Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

A Simple Convolutional Neural Network for Accurate P300 Detection and Character Spelling in Brain Computer Interface

Hongchang Shan, Yu Liu, Todor Stefanov
Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands
h.shan@liacs.leidenuniv.nl, y.liu@liacs.leidenuniv.nl, t.p.stefanov@liacs.leidenuniv.nl

Abstract

A Brain Computer Interface (BCI) character speller allows human-beings to directly spell characters using eye-gazes, thereby building communication between the human brain and a computer. Convolutional Neural Networks (CNNs) have shown better performance than traditional machine learning methods for BCI signal recognition and its application to the character speller. However, current CNN architectures limit further accuracy improvements of signal detection and character spelling and also need high complexity to achieve competitive accuracy, thereby preventing the use of CNNs in portable BCIs. To address these issues, we propose a novel and simple CNN which effectively learns feature representations from both raw temporal information and raw spatial information. The complexity of the proposed CNN is significantly reduced compared with state-of-the-art CNNs for BCI signal detection. We perform experiments on three benchmark datasets and compare our results with those in previous research works which report the best results. The comparison shows that our proposed CNN can increase the signal detection accuracy by up to 15.61% and the character spelling accuracy by up to 19.35%.

1 Introduction

A Brain Computer Interface (BCI) translates brain signals into computer commands, thereby building communication between the human brain and outside devices. In this way, human-beings can use only the brain to express their thoughts without any real movement. BCI has been developed to help locked-in (e.g. Amyotrophic Lateral Sclerosis (ALS)) patients [Sellers and Donchin, 2006]. In recent years, BCI has also been popularly developed for healthy people, in application domains such as entertainments [Gilroy *et al.*, 2013], mental state monitoring [Lin *et al.*, 2013] as well as in IoT services [Lin *et al.*, 2014]. Electroencephalogram (EEG)-based BCI attracts most of the research due to its noninvasive way of measuring/acquiring brain signals and easy recording with inexpensive equipment. Among all kinds of EEG signals, P300 performs outstandingly well in character spelling applications. Therefore, this paper considers the P300 signal detection as our main BCI task and the P300-based character speller as our target BCI application.

A P300 signal is very difficult to detect because of its very low signal-to-noise ratio (SNR). Previous research on P300 detection and P300-based spellers uses traditional machine learning methods, namely manually-designed signal processing techniques for feature extraction as well as classifiers like Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA). It focuses on enhancing P300 potentials [Rivet *et al.*, 2009], extracting useful features [Bostanov, 2004], choosing the most relevant EEG electrodes [Cecotti *et al.*, 2011], or removing artifacts caused by the muscle contraction [Gao *et al.*, 2010], the eye movement [Mennes *et al.*, 2010] and the body movement [Gwin *et al.*, 2010]. Unfortunately, manually-designed feature extraction and traditional classification techniques have the following problems: 1) they can only learn the features that researchers are focusing on but lose or remove other underlying features; 2) brain signals have subject-to-subject variability, which makes it possible that methods performing well on certain subjects (with similar age or occupation) may not give a satisfactory performance on others. These problems limit the potential of manually-designed feature extraction and traditional classification techniques for further accuracy improvements.

In recent years, deep learning, especially using Convolutional Neural Networks (CNNs), has achieved significant performance improvements in the computer vision field [Krizhevsky *et al.*, 2012; Simonyan and Zisserman, 2014; He *et al.*, 2016]. Deep CNNs have the advantage of automatically learning feature representations from raw data. They can learn not only something we know but also something important and unknown to us. Automatically learning from raw data has better ability to achieve good results which are invariant to different subjects. Thus, CNNs are able to boost the full potential of recognizing BCI signals, overcoming the aforementioned shortcomings of traditional machine learning methods.

Therefore, in recent years, researchers have started to design (deep) CNNs for P300-based BCIs [Cecotti and Graser, 2011; Manor and Geva, 2015; Liu *et al.*, 2017] and achieved better accuracy than traditional techniques. However, these CNNs first perform spatial convolution on raw data and then they perform temporal convolution on the abstract data coming from the spatial convolution. In this way, the input for a temporal convolution layer is not raw temporal signals. In fact, raw temporal signals are more important

to learn P300-related feature representations. Therefore, these CNN architectures lose useful raw temporal information and this leads to problems that: 1) they prevent further P300 detection and spelling accuracy improvements; 2) they require high network complexity to achieve competitive accuracy, which prevents the use of these CNNs for portable BCIs, like the mobile-based BCI [Wang *et al.*, 2011; Chen *et al.*, 2016].

To solve the problems mentioned above, we propose a simple, yet efficient CNN architecture which can capture feature representations from both raw temporal and raw spatial information. The complexity is significantly reduced while improving the P300 detection accuracy and P300-based spelling accuracy. The novel contributions of this paper are the following:

- We propose a CNN architecture with only one convolution layer. Our CNN is able to better learn P300-related features from both raw temporal information and raw spatial information. It exhibits very low network complexity.
- We perform experiments on three benchmark datasets and compare our results with those in previous research works which report the best results. The comparison shows that our proposed CNN can increase the P300 signal detection accuracy by up to 15.61% and the character spelling accuracy by up to 19.35%.

The rest of the paper is organized as follows: Section 2 describes the related work. Section 3 provides background information on the P300 signal, its detection, its application on character spelling, and the datasets used in this paper. Section 4 presents the proposed CNN. Section 5 compares the complexity, the P300 signal detection accuracy, and the character spelling accuracy between the proposed CNN and other methods on P300 detection and spelling. Section 6 ends the paper with conclusions.

2 Related Work

The general architecture of current CNNs for P300-based BCI [Cecotti and Graser, 2011; Manor and Geva, 2015; Liu *et al.*, 2017] uses the input tensor ($N \times C$) shown in Figure 1, where N denotes the number of temporal signal samples and C denotes the number of electrodes used for EEG signal recording and obtaining the samples. This architecture has three stages. In the first stage, it performs convolution along space to learn spatial features. In the second stage, it performs convolution along time to learn temporal features. In the final stage, it uses fully-connected layers to make accurate correlation between learned features and a particular class.

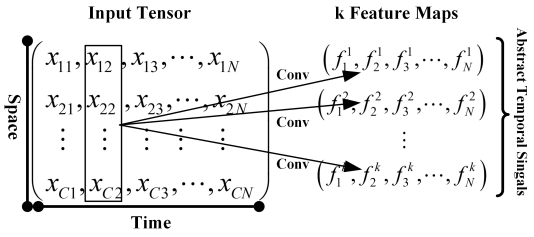

Figure 1: Spatial convolution in current CNNs. x denotes a signal sample in the input tensor. f denotes a datum in a feature map.

Cecotti [Cecotti and Graser, 2011] is the first to propose the aforementioned architecture. Let us call his architecture CCNN. Table 1 shows the detailed architecture of CCNN. The first column in the table describes the sequence of layers. The second column describes the operation in a layer. The third column describes the kernel size in the convolution layers. The last column describes the number of feature maps/neurons in a layer.

Layer	Operation	Kernel Size	Feature Maps/Neurons
1	Convolution	(1,C)	10
2	Convolution	(13,1)	50
3	Fully-Connected	—	100
Output	Fully-Connected	—	2

Table 1: CCNN architecture.

Liu [Liu *et al.*, 2017] improves CCNN by combining Batch Normalization and Droupout techniques (see Table 2). This CNN is named BN3 in [Liu *et al.*, 2017]. BN3 does not perform input normalization in the preprocessing but uses Batch Normalization: one is in Layer 1 and the other is in Layer 3. BN3 also employs dropout in the fully-connected layers to reduce overfitting. Before the output layer, BN3 uses two fully-connected layers instead of one for better generalization and accumulation of features.

Layer	Operation	Kernel Size	Feature Maps/Neurons
1	Batch Norm	—	—
	Convolution	(1,C)	16
2	Convolution	(20,1)	16
	Batch Norm	—	16
3	Fully-Connected	—	128
	Dropout	—	128
4	Fully-Connected	—	128
	Dropout	—	128
Output	Fully-Connected	—	2

Table 2: BN3 architecture.

Manor [Manor and Geva, 2015] proposes a deep CNN for P300 signal detection. Let us call his architecture CNN-R. It is shown in Table 3. CNN-R improves CCNN by using a deeper and wider network architecture. It uses smaller kernel size but more layers for temporal convolution. It also uses two fully-connected layers before the output layer. In addition, CNN-R uses more feature maps for the convolution layers and more neurons for the fully-connected layers. For such complex network, CNN-R uses pooling as well as dropout to reduce overfitting.

Layer	Operation	Kernel Size	Feature Maps/Neurons
1	Convolution	(1,C)	96
	Pooling	(3,1)	96
2	Convolution	(6,1)	96
	Pooling	(3,1)	96
3	Convolution	(6,1)	96
4	Fully-Connected	—	2048
	Dropout	—	2048
5	Fully-Connected	—	4096
	Dropout	—	4096
Output	Fully-Connected	—	2

Table 3: CNN-R architecture.

The problem of the aforementioned CNNs is that they all perform a spatial convolution with kernel (1,C) in the first layer, which makes these CNNs not able to learn temporal features well. This spatial convolution operation is shown in Figure 1. Every column in the input tensor contains a set of C signal samples. These samples come from C electrodes at a certain sampling time point. The spatial convolution opera-

tion converts each column of spatial data from the input tensor into an abstract datum in a feature map. The spatial convolution layer (the first layer) outputs several feature maps, which are given as input to the temporal convolution layer. These feature maps are abstract temporal signals instead of raw temporal signals. Thus, the spatial convolution operation leads to losing raw temporal information. Losing raw temporal information means losing important temporal features, because the nature of P300 signals is the positive voltage potential in raw temporal information, see Figure 2 explained in Section 3.1, as well as many important P300-related features are also embodied in raw temporal information [Polich, 2007]. As a result, the network can not learn temporal features well. Due to this problem, the aforementioned CNNs have to use a deeper and wider network architecture to learn temporal features better and achieve competitive accuracy. As a result, these CNNs exhibit high complexity.

In contrast, our novel CNN architecture performs both spatial convolution and temporal convolution in the first layer instead of performing only spatial convolution as in the aforementioned CNNs. As a result, our CNN is able to learn feature representations from raw temporal information and at the same time, it can also learn spatial features. Therefore, our CNN learns P300-related features better. By learning in this way, our CNN can achieve better accuracy (see Section 5.3 and Section 5.4) with only one convolution layer and without fully-connected layers before the output layer, which reduces the network complexity significantly (see Section 5.2).

3 Background

In this section, we first introduce the P300 signal and its detection followed by its character speller application. Then, we describe the benchmark datasets used in this paper.

3.1 P300 Detection and Speller

The P300 signal is the largest event-related potential (ERP), first reported by Sutton [Sutton *et al.*, 1967]. A P300 signal, recorded in EEG, occurs with a positive deflection in voltage at a latency about 300ms after a rare stimulus, as shown in Figure 2. The P300 detection is a binary classification problem: one class corresponds to a P300 signal within a certain time period while the second class corresponds to non-P300 within the time period.

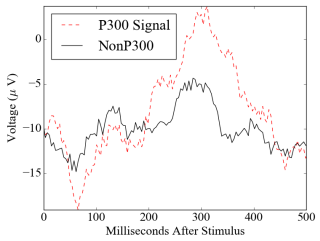

Figure 2: P300 signal.

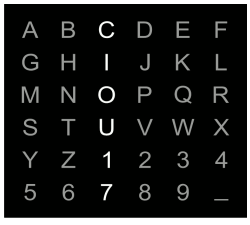

Figure 3: P300 speller character matrix.

Farwell and Donchin developed the first P300-based BCI speller in 1988 [Farwell and Donchin, 1988]. The subject in the experiment is presented with a 6 by 6 character matrix (see Figure 3) and he focuses his attention on a target character he wants to spell. All rows and columns in this matrix are intensified successively and randomly but separately. Two out of twelve intensifications contain the target character, i.e., one target row and one target column. As a result, the target row/column intensification becomes a rare stimulus to the subject. A P300 signal is then evoked by the rare stimulus. By detecting the P300 signal, we can infer which row or column the subject is focused on. By combing the row and column positions, we can infer the target character position.

Assume that one epoch includes 12 intensifications, in which there exist one target row intensification and one target column intensification. Then, in theory, one epoch is sufficient to infer one target character. However, in practice, since the P300 signal has a very low SNR and is also influenced by artifacts, one epoch can hardly be sufficient to infer one target character correctly. As a result, in practice, experimenters use many epochs to help the subject's brain generate more P300 signals. Then by counting which row/column intensification has evoked the most P300 signals, we can inter the target character. Using more epochs guarantees higher character spelling accuracy but impairs the communication speed between the human brain and the computer.

3.2 Datasets

This paper uses three benchmark datasets, namely, BCI Competition II - Data set IIb [Blankertz, 2003] as well as BCI Competition III - Data set II Subject A and Subject B [Blankertz, 2008]. Since many P300-based BCI algorithms use these three benchmark datasets, we can fairly compare the performance of our CNN with those of other state-of-the-art methods on P300 detection and spelling. Here, we give a short description of the three datasets.

BCI Competition II - Data set IIb and BCI Competition III - Data set II Subject A and Subject B are provided by the Wadsworth Center, NYS Department of Health. They are recorded with the BCI2000 platform, using the P300 speller developed by Farwell and Donchin. Brain signals are collected from 64 electrodes at a sampling frequency of 240Hz. One intensification lasts for 100ms, followed by a 75ms blank period for the matrix. The experiment uses 15 epochs for each character. After each sequence of 15 epochs, the matrix is blank for 2.5s, to inform the subject that this character is completed and to focus on the next character.

In BCI Competition II - Data set IIb, there is one subject with separated training and test datasets. The training dataset has 42 characters and the test dataset has 31 characters. In each character epoch, composed of 12 sets of signal samples, 2 sets are supposed to have a P300 signal and 10 sets are supposed to not have a P300 signal. So, the training dataset has 42 * 15 * 2 = 1260 sets of signal samples labelled "P300", and there are 42 * 15 * 10 = 6300 sets labelled "non-P300". The test dataset has 930 sets of signal samples labelled "P300" and 4650 sets labelled "non-P300".

In BCI Competition III - Data set II, there are two subjects. We call them Subject A and Subject B. For each subject, the training dataset has 85 characters and the test dataset has 100 characters. So, the training dataset has 2550 sets of signal samples labelled "P300" and 12750 sets labelled "non-P300". The test dataset has 3000 sets of signal samples labelled "P300" and 15000 sets labelled "non-P300".

Table 4 shows the number of P300s/non-P300s for each dataset. II denotes BCI Competition II - Data set IIb, III-A denotes BCI Competition III - Data set II Subject A, and III-B denotes BCI Competition III - Data set II Subject B.

Dataset	Train		Test	
	P300	non-P300	P300	non-P300
II	1260	6300	930	4650
III-A	2550	12750	3000	15000
III-B	2550	12750	3000	15000

Table 4: Number of P300s/non-P300s for each dataset.

4 Proposed Convolutional Neural Network

In this section, we introduce our novel CNN. We call it One Convolution Layer Neural Network (OCLNN). First, in Section 4.1, we describe the input to the network. In Section 4.2, we describe our proposed network architecture. In Section 4.3, we explain how we train the network. Finally, in Section 4.4, we describe how our CNN is used in a character speller application.

4.1 Input to the Network

The input to OCLNN is the tensor ($N \times C$) shown in Figure 4. C denotes the number of electrodes used for EEG signal recording and obtaining the samples. N denotes the number of temporal signal samples. Here $N = T_s \times F_s$, where T_s denotes the time period between 0 and T_s posterior to the beginning of each row/column intensification (see Section 3.1), and F_s denotes the signal sampling frequency.

In the input tensor, the temporal signal samples are band-pass filtered between 0.1Hz and 20Hz to remove high frequency noise. Then, the temporal signal samples are normalized to have zero mean and unit variance based on each individual pattern and for each electrode. Each individual pattern represents N signal samples in the time period between 0 and T_s posterior to the beginning of each intensification.

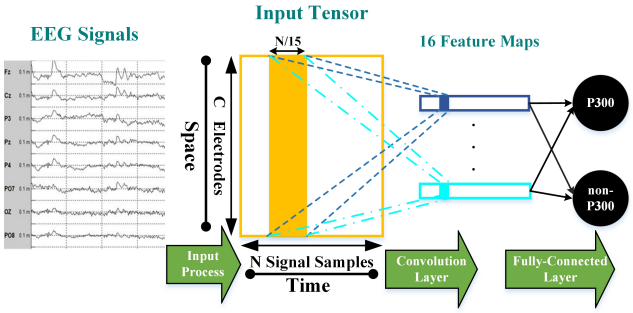

Figure 4: Illustration of OCLNN for P300 signal detection.

4.2 Network Architecture

The architecture of OCLNN is described in Table 5 and illustrated in Figure 4. The first column in the table describes the sequence of layers. The second column describes the operation in a layer. The third column describes the kernel size in the convolution layer. The last column describes the number of feature maps/neurons in a layer. We have 2 layers in total, i.e., Layer 1 and Layer Output.

Layer	Operation	Kernel Size	Feature Maps/Neurons
1	Convolution	(N/15,C)	16
	Dropout	—	—
Output	Fully-Connected	—	2

Table 5: OCLNN architecture.

In Layer 1, we segment the temporal signals from all electrodes into 15 parts and perform convolution operation on each part to learn features. Therefore, the kernel size of the convolution operation is (N/15, C) and each receptive field of the input tensor contains a (N/15, C) tensor of signal samples. In the time domain, these signal samples come from a time period of T_s/15. In the space domain, these signal samples come from all C electrodes. The convolution operation in this layer converts each receptive filed of data into an abstract datum in a feature map. In this way, this layer learns features from both raw temporal information and raw spatial information. We do not employ overlapped convolution, so the stride for the convolution operation is N/15. We use a Rectified Linear Unit (ReLU) as an activation function to model a neuron's output in this layer because a network with ReLUs is trained much faster than with traditional activation functions [Krizhevsky *et al.*, 2012]. In this layer, we employ dropout [Srivastava *et al.*, 2014] to reduce overfitting. The dropout rate is set to be 0.25. This layer generates 16 feature maps.

In Layer Output, OCLNN performs fully-connected operation. There are two neurons in this layer. One neuron represents the class "P300" and the other neuron represents the class "non-P300". The fully-connected operation makes correlation between the feature maps from Layer 1 and the two classes. We employ Softmax as an activation function for the neurons in this layer. The output of the Softmax function for class "P300" and class "non-P300" is denoted by $P^1_{(i,j)}$ and $P^0_{(i,j)}$, respectively. Therefore, $P^1_{(i,j)}$ represents the probability of having a P300 signal and $P^0_{(i,j)}$ represents the probability of not having a P300 signal at epoch i and intensification j. Thus, the detection of a P300 signal is defined by Equation 1, where $X_{(i,j)}$ is the input tensor to be classified and E is the binary classifier.

$$E(X_{(i,j)}) = \begin{cases} 1 & if \quad P^1_{(i,j)} > P^0_{(i,j)} \\ 0 & otherwise \end{cases} \tag{1}$$

4.3 Training

The training of OCLNN is carried out by minimizing the binary cross-entropy loss function. It uses Stochastic Gradient Descent as an optimizer with momentum and weight decay. The learning rate is set to be 0.01. The momentum is set to be 0.9. The batch size is set to be 128. The weight decay is set to be 0.0005. The weights and biases of all neurons in the convolution layer are regularised by L2 Regularizer.

4.4 Character Spelling Using the Network

We use $P^1_{(i,j)}$, the output of OCLNN for class "P300", to calculate the position of the target character in the P300 speller application described in Section 3.1. The detailed calculation process is defined by Equation 2, 3 and 4, where $C_{(j)}$ denotes the sum of the probabilities, $index_{col}$ denotes the column index of the target character in the matrix in Figure 3, and $index_{row}$ denotes the row index of the target character. When $j \in [1, 6]$, j denotes a column intensification. When $j \in [7, 12]$, j denotes a row intensification.

Equation 2 cumulates the probabilities of having a P300 signal evoked by intensification j over all the epochs. In Equation 3, we assign the index of the maximum $C_{(j)}$ to $index_{col}$ when $j \in [1, 6]$. This equation finds the index of the column intensification, with the maximum sum of probabilities, to have evoked a P300 signal. This index is the column position of the target character. In Equation 4, the row position of the target character is calculated in the same way as in

Equation 3. The position of the target character in the matrix in Figure 3 is the coordinate formed by the row position and the column position.

$$C_{(j)} = \sum_{i=1}^{n} P^1_{(i,j)} \quad (2)$$

$$index_{col} = \underset{1 \le j \le 6}{argmax}\ C_{(j)} \quad (3)$$

$$index_{row} = \underset{7 \le j \le 12}{argmax}\ C_{(j)} \quad (4)$$

5 Experimental Results

First, we introduce our experimental setup in Section 5.1. Then, we present the experimental results and show the performance comparison between OCLNN and other related research works in terms of complexity (see Section 5.2), P300 detection accuracy (see Section 5.3) and character spelling accuracy (see Section 5.4).

5.1 Experimental Setup

We train OCLNN using each training dataset in Dataset II, III-A and III-B, described in Section 3.2, separately. Thus, the number of used electrodes is 64 and the signal sampling frequency is 240 Hz. Therefore, for the input to OCLNN (see Section 4.1), we have C = 64 and F_s = 240 Hz. T_s = 1000ms because we take each individual pattern to be the signal samples between 0 and 1000 ms posterior to the beginning of each intensification. Then, $N = T_s \times F_s = 240$.

We run each of our trained OCLNNs on the corresponding test dataset in Dataset II, III-A and III-B and calculate the P300 detection accuracy using Equation 5 and the character spelling accuracy using Equation 6 for each test dataset. In Equation 5, acc_{P300} denotes the P300 detection accuracy, N_{tp} denotes the number of truly classified P300s for a test dataset, N_{tn} denotes the number of truly classified non-P300s for the test dataset, and S_{pn} denotes the number of all P300s and non-P300s in the test dataset. In Equation 6, $acc_{char(k)}$ denotes the character spelling accuracy when using the first k epochs for each character in a test dataset, $N_{tc(k)}$ denotes the number of truly predicted characters when using the first k epochs for each character in the test dataset, and S_c denotes the number of all characters in the test dataset.

$$acc_{P300} = \frac{N_{tp} + N_{tn}}{S_{pn}} \quad (5) \qquad acc_{char(k)} = \frac{N_{tc(k)}}{S_c} \quad (6)$$

OCLNN is implemented using Keras [Chollet and others, 2015] with the Tensorflow [Abadi *et al.*, 2016] backend. The network is trained on an NVIDIA GeForce GTX 980 Ti GPU.

For a fair comparison with CNN-R [Manor and Geva, 2015], we apply the bandpass filtering and signal decimation methods used for our OCLNN on CNN-R because we obtain low character spelling accuracy for CNN-R using the original filtering and decimation methods in [Manor and Geva, 2015].

5.2 Complexity

In this section, we compare the complexity, in terms of number of parameters and layers, of OCLNN with the networks CCNN, BN3, and CNN-R described in Section 2. The number of parameters is the number of weights and biases for all neurons in a network. We show the complexity in Table 6. The first row in the table lists the CNNs for comparison. The second row provides the number of parameters for each CNN. The third row shows the number of layers used in each CNN.

	OCLNN	CCNN	BN3	CNN-R
Parameters	16882	37502	39489	21950818
Layers	2	4	5	6

Table 6: Complexity comparison of different CNNs.

In terms of number of parameters, OCLNN is much smaller than the other three CNNs. OCLNN has only 16882 parameters while CCNN has 37502 parameters[1], BN3 has 39489 parameters, and CNN-R has 21950818 parameters. Thus, the number of parameters for OCLNN is only 45%, 42%, and 0.07% of that for CCNN, BN3, and CNN-R, respectively.

In terms of number of layers used in a CNN, OCLNN has less layers than the other three CNNs. OCLNN has only 2 layers while CCNN has 4 layers, BN3 has 5 layers, and CNN-R has 6 layers. Thus, the number of layers in OCLNN is only 50%, 40%, and 33.33% of that in CCNN, BN3, and CNN-R, respectively.

5.3 P300 Detection Accuracy

This section compares the P300 detection accuracies achieved by OCLNN with accuracies obtained by CCNN, MCNN-1, BN3, and CNN-R on Dataset II, III-A and III-B. MCNN-1 [Cecotti and Graser, 2011] is a multi-classifier with five CCNNs.

The P300 detection accuracy is shown in Table 7. The first row in the table lists the CNNs used for comparison. The second, third, and last row show the P300 detection accuracy of the different CNNs on Dataset II, III-A, III-B, respectively. The numbers are given in percentage (%) and calculated using Equation 5. An accuracy number in bold indicates the highest accuracy along a row. "–" in the table means that the accuracy is not reported in the reference paper describing the corresponding CNN.

	OCLNN	CCNN	MCNN-1	BN3	CNN-R
P300 Accuracy on II	**92.41**	–	–	84.44	86.29
P300 Accuracy on III-A	**84.60**	70.37	68.99	75.13	73.06
P300 Accuracy on III-B	**86.40**	78.19	75.86	79.02	79.80

Table 7: P300 detection accuracy of different CNNs on Dataset II, III-A and III-B.

Overall, OCLNN achieves the highest accuracies among all CNNs on Dataset II, III-A and III-B. It increases the P300 detection accuracies obtained from the other CNNs by up to 15.61%. For Dataset II, OCLNN achieves 92.41% P300 detection accuracy. The accuracy achieved by OCLNN is 7.97% and 6.12% higher than that achieved by BN3 and CNN-R, respectively. For Dataset III-A, OCLNN achieves 84.60% P300 detection accuracy. The accuracy achieved by OCLNN is 14.23%, 15.61%, 9.47%, and 11.54% higher than that achieved by CCNN, MCNN-1, BN3, and CNN-R, respectively. For Dataset III-B, OCLNN achieves 86.40% P300 detection accuracy. The accuracy achieved by OCLNN is 8.21%, 10.54%, 7.38%, and 6.60% higher than that achieved by CCNN, MCNN-1, BN3, and CNN-R, respectively.

5.4 Character Spelling Accuracy

This section compares the character spelling accuracies achieved by OCLNN and the accuracies achieved by CCNN,

[1]Cecotti [Cecotti and Graser, 2011] calculated the number of parameters erroneously for L_2. It should be $5N_s$*(13*N_s+1) instead of $5N_s$*(13+1)

MCNN-1, BN3, CNN-R, and ESVM [Rakotomamonjy and Guigue, 2008] for Dataset III-A and III-B, as well as the character spelling accuracies achieved by OCLNN and the accuracies achieved by BN3, CNN-R, and Bostanov [Bostanov, 2004] for Dataset II. The paper [Cecotti and Graser, 2011] describing CCNN and MCNN-1 does not report the accuracies for Dataset II. ESVM is the champion spelling method of BCI Competition III - Data set II. Bostanov is the champion spelling method of BCI Competition II - Data set IIb.

Table 8, 9, and 10 show the character spelling accuracies of different methods on Dataset II, III-A and III-B, respectively. The first column in a table lists the different methods we compare. Each row provides the character spelling accuracy of a method calculated by Equation 6 for different epoch numbers $k \in [1, 15]$. An accuracy number in bold indicates the highest accuracy along a column. "–" in a table means the accuracy is not reported in the reference paper describing the corresponding method.

Method	Character Spelling Accuracy (in%) / Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	**77.42**	**90.32**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
CNN-R	70.97	83.87	93.55	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
BN3	**77.42**	74.19	80.65	83.87	93.55	96.77	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Bostanov	64.52	83.87	93.55	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**

Table 8: Spelling accuracy of different methods on Dataset II.

Method	Character Spelling Accuracy (in%) / Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	**23**	**39**	56	63	**73**	**79**	**82**	**85**	**90**	**91**	**94**	**95**	**95**	**96**	**99**
CCNN	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
CNN-R	14	28	38	53	57	62	71	75	77	82	89	87	87	92	95
BN3	22	**39**	**58**	**67**	**73**	75	79	81	82	86	89	92	94	**96**	98
ESVM	16	32	52	60	72	–	–	–	–	83	–	–	94	–	97

Table 9: Spelling accuracy of different methods on Dataset III-A.

Method	Character Spelling Accuracy (in%) / Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	46	**62**	**72**	**79**	**84**	**87**	**89**	**93**	**94**	**96**	**97**	**97**	**97**	**98**	**98**
CCNN	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
CNN-R	36	46	66	70	77	80	86	86	88	91	94	95	95	96	96
BN3	**47**	59	70	73	76	82	84	91	**94**	95	95	95	94	94	95
ESVM	35	53	62	68	75	–	–	–	–	91	–	–	96	–	96

Table 10: Spelling accuracy of different methods on Dataset III-B.

The goal of the aforementioned competitions is to achieve the highest character spelling accuracy using epoch number $k = 15$. For this goal, OCLNN achieves the best results for all three datasets in the two competitions. For Dataset III-A and III-B, we achieve 99% and 98% spelling accuracy for $k = 15$. For Dataset II, we only need 3 epochs to achieve 100% spelling accuracy.

Improving the character spelling accuracy using less epochs will increase the Information Transfer Rate (ITR) [Wolpaw and Wolpaw, 2012], thereby increasing the communication speed between a human brain and a computer. Therefore, we also analyse the character spelling accuracies for every epoch number $k \in [1, 15]$. Overall, in most cases, OCLNN achieves better accuracies than the other methods. OCLNN increases the character spelling accuracies obtained from the other methods by up to 19.35%.

For Dataset II, OCLNN achieves the highest character spelling accuracies for every epoch number $k \in [1, 15]$ among all methods. Compared with the accuracies obtained by CNN-R, BN3, and Bostanov, our OCLNN increases the accuracies by up to 6.45%, 19.35%, and 12.90%, respectively.

For Dataset III-A, when compared with methods CCNN, MCNN-1, CNN-R, and ESVM, our OCLNN achieves the highest character spelling accuracies for every epoch number $k \in [1, 15]$ among all the methods. OCLNN increases the accuracies by up to 14%, 12%, 18%, and 8% compared with the accuracies obtained by CCNN, MCNN-1, CNN-R, and ESVM, respectively.

For Dataset III-B, when compared with methods CCNN, MCNN-1, CNN-R, and ESVM, our OCLNN achieves the highest character spelling accuracies for every epoch number $k \in [1, 15]$ among all the methods. OCLNN increases the accuracies by up to 13%, 15%, 16%, and 11% compared with the accuracies obtained by CCNN, MCNN-1, CNN-R, and ESVM, respectively.

When compared with BN3 on Dataset III-A and III-B, our OCLNN increases the accuracies by up to 8% considering epoch numbers 1, 2, and 5 to 15 on Dataset III-A as well as OCLNN increases the accuracies by up to 8% considering epoch numbers 2 to 15 on Dataset III-B. However, OCLNN decreases the accuracies for epoch numbers 3 and 4 on Dataset III-A and for epoch number 1 on Dataset III-B. This is because BN3 uses the Batch Normalization operation to improve the accuracies on smaller epoch numbers [Liu *et al.*, 2017]. However, the Batch Normalization operation used in BN3 can only improve the accuracies on Dataset III-A and III-B. On Dataset II, BN3 achieves much worse results on smaller epoch numbers. In OCLNN, we do not use the Batch Normalization operation because we aim at a CNN with better potential to achieve higher accuracies across different datasets obtained from different subjects. The Batch Normalization operation is not very helpful to our OCLNN because it is more useful in deep CNNs [Ioffe and Szegedy, 2015] but our network has only 2 layers while BN3 has 5 layers. We have performed extra experiments, which are not presented in this paper. These extra experiments show that the Batch Normalization operation impairs the accuracies on Dataset II and III-A and only increases the accuracies a bit on Dataset III-B. Therefore, in order to achieve higher accuracies across all three datasets obtained from different subjects, we abandon the Batch Normalization operation for our OCLNN.

6 Conclusions

In this paper, we propose a simple CNN, called OCLNN, for P300 signal detection and its application for character spelling. Our CNN learns P300-related features better by performing both spatial convolution and temporal convolution in the first layer. Compared with the state-of-the-art CNNs for P300 signal detection, our CNN has only two layers and much smaller number of parameters, which reduces the complexity significantly. Experimental results on three datasets show that our CNN always increases the P300 signal detection accuracy and increases the character spelling accuracy in most cases, when compared with the state-of-the-art methods for P300 signal detection and character spelling. Our CNN exhibits lower complexity while still achieving better accuracy, which enables the use of CNNs in resource-constrained embedded portable BCIs. In addition, our CNN can serve as a base architecture to learn low-level features. On top of it, researchers can design deep neural networks to further increase the P300 signal detection accuracy and the character spelling accuracy when the complexity is not a constraint.

References

[Abadi *et al.*, 2016] Martín Abadi, Ashish Agarwal, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[Blankertz, 2003] B Blankertz. BCI competition II webpage. *http://www.bbci.de/competition/ii/*, 2003.

[Blankertz, 2008] B Blankertz. BCI competition III webpage. *http://www.bbci.de/competition/iii/*, 2008.

[Bostanov, 2004] Vladimir 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, 2004.

[Cecotti and Graser, 2011] 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.

[Cecotti *et al.*, 2011] Hubert Cecotti, Bertrand Rivet, et al. A robust sensor-selection method for P300 brain–computer interfaces. *Journal of Neural Engineering*, 8(1):016001, 2011.

[Chen *et al.*, 2016] Tsan-Yu Chen, Chih-Wei Feng, and Wai-Chi Fang. Development of a reliable SSVEP-based BCI mobile dialing system. In *Consumer Electronics (ICCE), 2016 IEEE International Conference on*, pages 269–272. IEEE, 2016.

[Chollet and others, 2015] François Chollet et al. Keras. *https://github.com/keras-team/keras*, 2015.

[Farwell and Donchin, 1988] 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.

[Gao *et al.*, 2010] Junfeng Gao, Chongxun Zheng, and Pei Wang. Online removal of muscle artifact from electroencephalogram signals based on canonical correlation analysis. *Clinical EEG and Neuroscience*, 41(1):53–59, 2010.

[Gilroy *et al.*, 2013] Stephen William Gilroy, Julie Porteous, et al. A brain-computer interface to a plan-based narrative. In *International Joint Conference on Artificial Intelligence*, pages 1997–2005, 2013.

[Gwin *et al.*, 2010] Joseph T Gwin, Klaus Gramann, et al. Removal of movement artifact from high-density EEG recorded during walking and running. *Journal of Neurophysiology*, 103(6):3526–3534, 2010.

[He *et al.*, 2016] 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.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[Krizhevsky *et al.*, 2012] 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.

[Lin *et al.*, 2013] Chin-Teng Lin, Shu-Fang Tsai, and Li-Wei Ko. EEG-based learning system for online motion sickness level estimation in a dynamic vehicle environment. *IEEE Transactions on Neural Networks and Learning Systems*, 24(10):1689–1700, 2013.

[Lin *et al.*, 2014] Chin-Teng Lin, Bor-Shyh Lin, et al. Brain computer interface-based smart living environmental auto-adjustment control system in UPnP home networking. *IEEE Systems Journal*, 8(2):363–370, 2014.

[Liu *et al.*, 2017] Mingfei Liu, Wei Wu, Zhenghui Gu, Zhuliang Yu, FeiFei Qi, and Yuanqing Li. Deep learning based on Batch Normalization for P300 signal detection. *Neurocomputing*, 2017.

[Manor and Geva, 2015] Ran Manor and Amir B Geva. Convolutional neural network for multi-category rapid serial visual presentation BCI. *Frontiers in Computational Neuroscience*, 9, 2015.

[Mennes *et al.*, 2010] Maarten Mennes, Heidi Wouters, Bart Vanrumste, Lieven Lagae, and Peter Stiers. Validation of ICA as a tool to remove eye movement artifacts from EEG/ERP. *Psychophysiology*, 47(6):1142–1150, 2010.

[Polich, 2007] John Polich. Updating P300: an integrative theory of P3a and P3b. *Clinical Neurophysiology*, 118(10):2128–2148, 2007.

[Rakotomamonjy and Guigue, 2008] 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.

[Rivet *et al.*, 2009] Bertrand Rivet, Antoine Souloumiac, et al. xDAWN algorithm to enhance evoked potentials: application to brain–computer interface. *IEEE Transactions on Biomedical Engineering*, 56(8):2035–2043, 2009.

[Sellers and Donchin, 2006] Eric W Sellers and Emanuel Donchin. A P300-based brain–computer interface: initial tests by ALS patients. *Clinical Neurophysiology*, 117(3):538–548, 2006.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey E Hinton, et al. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Sutton *et al.*, 1967] Samuel Sutton, Patricia Tueting, et al. Information delivery and the sensory evoked potential. *Science*, 155(3768):1436–1439, 1967.

[Wang *et al.*, 2011] Yu-Te Wang, Yijun Wang, and Tzyy-Ping Jung. A cell-phone-based brain–computer interface for communication in daily life. *Journal of Neural Engineering*, 8(2):025018, 2011.

[Wolpaw and Wolpaw, 2012] Jonathan Wolpaw and Elizabeth Winter Wolpaw. *Brain-computer interfaces: principles and practice*. OUP USA, 2012.