Towards high performance and efficient brain computer interface character speller : convolutional neural network based methods

Shan, H.

Chapter 3

A Simple Convolutional Neural Network for P300 Signal Detection and Character Spelling

Hongchang Shan, Yu Liu, and Todor Stefanov,
"A Simple Convolutional Neural Network for Accurate P300 Detection and Character Spelling in Brain Computer Interface,"
In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18)*, pp. 1604-1610, Stockholm, Sweeden, July 13-19, 2018.

THE P300 speller has been the benchmark and the most-commonly used application of P300-based BCI systems [FRAG+12]. Previous research on the P300 signal detection and character spelling in the P300 speller 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 [RS+09], extracting useful features [Bos04], choosing the most relevant EEG sensors [CR+11], or removing artifacts caused by the muscle contraction [GZW10], the eye movement [MWV+10] and the body movement [GG+10]. Unfortunately, manually-designed feature extraction and traditional classification techniques have the following problems: 1) they can only learn the features that researchers focus 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 P300 signal detection and character spelling accuracy improvements.

In recent years, deep learning, especially using Convolutional Neural Networks (CNNs), has achieved significant performance improvements in the computer vision field [KSH12, SZ14, HZRS16]. Deep CNNs have the advantage of automatically learning feature representations from raw data[1]. 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 [CG11, MG15, LWG+18] and achieved better P300 signal detection and character spelling accuracy than traditional techniques. However, these CNNs have some limitations in increasing the P300 signal detection and character spelling accuracy. These CNNs first use a spatial convolution layer to learn P300-related spatial features from raw signals. Then, they use several temporal convolution layers to learn P300-related temporal features from the abstract temporal signals generated by the spatial convolution layer (the first layer). In this way, the input to the temporal convolution layers is the abstract temporal signals instead of raw temporal signals. In fact, raw temporal signals are more important to learn P300-related temporal feature. Therefore, these CNN architectures cannot learn P300-related temporal features well and this leads to problems that: 1) they prevent further P300 signal detection and character spelling accuracy improvements; 2) they require high network complexity to achieve competitive accuracy, which prevents the use of these CNNs for practical mobile-based BCIs [WWJ11, CFF16].

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 network complexity is significantly reduced while increasing the P300 signal detection accuracy, character spelling accuracy, and the communication speed. The novel contributions of this chapter 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. Our CNN exhibits much lower network complexity compared to other state-of-the-art CNNs [CG11, MG15, LWG+18] for the P300 speller.

- We perform experiments on three benchmark datasets and compare our results

[1] In this dissertation, we use "raw data, information, or signals" to denote the data that is only preprocessed (e.g., bandpass filtering and normalization) but not abstracted by a feature extraction method (e.g., a CNN).

with those in previous research works that report the best results. The comparison shows that our proposed CNN can increase the P300 signal detection accuracy with up to 14.23% and the character spelling accuracy with up to 35.49%. The comparison also shows that our proposed CNN achieves comparable communication speed with the related BN3 method [LWG+18]. Moreover, our CNN achieves higher communication speed compared to other state-of-the-art related methods [CG11, MG15, RG08, Bos04].

The rest of this chapter is organized as follows: Section 3.1 describes the related work. Section 3.2 presents our proposed CNN. Section 3.3 compares the complexity, the P300 signal detection accuracy, the character spelling accuracy, and the communication speed between the proposed CNN and other methods for P300 signal detection and character spelling. Section 3.4 ends this chapter with conclusions.

3.1 Related Work

The general architecture of the CNNs for P300-based BCI [CG11, MG15, LWG+18] uses the input tensor[2] ($N \times C$) shown in Figure 3.1, where N denotes the number of temporal signal samples and C denotes the number of sensors 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 P300-related spatial features. In the second stage, it performs convolution along time to learn P300-related temporal features. In the final stage, it uses fully-connected layers to make accurate correlation between learned features and a particular class.

Cecotti [CG11] is the first to propose the aforementioned architecture. Let us call his architecture CCNN. Table 3.1 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 of the convolution operation[3] in the convolution layers. The last column describes the number of feature maps/neurons in a layer.

Liu [LWG+18] improves CCNN by combining Batch Normalization [IS15] and Droupout [SH+14] techniques (see Table 3.2). This CNN is named BN3 in [LWG+18]. BN3 uses the Batch Normalization operation in two layers: one is in Layer 1 and the other is in Layer 3. BN3 also employs Dropout in the fully-connected layers to reduce overfitting[4]. Before the output layer, BN3 uses two fully-connected layers instead of one for better generalization and accumulation of features.

[2]The notion of a tensor is introduced in the second paragraph in Section 2.3.

[3]The convolution operation and the notion of the kernel size are introduced in Section 2.3.1.

[4]The problem of overfitting is introduced in Section 2.2.3.

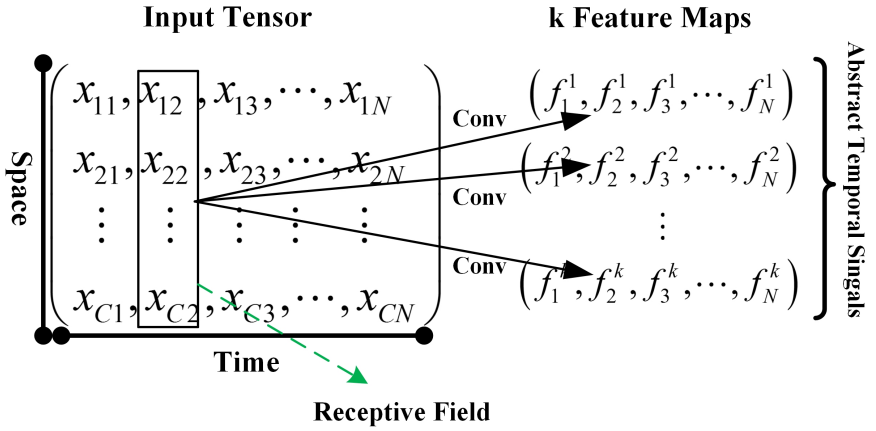

Figure 3.1: Abstraction of the raw signals in the spatial convolution layer in current CNNs. x denotes a signal sample in the input tensor. f denotes a datum in a feature map. Every column in the input tensor contains a set of C signal samples. These samples come from C sensor at a certain sampling time point. The spatial convolution operation converts each column of spatial data (receptive field) from the input tensor into an abstract datum in a feature map.

Manor [MG15] proposes a deep CNN achitecture for the P300-based BCI. Let us call his architecture CNN-R. It is shown in Table 3.3. CNN-R improves CCNN by using a deeper and wider network architecture. It uses a smaller kernel size for the temporal convolution operations but more layers for these temporal convolution operations (see Layer 2 and Layer 3 in Table 3.3). 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 (see Section 2.3.2) as well as Dropout to reduce overfitting.

The problem of the aforementioned CNNs is that they learn P300-related temporal features from abstract signals instead of raw signals, which makes these CNNs not able to learn P300-related temporal features well. P300-related temporal features are extracted by the temporal convolution layers of these CNNs. The input to these temporal convolution layers is the feature maps generated by the spatial convolution layer (the first layer). These feature maps are abstract temporal signals instead of raw signals because this spatial convolution layer converts each receptive field (see Section 2.3.2) of raw signals into an abstract datum in a feature map, as shown in Figure 3.1. These abstract temporal signals in the feature maps lose raw temporal information. Losing raw temporal information means losing important temporal features because the

Table 3.1: CCNN architecture.

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 3.2: BN3 architecture.

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

nature of P300 signals is the positive voltage potential in raw temporal information (see Figure 2.9 explained in Section 2.4.1) as well as many important P300-related features are also embodied in raw temporal information [Pol07]. As a result, these CNNs cannot learn P300-related 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. Thus, the input to this convolution layer in our CNN is raw signals. In this way, the data used to extract P300-related temporal features is raw signals instead of the abstract signals in the aforementioned CNNs. As a result, our CNN is able to learn P300-related feature representations from raw temporal information and at the same time, it can also learn P300-related spatial features. Therefore, our CNN learns P300-related temporal features better. By learning in this

Table 3.3: CNN-R architecture.

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

way, our CNN can achieve better P300 signal detection and character spelling accuracy (see Section 3.3.3 and Section 3.3.4) with only one convolution layer and without fully-connected layers before the output layer, which reduces the network complexity significantly (see Section 3.3.2).

3.2 Proposed Convolutional Neural Network

In this section, we introduce our novel CNN. We call it One Convolution Layer Neural Network (OCLNN). First, in Section 3.2.1, we describe the input to the network. Then, in Section 3.2.2, we describe our proposed network architecture. Finally, in Section 3.2.3, we explain how we train the network.

3.2.1 Input to the Network

The input to OCLNN is the input tensor ($N \times C$) shown in Figure 3.2 and Figure 3.3, where C denotes the number of sensors 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 2.4.2 and Section 2.5), and F_s denotes the signal sampling frequency.

Figure 3.2 shows that a set of signal samples from the EEG signals, introduced in Section 2.5, is preprocessed to obtain the input tensor which is used as the input to our proposed CNN. In such set of signal samples, the temporal signal samples are

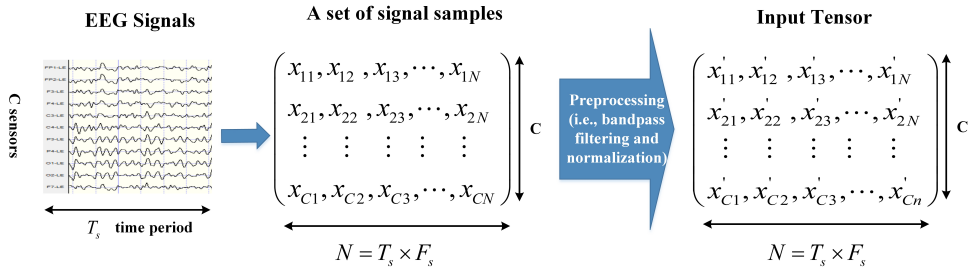

Figure 3.2: Input tensor for our proposed OCLNN.

bandpass filtered between 0.1Hz and 20Hz to remove the high frequency noise. Then, the filtered samples are normalized using Equation (3.1), (3.2), and (3.3) to have zero mean and unit variance based on each individual pattern and for each sensor. Each individual pattern represents N signal samples in the time period between 0 and T_s posterior to the beginning of each intensification. Such normalization is a common practice for preprocessing input data to CNNs. The normalization helps the CNN to perform well for the P300 signal detection and character spelling [CG11].

$$x'_{ij} = \frac{x_{ij} - \varepsilon}{\delta} \tag{3.1}$$

$$\varepsilon = \frac{1}{N}\sum_{j=1}^{N} x_{ij} \tag{3.2}$$

$$\delta = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(x_{ij} - \varepsilon)^2} \tag{3.3}$$

3.2.2 Network Architecture

The architecture of OCLNN is described in Table 3.4 and illustrated in Figure 3.3. 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 of the convolution operation 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.

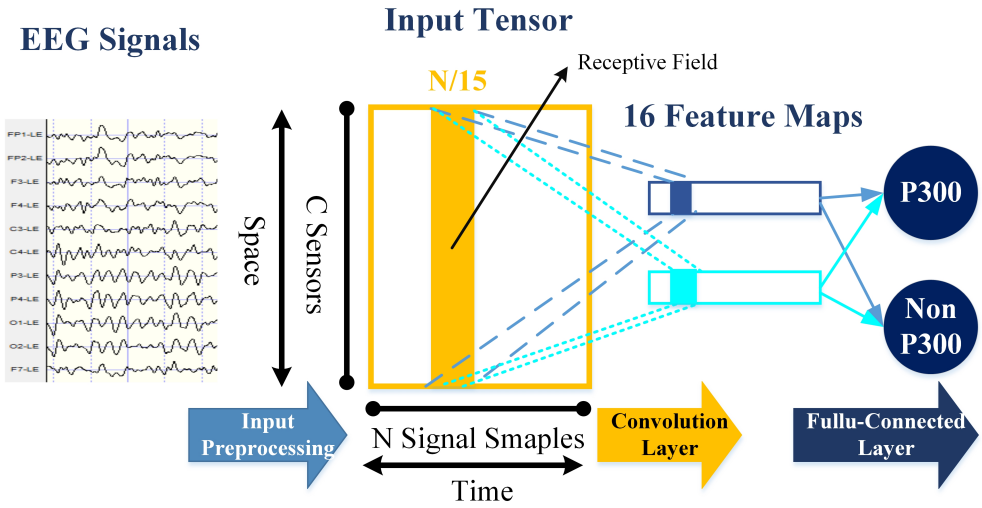

Figure 3.3: Illustration of OCLNN for P300 signal detection.

Table 3.4: OCLNN architecture.

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

In Layer 1, we segment the input tensor over the time domain 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 (see the orange rectangle in Figure 3.3) of the input tensor is a tensor (N/15, C) 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 sensors. 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. The stride[5] used for the convolution operation in this layer is (N/15, C). We use the Rectified Linear Unit (ReLU) function[6] as an activation function to model a neuron's output in this layer because a network with ReLUs is trained much faster than with other traditional activation functions [KSH12]. In this

[5]The notion of the stride is introduced in Section 2.3.1.

[6]The Rectified Linear Unit (ReLU) function is introduced in Section 2.2.1.2.

layer, we employ Dropout [SH+14] to reduce overfitting. The Dropout rate is set to be 0.25. This layer generates 16 feature maps.

In Layer Output, OCLNN performs the 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 the Softmax function[7] 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 (3.4), where $X_{(i,j)}$ is the input tensor to be classified and E_{ocl} denotes our OCLNN. By using Equation (3.4) to detect P300 signals, we can assess the performance of our proposed OCLNN in terms of the P300 signal detection accuracy (see Section 3.3.1 and Section 3.3.3).

$$E_{ocl}(X_{(i,j)}) = \begin{cases} 1 & if \quad P^1_{(i,j)} > P^0_{(i,j)} \\ 0 & otherwise \end{cases} \tag{3.4}$$

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 character matrix. For the detailed calculation process please refer to Section 2.4.2, Equation (2.30), (2.31), and (2.32).

3.2.3 Training

The training of OCLNN is carried out by minimizing the cross-entropy cost function[8]. We use a Stochastic Gradient Descent (SGD) based learning algorithm, which is a modified version of the gradient descent based learning algorithm[9]. For more details on the SGD-based learning algorithm, please refer to [Bot10]. We use the SGD-based learning algorithm with momentum[10] and with weight decay[11]. The momentum parameter μ is set to 0.9. The weight decay parameter λ is set to 0.0005. The learning rate[12] η is fixed to 0.01. The aforementioned setup of the training parameters follows the suggestion in [SZ14] because [SZ14] has shown that when using these parameters to train a CNN on a training dataset, this trained CNN is able to achieve good performance on a test dataset. For details on the training process of a neural network please refer to Section 2.2.3.

[7] The Softmax function is given in Equation (2.8) introduced in Section 2.2.1.2.

[8] The cross-entropy cost function is given in Equation (2.10) introduced in Section 2.2.3.

[9] The gradient descent based learning algorithm is introduced in Section 2.2.3.

[10] The momentum technique is given in Equation (2.23), (2.24), (2.25), and (2.26) introduced in Section 2.2.3.

[11] The weigh decay technique is given in Equation (2.27) introduced in Section 2.2.3.

[12] The notion of the learning rate is introduced in Equation (2.15) in Section 2.2.3.

3.3 Experimental Evaluation

First, we introduce our experimental setup in Section 3.3.1. Then, we show the performance comparison between OCLNN and other related research works in terms of complexity (see Section 3.3.2), P300 signal detection accuracy (see Section 3.3.3) and character spelling accuracy (see Section 3.3.4). Finally, we compare the ITR of the P300 speller based on our OCLNN and other methods (see Section 3.3.5).

3.3.1 Experimental Setup

Our OCLNN is implemented using Keras [C$^+$15] with the Tensorflow [AA$^+$16] backend. The network is trained on an NVIDIA GeForce GTX 980 Ti GPU.

We train our OCLNN using each training dataset in Dataset II, III-A and III-B, described in Section 2.5, separately, thereby obtaining three different OCLNNs with different parameters (i.e., different weights and biases). The number of used sensors is 64 and the signal sampling frequency is 240 Hz (Section 2.5). Therefore, for the input to OCLNN (see Section 3.2.1), 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, the number of temporal signal samples $N = T_s \times F_s = 240$.

We run each of the three trained OCLNNs on the corresponding test dataset in Dataset II, III-A and III-B and calculate the P300 signal detection accuracy using Equation (3.5), the character spelling accuracy using Equation (2.33) (introduced in Section 2.4.3), and the Information Transfer Rate (ITR) using Equation (2.34) and (2.35) (introduced in Section 2.4.3) for each test dataset. In Equation (3.5), acc_{P300} denotes the P300 signal 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.

$$acc_{P300} = \frac{N_{tp} + N_{tn}}{S_{pn}} \tag{3.5}$$

For a fair comparison with CNN-R [MG15], we apply the bandpass filtering methods used for our OCLNN on CNN-R because we obtain low character spelling accuracy for CNN-R using the original filtering method in [MG15].

3.3.2 Complexity

In this section, we compare the complexity, in terms of the number of parameters and layers, of OCLNN with the networks CCNN [CG11], BN3 [LWG$^+$18], and CNN-R [MG15] briefly described in Section 3.1. The number of parameters is the number

of weights and biases for all neurons in a network. We show the complexity in Table 3.5. The first row in the table lists the CNNs, we compare. The second row provides the number of parameters for each CNN. The third row shows the number of layers used in each CNN.

Table 3.5: Complexity comparison of different CNNs.

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

In terms of the number of parameters, OCLNN is much smaller than the other three CNNs. OCLNN has only 16882 parameters whereas CCNN has 37502 parameters[13], 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 whereas 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.

3.3.3 P300 Signal Detection Accuracy

This section compares the P300 signal detection accuracies achieved by OCLNN with the accuracies achieved by CCNN, BN3, and CNN-R on Dataset II, III-A and III-B.

The P300 signal detection accuracy is shown in Table 3.6. The first row in the table lists the CNNs used for comparison. The second, third, and last row show the P300 signal detection accuracy of the different CNNs on Dataset II, III-A, III-B, respectively. The numbers are given in percentage (%) and calculated using Equation (3.5). 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.

Overall, OCLNN achieves the highest accuracies among all CNNs on Dataset II, III-A and III-B. It increases the P300 signal detection accuracies achieved by the other CNNs by up to 14.23%. For Dataset II, OCLNN achieves 92.41% P300 signal detection accuracy. The accuracy achieved by OCLNN is 7.97% and 6.12% higher than the accuracy achieved by BN3 and CNN-R, respectively. For Dataset III-A, OCLNN

[13]Cecotti [CG11] calculated the number of parameters erroneously for L_2. It should be $5N_s \times (13 \times N_s + 1)$ instead of $5N_s \times (13+1)$

Table 3.6: P300 signal detection accuracy of different CNNs on Dataset II, III-A and III-B.

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

achieves 84.60% P300 signal detection accuracy. The accuracy achieved by OCLNN is 14.23%, 9.47%, and 11.54% higher than the accuracy achieved by CCNN, BN3, and CNN-R, respectively. For Dataset III-B, OCLNN achieves 86.40% P300 signal detection accuracy. The accuracy achieved by OCLNN is 8.21%, 7.38%, and 6.60% higher than the accuracy achieved by CCNN, BN3, and CNN-R, respectively.

3.3.4 Character Spelling Accuracy

This section compares the character spelling accuracies achieved by OCLNN and the accuracies achieved by CCNN, BN3, CNN-R, and ESVM [RG08] for Dataset III-A and III-B, as well as the character spelling accuracies achieved by OCLNN and the accuracies achieved by CCNN, BN3, CNN-R, and Bostanov [Bos04] 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 3.7, 3.8, and 3.9 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 (2.33) for different epoch numbers $k \in [1, 15]$. An accuracy number in bold indicates the highest accuracy along a column. "–" in a table means that the accuracy is not reported in the reference paper describing the corresponding method.

The goal of the aforementioned competitions (BCI Competition III and Competition II) is to compare which method is able to achieve the highest character spelling accuracy using all epochs (i.e., $k = 15$). For this goal, OCLNN is able to achieve the highest character spelling accuracy using all epochs for all three datasets in the two competitions. For Dataset III-A and III-B, OCLNN achieves 99% and 98% spelling accuracy for epoch number $k = 15$. For Dataset II, OCLNN only needs 3 epochs to achieve 100% spelling accuracy.

We also analyse the character spelling accuracies achieved by different methods for every epoch number $k \in [1, 15]$. Overall, in most cases, OCLNN achieves better

Table 3.7: Spelling accuracy achieved by 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	**77.42**	**90.32**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
CCNN	58.06	54.83	77.41	93.54	93.54	93.54	93.54	96.77	96.77	**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 3.8: Spelling accuracy achieved by 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	**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
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 3.9: Spelling accuracy achieved by different methods on Dataset III-B.

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
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

accuracies than the other methods. OCLNN increases the character spelling accuracies achieved by the other methods by up to 35.49%.

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 achieved by CCNN, CNN-R, BN3, and Bostanov, our OCLNN increases the accuracies with up to 35.49%, 6.45%, 19.35%, and 12.90%, respectively.

For Dataset III-A, when compared with methods CCNN, 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 achieved by CCNN, CNN-R, and ESVM, respectively.

For Dataset III-B, when compared with methods CCNN, 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 achieved by CCNN, 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 [LWG$^+$18]. 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 [IS15] but our network has only 2 layers while BN3 has 5 layers. We have done experiments to obtain the spelling accuracy achieved by OCLNN when OCLNN uses and does not use the Batch Normalization operation. These experimental results are shown in Table 3.10, 3.11, and 3.12, where OCLNN-BN denotes that our OCLNN uses the Batch Normalization operation. Table 3.12 shows that when OCLNN uses the Batch Normalization operation, the spelling accuracy is increased on epoch number k=1, 2, 3, 4, and 6. Unfortunately, Table 3.10 shows that when OCLNN uses the Batch Normalization operation, the spelling accuracy is decreased on epoch number k= 3 and 4. Table 3.11 shows that when OCLNN uses the Batch Normalization operation, the spelling accuracy is decreased on epoch number k= 8, 9, 11, 12, 13, and 15. These experimental results show that when used in our OCLNN, the Batch Normalization operation impairs the accuracies on Dataset II and III-A and only increases the accuracies 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.

3.3.5 Information Transfer Rate

This section compares the communication speed, i.e., Information Transfer Rate (ITR), of the P300 speller based on our OCLNN and other methods. ITR is calculated using

Table 3.10: Spelling accuracy achieved by OCLNN when using and not using the Batch Normalization operation 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	**77.42**	**90.32**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
OCLNN-BN	**77.42**	**90.32**	96.77	96.77	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**

Table 3.11: Spelling accuracy achieved by OCLNN when using and not using the Batch Normalization operation 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	**23**	**39**	**56**	**63**	**73**	**79**	**82**	**85**	**90**	**91**	**94**	**95**	**95**	**96**	**99**
OCLNN-BN	**23**	**39**	**56**	**63**	**73**	**79**	**82**	84	88	**91**	93	93	93	**96**	98

Table 3.12: Spelling accuracy achieved by OCLNN when using and not using the Batch Normalization operation on Dataset III-B.

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**
OCLNN-BN	**48**	**64**	**73**	**80**	**84**	**88**	**89**	**93**	**94**	**96**	**97**	**97**	**97**	**98**	**98**

Equation (2.34) and (2.35) (introduced in Section 2.4.3). The ITR of the P300 speller based on our OCLNN and other methods for Dataset II, Dataset III-A and Dataset III-B is shown in Table 3.13, 3.14, and 3.15, respectively. In these tables, the different methods, we compare, are shown in the first column. The ITR for different epoch numbers $k \in [1, 15]$ is shown in each row of a table. A number in bold denotes that the number is the highest ITR along a row. "–" in a table denotes that the ITR cannot be calculated because the corresponding paper, describing the method, does not provide the spelling accuracy. The ITR is shown in bits/minute.

In the context of ITR, i.e, the communication speed of the P300 speller, we compare the maximum ITR achieved by each method because the maximum ITR represents the maximum communication speed achieved by a method. We call this maximum ITR max-ITR. Overall, our OCLNN achieves comparable max-ITR with BN3 and higher max-ITR than the other related methods, i.e., CCNN, CNN-R, Bostanov, and ESVM. Our OCLNN can increase the max-ITR achieved by CCNN, CNN-R, Bostanov, and ESVM with up to 15.7 bits/min.

For Dataset II, shown in Table 3.13, when compared with the max-ITR achieved

Table 3.13: The ITR of the P300 speller based on different methods on Dataset II.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	**42.28**	37.74	35.25	28.46	23.86	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12
CCNN	**26.58**	16.65	22.09	24.73	20.74	17.85	15.67	14.92	13.45	13.2	12.12	11.2	10.41	9.72	9.12
CNN-R	**36.68**	33.18	30.64	26.41	23.86	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12
BN3	**42.28**	27.06	23.65	20.4	20.74	19.07	16.74	14.92	14.5	13.2	12.12	11.2	10.41	9.72	9.12
Bostanov	31.46	**33.18**	30.64	26.41	22.15	20.54	18.03	16.07	14.5	13.2	12.12	11.2	10.41	9.72	9.12

Table 3.14: The ITR of the P300 speller based on different methods on Dataset III-A.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	5.77	9.64	13.11	12.78	**13.59**	13.32	12.44	11.78	11.74	10.91	10.63	10.02	9.32	8.88	8.89
CCNN	2.96	7.33	9.91	9.41	10.18	9.7	**11.21**	10.2	10.63	9.87	9.82	9.25	8.6	8.36	8.51
CNN-R	2.28	5.56	7.03	9.7	9.13	8.99	**9.82**	9.56	9.01	9.11	9.62	8.55	7.94	8.2	8.17
BN3	5.33	9.64	13.87	**14.1**	13.59	12.22	11.69	10.86	10	9.87	9.62	9.44	9.13	8.88	8.69
ESVM	2.96	6.96	11.65	11.82	**13.28**	–	–	–	–	9.29	–	–	9.13	–	8.51

Table 3.15: The ITR of the P300 speller based on different methods on Dataset III-B.

Method	Epochs														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCLNN	18.32	**20.26**	19.62	18.45	17.15	15.68	14.32	13.82	12.71	12.06	11.3	10.44	9.71	9.27	8.69
CCNN	11.76	15.3	14.25	14.44	**15.47**	13.88	12.44	12.76	12.22	10.91	10.01	9.07	8.6	8.2	7.69
CNN-R	12.32	12.58	**17.05**	15.14	14.83	13.6	13.49	12.02	11.29	10.91	10.63	10.02	9.32	8.88	8.33
BN3	**18.97**	18.72	18.75	16.2	14.51	14.17	12.96	13.28	12.71	11.81	10.84	10.02	9.13	8.53	8.17
ESVM	11.76	**15.78**	15.43	14.44	14.2	–	–	–	–	10.91	–	–	9.51	–	8.33

by CCNN, CNN-R, and Bostanov, our OCLNN achieves the highest max-ITR. OCLNN increases the max-ITR achieved by CCNN, CNN-R, and Bostanov with 15.7 bits/min, 5.6 bits/min, and 9.1 bits/min, respectively.

For Dataset III-A, shown in Table 3.14, when compared with the max-ITR achieved by CCNN, CNN-R, and ESVM, our OCLNN achieves the highest max-ITR. OCLNN increases the max-ITR achieved by CCNN, CNN-R, and ESVM with 2.38 bits/min, 3.77 bits/min, and 0.31 bits/min, respectively.

For Dataset III-B, shown in Table 3.15, when compared with the max-ITR achieved by CCNN, CNN-R and ESVM, our OCLNN achieves the highest max-ITR. OCLNN increases the max-ITR achieved by CCNN, CNN-R, and ESVM with 4.79 bits/min

and 3.21 bits/min, and 4.48 bits/min, respectively.

When compared with BN3, on Dataset III-B, our OCLNN increases the max-ITR achieved by BN3 with 1.29 bits/min. On Dataset II, our OCLNN achieves the same max-ITR with BN3. However, on Dataset III-A, our OCLNN decreases the max-ITR achieved by BN3 with 0.51 bits/min. The reason is the following. On Dataset III-A, BN3 achieves the max-ITR 14.1 bits/min on epoch number $k = 4$ (see Table 3.14). On epoch number $k = 4$, our OCLNN achieves lower spelling accuracy than BN3 (see Table 3.8). The reason for this has been explained in the last paragraph in Section 3.3.4. Thus, our OCLNN cannot achieve the same max-ITR as BN3 does on Dataset III-A. The discussion above shows that overall, our OCLNN achieves comparable max-ITR with BN3. On the other hand, considering that the complexity of our OCLNN is much lower compared to the complexity of BN3 (see Table 3.5 in Section 3.3.2), our OCLNN could be considered better than BN3 for the P300 speller.

In Section 2.4.3, we have shown that when using Dataset II, III-A, and III-B, the theoretically achievable maximum ITR is 67.43 bits/min. The maximum ITR (i.e., max-ITR) achieved by our OCLNN is 42.28 bits/min, 13.59 bits/min, and 20.26 bits/min on Dataset II, III-A, and III-B, respectively. The maximum ITR achieved by our OCLNN still cannot reach the theoretically achievable maximum ITR. Thus, further research efforts are needed to find approaches to increase the maximum ITR of a P300 speller in order to bring it closer to the theoretically achievable maximum ITR.

3.4 Conclusions

In this chapter, we propose a simple CNN, called OCLNN, for P300 signal detection and character spelling in the P300 speller. 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. In terms of the communication speed, our OCLNN achieves comparable communication speed with BN3 and higher communication speed than the other state-of-art methods.