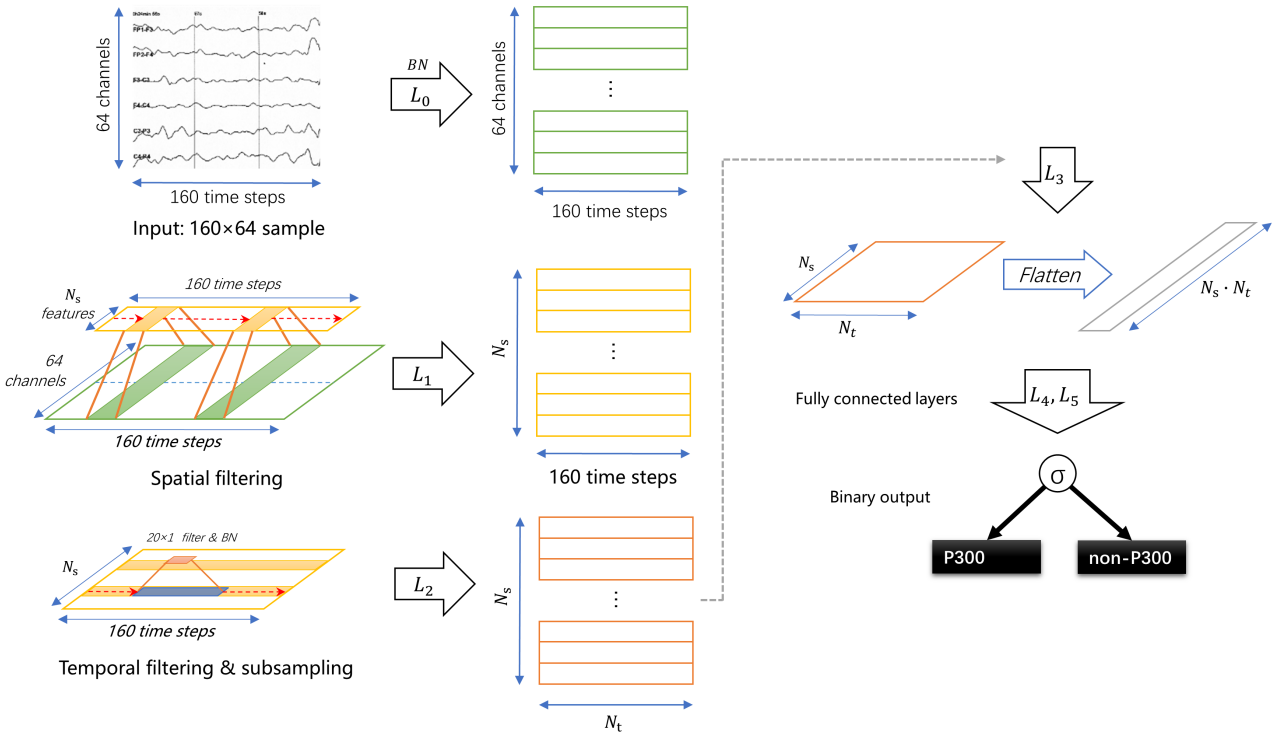


Fig. 4. Illustration of BN^3's workflow for P300 detection

3.3. *Network topology*

Our network consists of 6 sequential layers. L_0 is the input layer with batch normalization. L_1 is a 1-D convolutional layer for spatial feature extraction. L_2 is a 1-D convolutional and subsampling layer for temporal feature extraction. We use the Rectified Linear Unit (ReLU) as the activation function in this layer. Unlike the sigmoid function, the ReLU does not saturate and thus avoids the problem of vanishing gradients. Moreover, the efficient calculation of $ReLU(x) = max(x, 0)$ enables a faster convergence during the training stage.[21] L_3 and L_4 are fully-connected layers with *tanh* activations. L_5 is the output layer with a single sigmoid unit, which yields binary classification results. Figure 4 depicts the workflow of BN^3 for P300 detection.

Let X denote the 160×64 input tensor. The computations performed by each layer are described as follows.

- L_0:

$$a^{(0)} = BN(X) \tag{1}$$

First, we implement batch normalization for the input batches. The algorithm for $BN(x)$ is provided in the next subsection.

- L_1:

$$a_s^{(1)}(j) = \sum_{i=1}^{i \le N_{\text{elec}}} a_i^{(0)}(j) \cdot w_s^{(1)}(j) + b_s^{(1)} \tag{2}$$

L_1 serves as a spatial filtering layer, where $w_s^{(1)}$, $b_s^{(1)}$ are the weights and biases of the s^{th} filter, respectively, composing 64×1 convolution kernels. s indexes the spatial feature map with $1 \le s \le N_s$, and N_{elec} stands for the number of electrodes 64. Each $a_s^{(1)}$ output is a $T \times 1$ vector, where $T = 160$ is the length of the time window. Concatenating all $a_s^{(1)}$ $(1 \le s \le N_s)$ yields a $N_s \times T$ output tensor $a^{(1)}$.

- L_2:

$$z_k^{(2)}(j) = \sum_{s=1}^{s \le N_s} \sum_{t=1}^{t \le N_t} (a_k^{(1)}(t \cdot 20 + j) \cdot w_k^{(2)}(j) + b_k^{(2)}(j)) \tag{3}$$

$$\boldsymbol{a}_k^{(2)} = g(BN(\boldsymbol{z}_k^{(2)})) \tag{4}$$

where $g(x) = max(x, 0)$ is the ReLU activation function and t stands for the t^{th} temporal feature map. As suggested in Ref.22, batch normalization is performed before activation to avoid the distribution shift, which leads to saturation and decelerates learning. The convolutional kernels have a size of 20×1 with a stride of 20, thus the time course is downsampled into $N_t = 8$. In this layer, each kernel convolutes $a^{(1)}$ and outputs a $N_t \times 1$ vector $\boldsymbol{z}_k^{(2)}$, where k indexes the convoluted feature map. We extract N_k feature maps in this layer and the shape of data is thus transformed from $N_s \times T$ into $N_k \times N_t$. Then, it is flattened into a $N_k \cdot N_t$ dimensional feature vector $\boldsymbol{a}^{(2)}$ and fed to the next dense layer for classification.

We set the downsampling rate to 20 due to the following considerations. The original time window of each sample is 160, representing 667 ms of signals. We scale it 8 times gaining $667/8 = 83.375$ ms length of temporal filter. Our final setting for the temporal convolution of L_2 is non-overlapped. To prove the cost-efficiency of this configuration, we made an supplementary experiment setting the convolution stride to 10 and 5, leading to a half-overlapped and quarter-overlapped convolution. The result is presented in Fig. A.1. We can see that the single-trial performance of both overlapped-version models are slightly lower than the original version. Consider that the overlapped convolution requires more computation, while the original BN^3 model has 39649 parameters, BN^3-S10 has 53985 and BN^3-S5 has 82657. We can infer that making overlapped filters may provide redundant features that hamper the classification task, while significantly increasing the computations.

- L_3:

$$\boldsymbol{r}^{(3)} \sim \text{Bernoulli}(p^{(3)}) \tag{5}$$

$$\boldsymbol{a}^{(3)} = h((\boldsymbol{r}^{(3)} * \boldsymbol{a}^{(2)}) \cdot W^{(3)} + \boldsymbol{b}^{(3)}) \tag{6}$$

where $h(x) = \tanh(x)$ is the activation function, $W^{(3)}$, $b^{(3)}$ are the weights and biases, $p^{(3)}$ the scalar dropout rate. * denotes an element-wise product. Note that the dropout operation is only performed during training, which means that $p^{(3)}$ is always set to 1 during testing. Motivations and explanations of dropout are given in the next subsection.

- L_4:

$$\boldsymbol{r}^{(4)} \sim \text{Bernoulli}(p^{(4)}) \tag{7}$$

$$\boldsymbol{a}^{(4)} = h((\boldsymbol{r}^{(4)} * \boldsymbol{a}^{(3)}) \cdot W^{(4)} + \boldsymbol{b}^{(4)}) \tag{8}$$

The computation of L_4 is duplicated from L_3. We use two dense layers instead of one considering deep architectures have better generalization than shallow ones with the same number of neurons.

- L_5:

$$a^{(5)} = \sigma(\sum_{i=1}^{i \leq 128} (\boldsymbol{a}^{(4)} \cdot W^{(5)} + \boldsymbol{b}^{(5)})) \tag{9}$$

Where $\sigma(x) = \dfrac{1}{1 + e^{-x}}$ is the sigmoid activation function. With the final network output $a^{(5)}$, the P300 detection result is defined by:

$$E(X) = \begin{cases} 1 & a^{(5)} \geq 0.5 \\ 0 & a^{(5)} < 0.5 \end{cases} \tag{10}$$

We use a score vector $Q(j)$ to culminate the probabilities of the P300 detection for each of the 12 rows and columns, where $1 \leq j \leq 12$. Let n denotes the number of flash epochs that we use to predict the character, where $1 \leq i \leq 15$, then

$$Q(j) = \sum_{i=1}^{n} a^{(5)}(i) \qquad 1 \leq j \leq 12 \tag{11}$$

where $a^{(5)}(i, j)$ is the network output of epoch i corresponding to the flash j. Then the take the intersection of the row and column with the highest expectation of P300 detection as the predicted character. Let x and y denote the horizontal and vertical coordinates of the screen, then we have

$$x = \arg\max_{1 \leq i \leq 6} Q(i) \tag{12}$$

$$y = \arg\max_{7 \leq i \leq 12} Q(i) \tag{13}$$

Hence we obtain the exact location of the target character.

3.4. *Regularization techniques*

3.4.1. *Batch Normalization*

Batch Normalization[22] was first proposed to reduce internal covariate shift in neural network training. Let x denotes the input data and $g(x)$ the non-linear activation function. When x goes through a number of layers, it is likely that its distribution shifts into the saturating regime of the activation function. For instance, let the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ be the activation function, when $|x|$ becomes very large, the gradient of $\sigma(x)$ becomes very tiny. Such saturation greatly slows down training especially when the network goes deeper. Therefore, if we normalize the distribution of every input batch by $\hat{x} = \frac{x-\mu_x}{\sigma_x^2}$, which can be seen as a linear transformation, the issue of saturation can be avoided and faster training is guaranteed. In our network we apply feature-wise Batch Normalization, where each feature is normalized separately. The full computation process of BN is given as follows, where ϵ is a small constant for numerical stability.

- Input: Size m mini-batch $B = \{x_1, ..., x_m\}$, in which x_i for $i = 1...m$ are single train samples.
- Output: $\hat{x_i} = BN\{x_i\}$
- Algorithm:
  Mean of feature j in mini-batch

$$\mu(j) \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i(j) \tag{14}$$

  Variance of feature j in mini-batch

$$\sigma^2(j) \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i(j) - \mu(j))^2 \tag{15}$$

  Normalize feature j of each sample

$$\hat{x_i}(j) \leftarrow \frac{x_i(j) - \mu(j)}{\sigma^2(j) + \epsilon} \tag{16}$$

In our network, we add 2 batch normalization layers in L_0 and L_3 respectively. As stated in Section 2, we presume that the input BN layers has played the role of data normalization. Our later experiments show that removing any of these BN layers leads to much worse classification performances. Further details are presented in Section 5.

3.4.2. *Dropout*

Dropout[23] is an algorithm introduced to reduce complex co-adaptions between neurons, forcing each neuron to learn more robust features. It has been shown that dropout improves the generation of the network by incorporating three standard approaches to avoid overfitting: ensemble averaging, adding noise, and adding regularization terms to the error functions. In particular, dropout has led to vast improvements on several difficult speech and image recognition problems. Let p denotes the dropout rate of a layer, every neuron in this layer has a probability of $1 - p$ of being set to 0 during the forward propagation training stage. During the testing stage, every neuron is retained, so the whole model can be interpreted as an averaged one from many classifiers. A proper value of p helps neurons learn more distinct and robust features and thus less likely to overfit.

We employ dropout in the last two fully connected layers L_3 and L_4, with their computations provided in Section 3.3. Our experiments show that applying dropout in any of the previous layers only leads to worse classification performance, and a proper dropout rate in the dense layers contributes to a significantly higher character recognition rate under a small number of flashing epochs, which is advantageous for building faster and pragmatic speller systems.

3.5. *Training Settings*

We use the binary cross-entropy as the loss function and the gradient descent as the optimizer. The values of the hyper-parameters in the networks are determined as follows:

- We set $N_s = N_k = 16$.
- Learning rate of the Adam optimizer is set to 10^{-5} with no decay. Other parameters are set to the suggested values in,[24] which are $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.
- As suggested in Ref.23, optimal classification performance is achieved when the dropout rate is chosen from $0.4 \leq p \leq 0.8$. We perform a grid search within the interval of $0.4 \leq p \leq 1$ with a stride of 0.05 via a 5-fold cross validation, which yields 0.8 as the optimal value for both $p^{(3)}$ and $p^{(4)}$.
- The batch size for stochastic gradient descent is set to 64.
- All parameters are initialized with the Glorot Uniform[25] method.

Our model contains a total of 39,489 parameters. All codes are written in Python with Keras[26] library

with Tensorflow[27] backend.

4. Experiment

Our goal is to build an end-to-end P300 speller system, which consists of two parts. First, our classifier detects the P300 signals for each flashed row or column in the speller, Then, the intersection of the column and row with the most times of P300 detected is deemed the intent character. Accordingly, we measure our model from two perspectives: P300 detection rate and character recognition accuracy. To assess the detection rate, we introduce the following metrics: true positive (TP), true negative (TN), false positive (FP) and false negative (FN). The binary accuracy (Reco.) is defined as $Reco. = \frac{TP+TN}{TP+TN+FP+FN}$. In addition, we introduce Recall (also named Sensitivity), Precision and F1-Score as the classification metrics, which are calculated as follows:

$$Recall = \frac{TP}{TP+TN} \tag{17}$$

$$Precision = \frac{TP}{TP+FP} \tag{18}$$

$$F1\text{-}score = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \tag{19}$$

Diagrams of the P300 detection results are presented in the Appendix. In Table A.1, our model (BN^3) is compared with three CNN models (CNN-1, MCNN-1, MCNN-3) of the best performance reported in the literature[12] on data set III A and III B. Analysis of the correlation between P300 detection and the final character prediction performance is provided in the next section.

Table A.2 shows the results for the character recognition rates of our model in terms of flash epochs, compared with Cecotti's[12] three CNN models and Alain's ensembled SVM,[4] which yields the best result in the competition. Paired t-tests show that our model, BN^3, achieves significantly higher character prediction accuracy than the other models when the number of epochs is less than 6 (Table 4. all $p < 10^{-3}$).

Table 2. Results of paired t-tests between BN^3 and other models

BN^3 paired with	t Stat	$p-value$
CNN-1	5.047	3.47e-4
MCNN-1	5.653	1.56e-4
MCNN-3	6.101	8.95e-5
E-SVM	5.807	1.29e-4
Threshold of P	1.833	

To better illustrate the speed of our model, we plot the information transfer rate[28] (ITR) in bits per minute. Let P denote the probability to recognize a character, T the time to recognize and N the number of classes ($N = 36$). As each flash lasts for 100 ms followed by a pause of 75 ms and a pause of 2.5 s between each character epoch, T can be defined as

$$T = 2.5 + 2.1n \qquad 1 \le n \le 15 \tag{20}$$

Thus the ITR is defined by

$$ITR = \frac{60(Plog_2(P) + (1-P)log_2\frac{1-P}{N-1} + log_2 N))}{T} \tag{21}$$

The comparison of ITRs between BN^3 and other models is shown in Fig. 5. and Fig. 6. For data set II, we compare our result with the best in the competition, which was achieved with CWT and LDA by Vladimir Bostanov [a]. For data set III A and III B, we plot the ITR based on the average accuracy of two subjects with a standard deviation error. It can be seen from both figures that our model has a significantly higher ITR in the first epoch, which implies a higher single-trial accuracy and poses a big advantage in building pragmatic speller systems.

[a] http://www.bbci.de/competition/ii/results/bostanov_iib_desc.txt

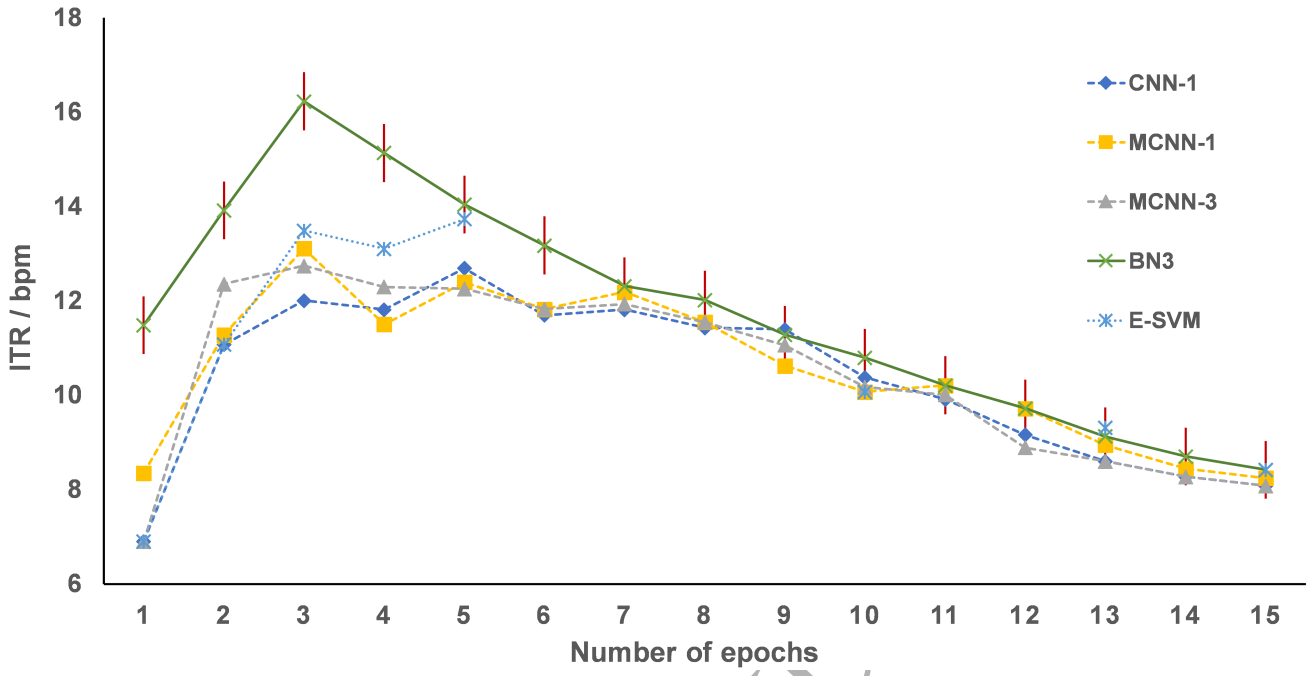


Fig. 6. ITR based on the average accuracy of two subjects III A and III B (from the data set II of BCI Competition III). The red vertical line is the error bar bzased on standard deviation.

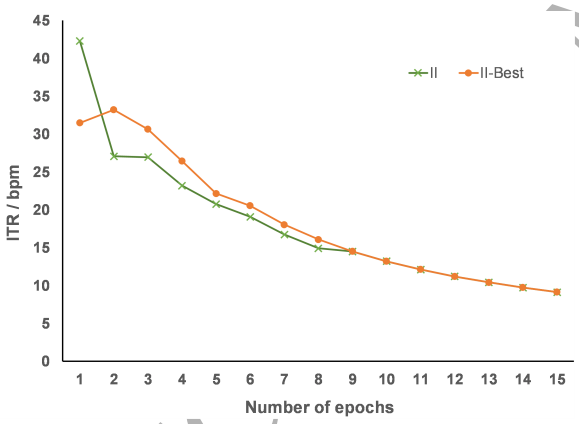


Fig. 5. ITR comparison between BN^3 (II) and Vladimir's method (II-Best) on data set II (Data Set IIb of BCI Competition II)

Table 3. Most discriminant electrodes of subject III A and III B

III A	FC_2	C_5	C_Z	CP_6
	F_{P1}	F_{P2}	FT_7	FT_5
	P_5	P_5	P_Z	PO_7
	PO_7	PO_3	O_1	O_1
III B	CP_2	CP_2	F_5	F_2
	F_6	T_7	T_8	T_9
	P_5	P_3	P_2	P_2
	P_6	P_6	PO_Z	PO_4

As CNN has the property of automatic feature extraction, we first probe into the trained filters of our model and explore the features they extracted, hoping that it could provide insights to the patterns of P300 signals. We first visualize the spatial filters of L_1. Fig. 7. shows the 4 electrodes with strongest weight of each learned filter in L_1. Table 6 lists the most discriminant electrodes of each filter. We can see that the learned spatial filters generally accords with those of previous studies,[4,12] and both subjects share some determinant electrodes like P_5. Generally our model has extracted more subject-dependent discriminant filters, indicating that the L_1 of our model is more relying on distinguished subject patterns.

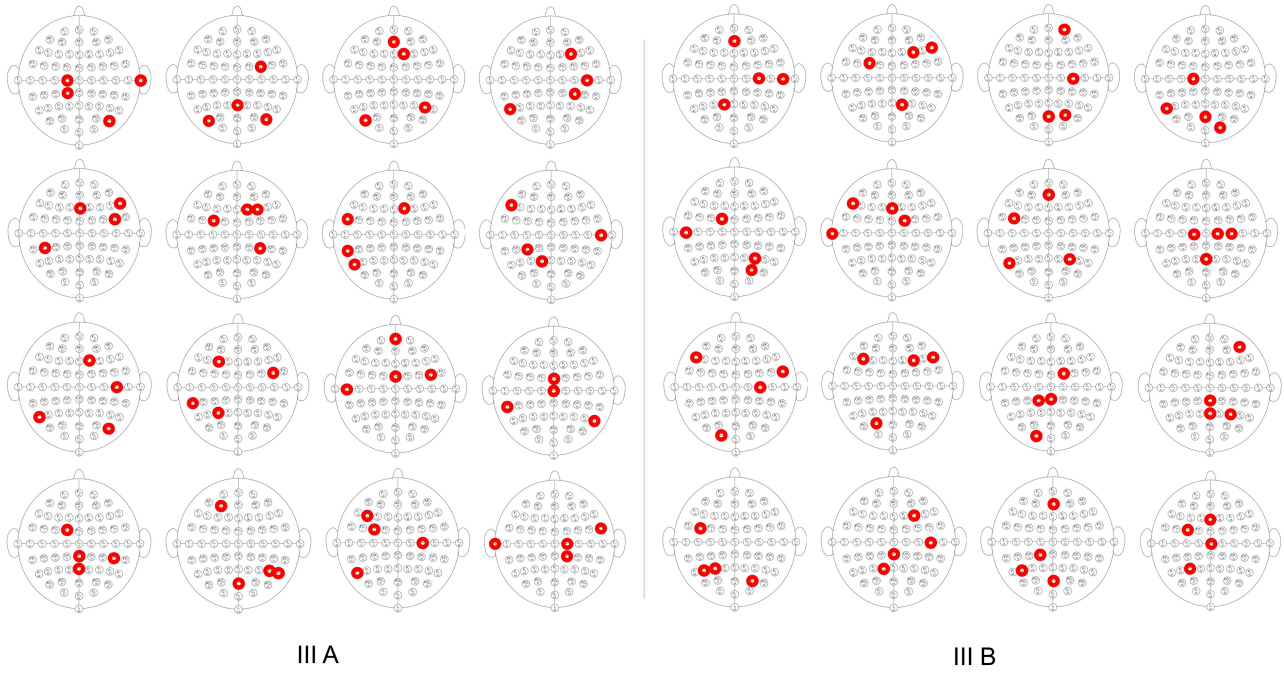


Fig. 7. Most discriminant electrodes of each filter in data set III A and III B

5. Discussions

5.1. *Role of BN layers*

To further test the effects of the two Batch Normalization layers in the network, we modified our model architecture into three morphs: BN^0 removes both BN layers from the original model, BN^1 removes only the second BN layer in the convolution, and BN^2 removes only the first BN layer after the input, respectively. We run our model under the same conditions and their ITRs are presented in Fig. 8. and Fig. 9. As shown, removing one BN layer does not markedly influence the model's performance, while removing both BN layers leads to considerable decline in classification accuracy in each data set. In particular, for both data sets, removing both BN layers significantly undermines the single-trial performance of the model. Such a decline may indicate that the BN layers play a role of powerful feature extractors, boosting the signal-to-noise ratio in single trials, thus reducing the need for more data repetitions and increasing the system speed.

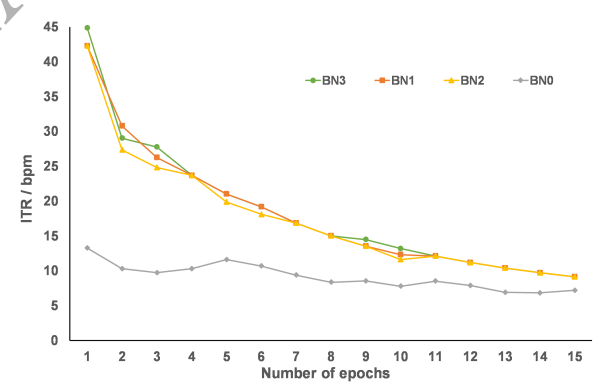


Fig. 8. ITR comparison of model BN^3, BN^2, BN^1 and BN^0 on data set II(Data Set IIb of BCI Competition II)

5.2. *Correlations between metrics and P300 detection*

The speller system is divided into two parts: P300 detection and character prediction. In Section 4, we introduced Recall, Precision, F1-Score and binary accuracy as metrics. However, it is unclear which metric is the most correlated to the accuracy of character prediction. Here, to provide further insight we explore the correlations between the final accuracy

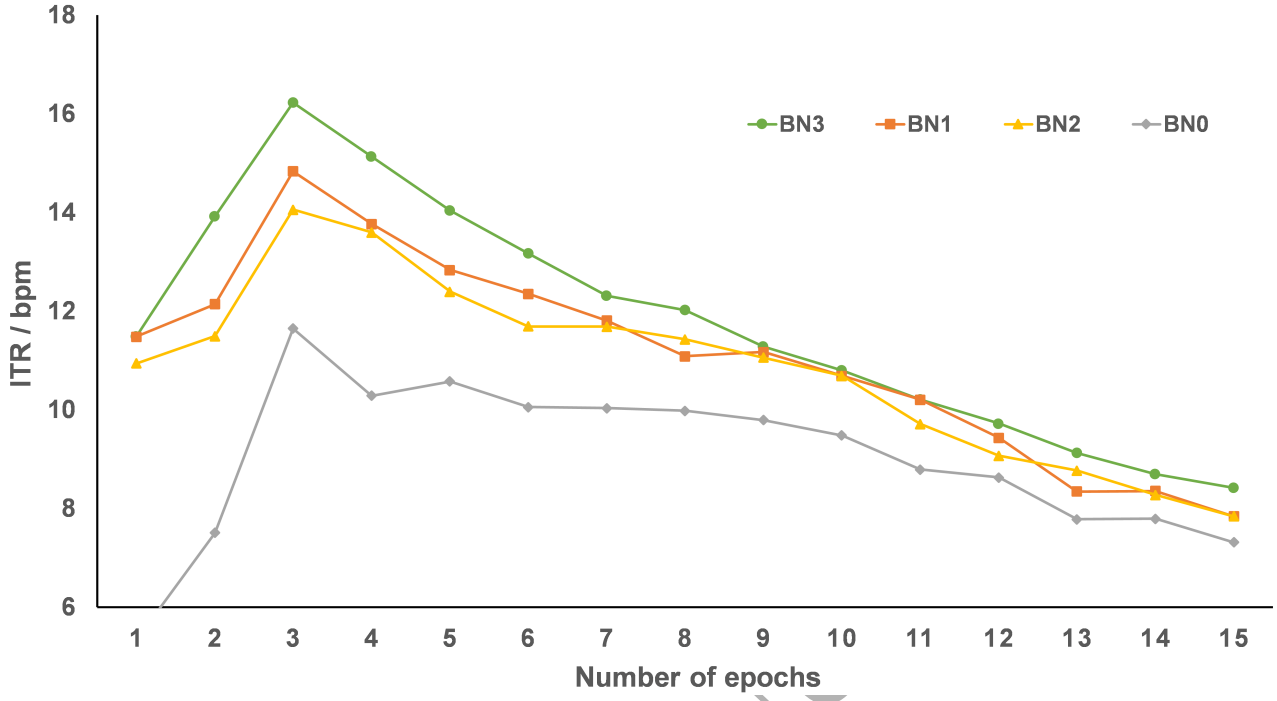


Fig. 9. ITR comparison of model BN^3, BN^2, BN^1 and BN^0 on data set III (Data set II of BCI Competition III)

and P300 detection performance. For subjects III A and III B , we randomly initialize 20 models and train them with the same configurations and record each model's metrics. Then we compute the correlation coefficients between their character prediction accuracy and Recall, Prediction, F1-score and Reco. respectively.

Fig. 10. illustrates the scatter plot of the P300 detection metrics and character prediction accuracy of 40 models. Let $M = \{m_1, m_2, ..., m_n\}$ denote a group of metric values and $Y = \{y_1, y_2, ..., y_n\}$ be its corresponding accuracy, then the correlation coefficient between them is defined by

$$Cor(M, Y) = \frac{\sum_{i=1}^{n}(m_i - \overline{M})(y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(m_i - \overline{M})^2 \sum_{i=1}^{n}(y_i - \overline{Y})^2}} \quad (22)$$

where $\overline{M}$ and $\overline{Y}$ are the mean values of M and Y and $n = 20$. Table 5 shows the correlation coefficient values between different metrics. We see that for subject III B, there are higher correlations between P300 detection and the character prediction performance. More importantly, the F1-Score has the highest correlation coefficient values for both subjects. These results suggest that the F1-score of P300 detection may be a better indicator of the overall performance than Recall.

Table 4. Correlation coefficients of different metrics

Metric	III A	III B
Recall	0.2076	0.4662
Precision	0.3519	0.6651
F1-Score	**0.4506**	**0.6700**
Recog.	0.3279	0.6099

5.3. *Speed increase by ReLU activation*

We also assess the speed increase due to the ReLU activation. We train 20 models on data set III B, with half of the models' activation of L_2 replaced by sigmoid function. Then we calculate the average run time of training these two types of models. Experiments are carried out on a Intel i7 3770 CPU with Windows 10 Pro OS without GPU acceleration. Table 3 is the comparison result, where the ReLU activation is on average 12 seconds faster than sigmoid, and the fact that the sigmoid version of models has

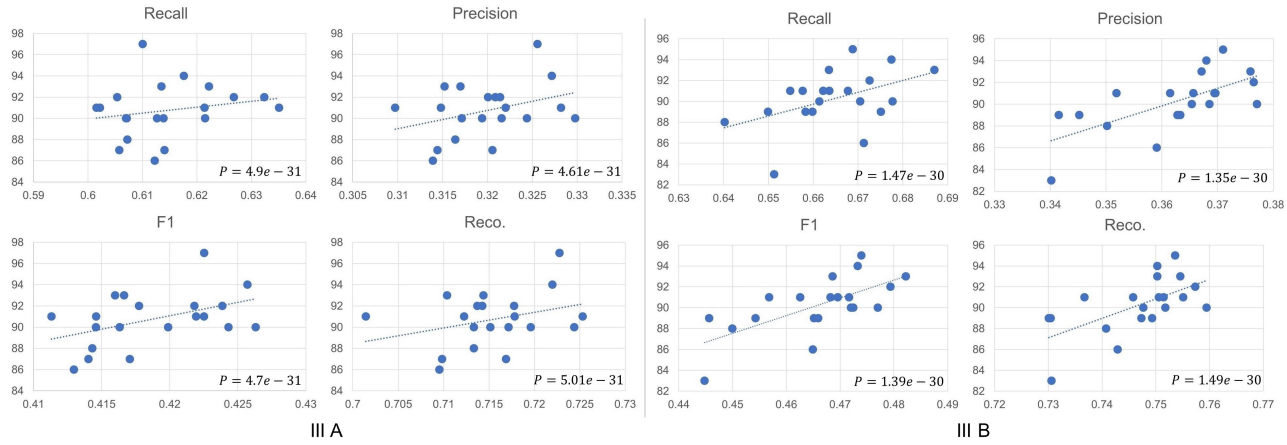


Fig. 10. Scatter plot of P300 detection metrics and final accuracy for data set III A and III B (from Data set II of BCI Competition III)

much worse character recognition performance. The speed increase are expected to be even more dramatic under bigger training batches and GPU support.

Currently out model is mainly focused on offline classifications of P300 signaFls. More practical applications require online and asynchronous signal detection. Considering the excessive motion artifacts in online scenarios, it is worth studying combining online artifacts reduction methods[29] with batch normalized networks to achieve good performance. Methods of object detection in the field of computer vision, such as Selective Search[30] in Region-based CNN can be applied to control-state detection.

Table 5. Run time comparision between ReLU and Sigmoid activations

	ReLu	Sigmoid
Mean(s)	602	614
Std.	3.85	3.74

6. Conclusions

We propose a novel deep learning method called BN^3 for P300 signal detection. Our method is fully data-driven while achieving the state-of-art character recognition performance by capturing subject-specific spatio-temporal features. In particular, under the settings where the number of the flashing epochs is small, our method achieves the best performance among the compared approaches. Compared with the standard CNN, our model is less susceptible to overfitting and training is faster. As a result, our method can be used to enhance P300 spellers. Since our method does not make specific assumptions regarding P300, it can also be employed as a general framework to all types of event-related potentials.

Despite the encouraging performance of BN^3, visualization of the features in the intermediate layers in the network is presented. This is essential for gaining more insight into the internal operation and behavior of the network. Future work will be conducted to study the outputs of the intermediate layers by using recently proposed visualization techniques.

Acknowledgment

This work is supported in part by the National Key Basic Research Program of China (973 Program) under Grant 2015CB351703, in part by the National Natural Science Foundation of China under Grants 61403144, 61573152, 61573150, 61633010, and 91420302, and in part by the tip-top Scientific and Technical Innovative Youth Talents of Guangdong special support program (No. 2015TQ01X361).

References

1. Dan Zhang, Huaying Song, Honglai Xu, Wei Wu, Shangkai Gao, and Bo Hong. An N200 Speller Integrating the Spatial Profile for the Detection of the Non-control State. *Journal of Neural Engineering*, 9(2):026016, 2012.
2. Jing Jin, Brendan Z Allison, Eric W Sellers, Clemens Brunner, Petar Horki, Xingyu Wang, and Christa

Neuper. An Adaptive P300-based Control System. *Journal of Neural Engineering*, 8(3):036006, 2011.
3. Zhenghui Gu, Zhuliang Yu, Zhifang Shen, and Yuanqing Li. An Online Semi-supervised Brain–Computer Interface. *IEEE Transactions on Biomedical Engineering*, 60(9):2614–2623, 2013.
4. Alain Rakotomamonjy and Vincent Guigue. BCI Competition III: dataset II-ensemble of SVMs for BCI P300 Speller. *IEEE Transactions on Biomedical Engineering*, 55(3):1147–1154, 2008.
5. Dean Krusienski and Gerwin Schalk. BCI Competition III Challenge 2004. 2004.
6. Yandong Li, Zhongwei Ma, Wenkai Lu, and Yanda Li. Automatic Removal of the Eye Blink Artifact from EEG Using an ICA-based Template Matching Approach. *Physiological Measurement*, 27(4):425, 2006.
7. V. Bostanov. BCI Competition 2003-Data Sets Ib and IIb: Feature Extraction from Event-related Brain Potentials with the Continuous Wavelet Transform and the t-value Scalogram. *IEEE Transactions on Biomedical Engineering*, 51(6):1057–1061, June 2004.
8. B Blankertz. Documentation Second Wadsworth BCI Dataset (P300 Evoked Potentials) Data Acquired Using BCI2000 P300 Speller Paradigm. *BCI Classification Contest November*, 2002.
9. Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.
10. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
11. Wei Wu, Srikantan Nagarajan, and Zhe Chen. Bayesian Machine Learning: EEG/MEG Signal Processing Measurements. *IEEE Signal Processing Magazine*, 33(1):14–36, 2016.
12. Hubert Cecotti and Axel Graser. Convolutional Neural Networks for P300 Detection with Application to Brain-computer Interfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):433–445, 2011.
13. Martin Längkvist, Lars Karlsson, and Amy Loutfi. Sleep Stage Classification Using Unsupervised Feature Learning. *Advances in Artificial Neural Systems*, 2012:5, 2012.
14. DF Wulsin, JR Gupta, R Mani, JA Blanco, and B Litt. Modeling Electroencephalography Waveforms with Semi-supervised Deep Belief Nets: Fast Classification and Anomaly Measurement. *Journal of Neural Engineering*, 8(3):036015, 2011.
15. Sebastian Stober, Daniel J Cameron, and Jessica A Grahn. Using Convolutional Neural Networks to Recognize Rhythm Stimuli from Electroencephalography Recordings. In *Advances in Neural Information Processing Systems*, pages 1449–1457, 2014.
16. Lawrence Ashley Farwell and Emanuel Donchin. Talking Off the Top of Your Head: Toward a Mental Prosthesis Utilizing Event-related Brain Potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6):510–523, 1988.
17. DJ Krusienski and G Schalk. Wadsworth BCI Dataset (P300 Evoked Potentials). *BCI Competition III Challenge*, 2004.
18. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances In neural Information Processing Systems*, pages 1097–1105, 2012.
19. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
20. Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.
21. George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving Deep Neural Networks for LVCSR Using Rectified Linear Units and Dropout. pages 8609–8613, 2013.
22. Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167*, 2015.
23. Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
24. Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.
25. Xavier Glorot and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Aistats*, volume 9, pages 249–256, 2010.
26. François Chollet. Keras. https://github.com/fchollet/keras, 2015.
27. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04467*, 2016.
28. C. E. Shannon. A Mathematical Theory of Communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.
29. Dan Zhang, Bisheng Huang, Wei Wu, and Siliang Li. An Idle-state Detection Algorithm for SSVEP-based Brain–Computer Interfaces Using a Maximum Evoked Response Spatial Filter. *International Journal of Neural Systems*, 25(07):1550030, 2015.

30. Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

Appendix A Supplementary materials

Table A.1. P300 Detection Results

Data Set	Model	TP	TN	FP	FN	Recog.	Recall	Precision	F-1 score
III A	BN^3	1910	11615	3385	1090	0.7513	0.6367	0.3607	**0.4605**
	CNN-1	2021	10645	4355	979	0.7037	0.6737	0.3170	0.4311
	MCNN-1	2071	10348	4652	929	0.6899	0.6903	0.3080	0.4260
	MCNN-3	2023	10645	4355	977	0.7038	0.6743	0.3172	0.4314
III B	BN^3	2084	12139	2861	916	0.7902	0.6947	0.4214	**0.5246**
	CNN-1	2035	12039	2961	965	0.7037	0.6783	0.4073	0.5090
	MCNN-1	2202	11453	3547	798	0.6899	0.7340	0.3830	0.5034
	MCNN-3	2077	11997	3003	923	0.7038	0.6923	0.4089	0.5141
II	BN^3	752	3960	690	178	0.8444	0.8086	0.5215	0.6341

Table A.2. Character recognition number versus the number of epochs

Data Set	Model	Epochs														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
III A	BN^3	**22**	**39**	**58**	**67**	**73**	**75**	**79**	**81**	82	**86**	89	**92**	**94**	**96**	**98**
	CNN-1	16	33	47	52	61	65	77	78	85	86	90	91	91	93	97
	MCNN-1	18	31	50	54	61	68	76	76	79	82	89	92	91	93	97
	MCNN-3	17	35	50	55	63	67	78	79	84	85	91	90	92	94	97
	E-SVM	16	32	52	60	72	-	-	-	-	83	-	-	94	-	97
III B	BN^3	**47**	**59**	**70**	**73**	76	**82**	84	91	**94**	**95**	**95**	**95**	94	94	95
	CNN-1	35	52	59	68	79	81	82	89	92	91	91	90	91	92	92
	MCNN-1	39	55	62	64	77	79	86	92	91	92	95	95	95	94	94
	MCNN-3	34	56	60	68	74	80	82	89	90	90	91	88	90	91	92
	E-SVM	35	53	62	68	75	-	-	-	-	91	-	-	96	-	96
II	BN^3	**24**	23	27	28	29	30	30	30	31	31	31	31	31	31	31
	Best	20	26	29	30	30	31	31	31	31	31	31	31	31	31	31

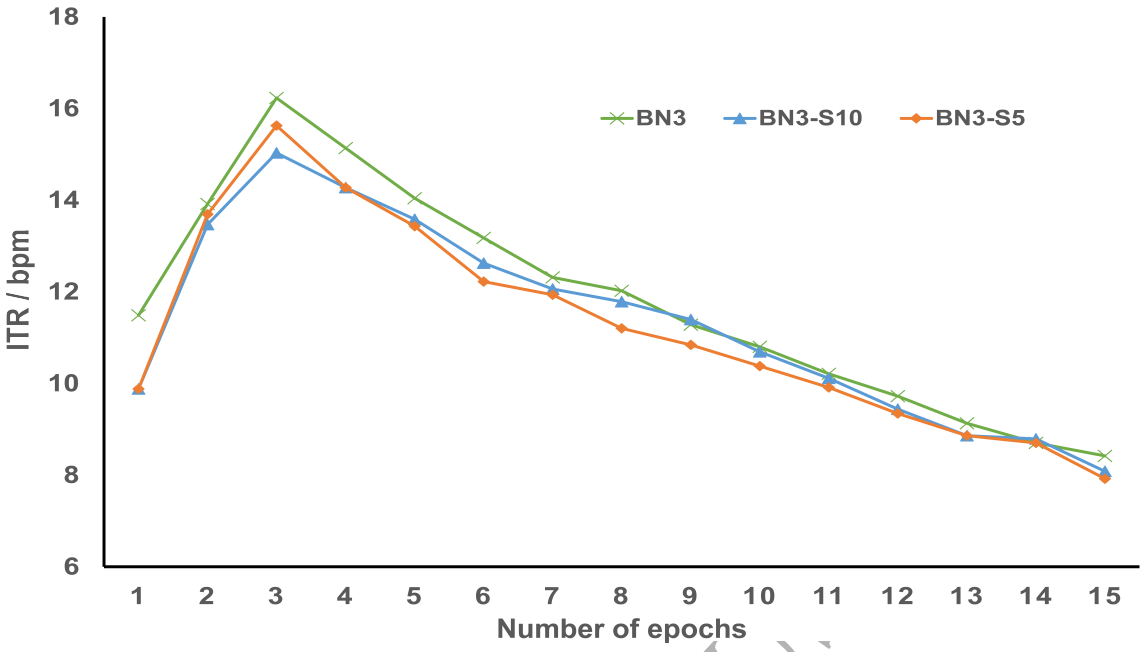


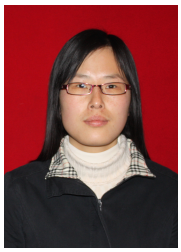
Fig. A.1. ITR comparison of model BN^3, BN^3-S5, BN^3-S10 on data set III

Mingfei Liu is a junior undergraduate student of School of Automation Science and Engineering, South China University of Technology. His research interests include deep learning and brain signal processing, in particular developing deep neural network models for the analysis of EEG signals.

Wei Wu received the PhD degree in biomedical engineering from Tsinghua University, China, in 2012. From 2008 to 2010, he was a visiting student at the Neuroscience Statistics Laboratory, Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology. Since 2012, he has been with the School of Automation Science and Engineering, South China University of Technology, as an associate professor. He works in the field of neural signal processing and neural engineering, specializing in particular on developing statistical models and algorithms for the analysis and decoding of brain signals. He is an associate editor of Neurocomputing (Elsevier) and Neural Processing Letters (Springer), and a member of IEEE Biomedical Signal Processing Technical Committee. He is a senior member of the IEEE.

Zhenghui Gu received the Ph.D. degree from Nanyang Technological University, Singapore, in 2003. From 2002 to 2008, she was with Institute for Infocomm Research, Singapore. She joined the College of Automation Science and Engineering, South China University of Technology, in 2009 as an Associate Professor. She was promoted to be a full Professor in 2015. Her research interests include the fields of signal processing and pattern recognition.

Zhu Liang Yu received the BSEE and MSEE degrees, both in electronic engineering, from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1995 and 1998, respectively, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2006. He joined Center for Signal Processing, Nanyang Technological University from 2000 as a Research Engineer, then as a Group Leader from 2001. In 2008, he joined the College of Automation Science and Engineering, South China University of Technology and was promoted to be a full Professor in 2011. His research interests include signal processing, pattern recognition, machine learning and their applications in communications, biomedical engineering, etc.



Feifei Qi received the B.S. degree in applied mathematics from Qufu Normal University, Qufu, China, in 2010. She is currently pursuing the Ph.D. degree in pattern recognition and intelligent systems with the South China University of Technology, Guangzhou, China. Her current research interests include pattern recognition and brain signal analysis.

Yuanqing Li was born in Hunan Province, China, in 1966. He received the B.S. degree in applied mathematics from Wuhan University, Wuhan, China, in 1988, the M.S. degree in applied mathematics from South China Normal University, Guangzhou, China, in 1994, and the Ph.D. degree in control theory and applications from South China University of Technology, Guangzhou, China, in 1997. Since 1997, he has been with South China University of Technology, where he became a full Professor in 2004. In 2002-2004, he worked at the Laboratory for Advanced Brain Signal Processing, RIKEN Brain Science Institute, Saitama, Japan, as a researcher. In 2004-08, he worked at the Laboratory for Neural Signal Processing, Institute for Inforcomm Research, Singapore, as a research scientist. His research interests include, blind signal processing, sparse representation, machine learning, brain–computer interface, EEG and fMRI data analysis. He is the author or coauthor of more than 60 scientific papers in journals and conference proceedings.