applied sciences

MDPI

Article

Transposed Convolution as Alternative Preprocessor for Brain-Computer Interface Using Electroencephalogram

Kenshi Machida [1,*], Isao Nambu [2] and Yasuhiro Wada [2]

[1] Department of Science of Technology Innovation, Nagaoka University of Technology, 1603-1 Kamitomioka, Nagaoka 940-2188, Japan
[2] Department of Electrical, Electronics and Information Engineering, Nagaoka University of Technology, 1603-1 Kamitomioka, Nagaoka 940-2188, Japan
* Correspondence: kmachida+n.utneuriplab@stn.nagaokaut.ac.jp

Abstract: The implementation of a brain–computer interface (BCI) using electroencephalography typically entails two phases: feature extraction and classification utilizing a classifier. Consequently, there are numerous disordered combinations of feature extraction and classification techniques that apply to each classification target and dataset. In this study, we employed a neural network as a classifier to address the versatility of the system in converting inputs of various forms into outputs of various forms. As a preprocessing step, we utilized a transposed convolution to augment the width of the convolution and the number of output features, which were then classified using a convolutional neural network (CNN). Our implementation of a simple CNN incorporating a transposed convolution in the initial layer allowed us to classify the BCI Competition IV Dataset 2a Motor Imagery Task data. Our findings indicate that our proposed method, which incorporates a two-dimensional CNN with a transposed convolution, outperforms the accuracy achieved without the transposed convolution. Additionally, the accuracy obtained was comparable to conventional optimal preprocessing methods, demonstrating the effectiveness of the transposed convolution as a potential alternative for BCI preprocessing.

Keywords: BCI competition IV; brain–computer interface; convolutional neural network; electroencephalogram; transposed convolution

Citation: Machida, K.; Nambu, I.; Wada, Y. Transposed Convolution as Alternative Preprocessor for Brain-Computer Interface Using Electroencephalogram. *Appl. Sci.* **2023**, *13*, 3578. https://doi.org/10.3390/app13063578

Academic Editors: Francesco Donnarumma, Francesco Isgrò and Roberto Prevete

Received: 13 February 2023
Revised: 4 March 2023
Accepted: 9 March 2023
Published: 10 March 2023

1. Introduction

Most studies on electroencephalograms (EEG) in the computer science field have focused on developing an interface called the brain–computer interface (BCI). The objective of a brain–computer interface (BCI) is to establish a direct pathway for the brain to communicate with the outside world, circumventing conventional sensory and motor pathways. After detecting brain activity, the BCI system utilizes algorithms and machine learning methodologies to decipher the signals and transform them into commands or actions that can operate external devices. The BCI is expected to be used in many applications such as supporting people with motor disabilities and providing a third (robotic) arm to a person [1–4]. The application of EEG classification techniques for the purpose of achieving highly accurate BCI has been extensively explored in various studies. One notable example of such studies is the classification of datasets from BCI Competitions, with the most recent dataset from BCI Competition IV. Since the publication of [5], various methods demonstrating improved accuracy in classification have been reported. For instance, Ref. [6] performed classification using a Naive Bayesian Parzen Window classifier after applying a filter bank common spatial pattern, as described by Ang et al. Meanwhile, Gaur et al. [7] employed a filtering method using multivariate empirical mode decomposition and executed classification by utilizing the minimum distance to the Riemannian mean method. When conducting EEG analysis, a two-step process is commonly used [8], which involves the steps of feature extraction followed by classification, as demonstrated in the

previously mentioned studies. As a result, a wide range of combinations of preprocessing and classification methods have been employed, leading to inconsistencies within each classification target. The effectiveness of the selected feature extraction and classification method is contingent on the data, which raises the question of whether the optimal method has been applied to BCI data.

The implementation of neural networks has become increasingly popular in recent times, given their ability to convert inputs into outputs, making them suitable for any classification problem. Furthermore, the development of specialized processing hardware for neural networks has enabled their integration into small devices such as smartphones [9].

The utilization of neural networks, including convolutional neural networks (CNN), may hold the potential for creating highly effective BCI using modern technology [10]. Several studies have leveraged the use of CNN in EEG classification algorithms for BCI research, including EEGNet [11]. The application of EEGNet to the BCI competition dataset results in accuracy comparable to that achieved by conventional methods. In contrast to conventional EEG classification methods, which often rely on feature extraction and classification, neural networks have the ability to convert any input into any output, thus eliminating the need for feature extraction and allowing the direct entry of EEG signals into the classification model.

In this study, we propose a neural network with transposed convolution for EEG analysis. Transposed convolution is a type of convolution that performs the inverse operation of a normal convolution. It is used to extract components from the convolved data by decomposing them. In other words, our method includes processing and feature extraction similar to window analysis (such as the filter bank [6], wavelet [12], or short-time Fourier transform (STFT) [13]) that has been used in many conventional EEG analyses. We demonstrate that conventional feature extraction can be replaced by the proposed method. In addition, a high classification accuracy can be obtained without explicit feature extraction. The concept of the proposed method has been reported in our previous study [14]. Here, we further explore the effectiveness of the proposed method.

2. Materials and Methods

Neural networks possess a remarkable degree of versatility, capable of processing a variety of data types, including time-series signals and images, and providing adjustable degrees of freedom in calculations. However, overfitting can be a common issue and the network's configuration often necessitates the fine-tuning of multiple hyperparameters. The learning process can also be complicated by the need to consider techniques such as adjusting the learning rate and implementing early stopping. These challenges can be mitigated by utilizing recent advancements in the field.

The proposed method incorporates the use of convolution and transposed convolution and employs batch normalization and global average pooling to suppress overfitting [13,15].

In order to assess the efficacy of incorporating transposed convolution, the proposed method was compared to several other methods, including those utilizing CNN and STFT, by evaluating their classification accuracy against two dimensional (2D) CNN with transposed convolution (proposed), one dimensional (1D) CNN, 2D CNN for time-series and electrode, and 2D CNN for STFT data.

2.1. Convolutional Layer

The convolution layer has spatial dimensions (in which convolution is performed in the feature space) and a channel dimension (in which the number of spaces can be converted between input and output). When the data are n-spatial dimensional, the input/output is $n + 1$ dimensional, which encompasses the spatial and channel dimensions. The kernel used for convolution is $n + 2$ dimensional, including the spatial dimension number n, an input channel dimension, and an output channel dimension [16]. For images, there are two spatial dimensions and the red–green–blue channel dimension (three dimensions in total). The kernel for images has four dimensions: the two-spatial dimensions, red–green–blue

(input) channel dimension, and output channel dimension. EEG data contain time-series (time domain) and electrodes, and the dimensional treatment of the data is described in each method in the "Network structure" section. An illustration of 2D convolution is shown in Figure 1 (note that the channels are ignored for the illustration purpose).

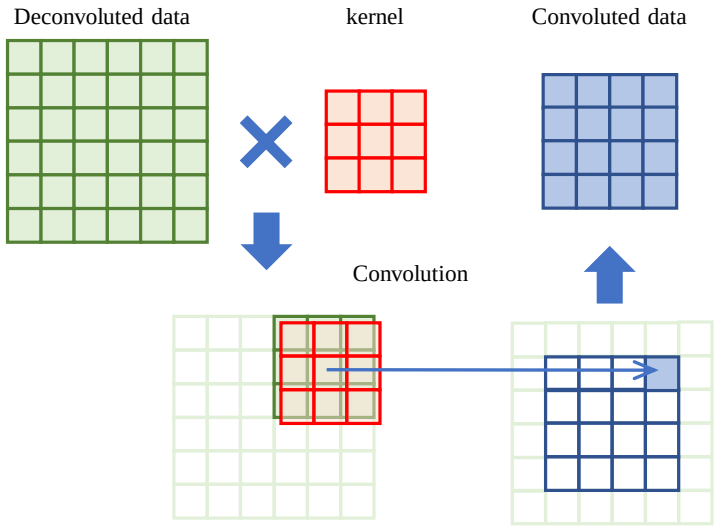

Figure 1. 2D convolution architecture.

The convolution layer is utilized to process inputs that have one or more features with spatial dimensions. A kernel is employed to perform convolution along the spatial dimensions of the features. The utilization of the convolution layer regularizes the weight parameters due to its smaller number of weight parameters as compared to the number of input–output features. As EEG signals are time series in nature, the consideration of regularization parameters is less crucial, given that the convolution layer operates on the time domain.

For the purpose of this study, the terms "spatial" and "channel" dimensions are used in their general sense to refer to dimensions, rather than implying any specific spatial information of EEG electrodes (EEG channels).

2.2. Transposed Convolutional Layer

The transposed convolution operates in the opposite direction of a convolution and functions as a decoder, whereas a convolution acts as an encoder [16]. Illustration of 2D transposed convolution is shown in Figure 2 (note that the channels are ignored for the illustration purpose).

Originally, transposed convolution performs convolution and up-sampling simultaneously. However, when applying transposed convolution to a new signal feature dimension, it expands the number of dimensions and projects the dimensions into a larger dimensional space with features. The number of dimensions must be expanded as a parameter. However, because the projection to the feature dimension is learned from each data point, the feature amount does not need to be selected in advance, as in conventional EEG research.

Transposed convolution using the convolution kernel $K[W, R, Ch_{in}, Ch_{out}]$ for the signal $S[Time, Ch_{in}]$, sample time t and EEG electrodes Ch_{in} follows (1). Here, W is the convolution width, R is the number of new feature dimensions to be expanded, and Ch_{out} is the number of output channels of the transposed convolution (Figure 3).

$$(S * K)(t, r, ch) = \sum_{w}^{W} \sum_{c}^{Ch_{in}} S(t - w, c) K(w, r, c, ch) \quad (1)$$

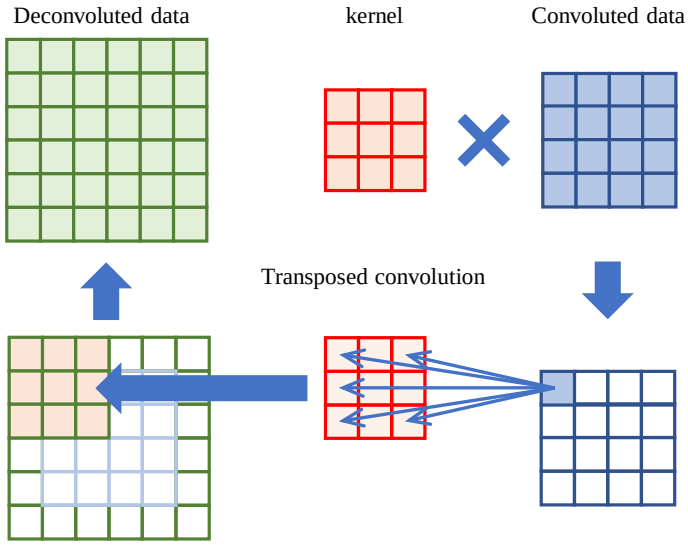

Figure 2. 2D transposed convolution architecture.

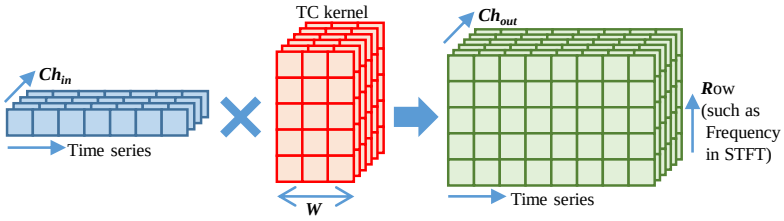

Figure 3. Transposing convolution 1D to 2D.

2.3. Batch Normalization

Batch normalization [17] is a technique employed in deep learning to normalize the inputs to each layer. This technique is designed to ensure that the mean of the inputs is zero and the variance is one within each mini-batch.

According to [18], a higher learning rate leads to more regularization of the data due to the increase in noise that results from the selection of mini-batches. Without normalization, the mean and variance of the inputs can rapidly increase as the depth of the layer increases, leading to a situation where the gradient becomes heavily dependent on the inputs. This can lead to divergence of the error when a higher learning rate is used. Batch normalization helps to maintain a constant mean and variance, thereby avoiding such issues and promoting stability in the learning process, even when the higher learning rate is used.

2.4. Global Average Pooling

Global average pooling outputs the average in the spatial direction for each CNN channel [19]. Therefore, when the signal time is arranged in the CNN dimensional space, the components can be classified regardless of the time zone of the input data. Furthermore, the feature can be extracted at any position on the extended axis (frequency or channel axis) without being dependent on the spatial position. This also prevents overfitting because the number of dimensions can be significantly reduced.

2.5. Network Structure

We constructed the following models and evaluated their classification performance. The codes were written in Python (version 3.6.9) and Keras on TensorFlow (version 2.4.1).

2.5.1. 2D CNN with Transposed Convolution (Proposed Method)

We allocated the time domain of EEG to the spatial dimension of the convolution, expanded it to two dimensions using transposed convolution, and then performed classification using a CNN with two spatial dimensions (Figure 3). The proposed model structure is presented in Table 1. We used the following parameters for the transposed convolution; $W = 8$, $R = 150$ and $Ch_{out} = 8$.

Table 1. The model structure of 2D CNN incorporating a transposed convolution.

Layers	CNN with Transposed Convolution Model Structure	Output Shape
Input	**-**	**(250, 1, 22)**
1	TransposedConv. [8, 150] 8 ch BatchNormalization	(257, 150, 8)
2	Convolution [3, 3] 8 ch BatchNormalization LeakyReLU (α = 0.2)	(255, 148, 8)
3	Convolution [3, 3] 16 ch BatchNormalization LeakyReLU (α = 0.2)	(253, 146, 16)
4	Convolution [3, 3] 32 ch BatchNormalization LeakyReLU (α = 0.2)	(251, 144, 32)
5	Convolution [3, 3] 32 ch BatchNormalization LeakyReLU (α = 0.2)	(249, 142, 32)
6	Global Average Pooling	(32)
7	Dense [4] Softmax	(4)

2.5.2. 1D CNN

To make comparisons with the case of classification without dimensional expansion, we allocated the time domain of the EEG to the spatial dimension, and then performed classification using a CNN with one spatial dimension. The model structure is presented in Table 2. The input size is the same as that of the proposed method (2D CNN with transposed convolution).

2.5.3. 2D CNN with Time and EEG Electrodes

Since EEG is a time-series signal, it is reasonable to perform convolution only in the time-series dimension. However, here we treat it as two spatial dimensions: time-series and EEG electrodes. The purpose is to confirm which is more effective: a simple two-spatial dimension convolution or a two-spatial dimension convolution of the time-series with the expanded dimension. We allocated the time domain and electrodes of the EEG to the spatial dimension of the convolution and performed the classification using a CNN with two spatial dimensions. The model structure of the 2D CNN is presented in Table 3.

Table 2. The model structure of 1D CNN.

Layers	1D CNN Model Structure	Output Shape
Input	**-**	**(250, 1, 22)**
1	Convolution [3, 1] 22 ch BatchNormalization LeakyReLU (α = 0.2)	(248, 1, 22)
2	Convolution [3, 1] 22 ch BatchNormalization LeakyReLU (α = 0.2)	(246, 1, 22)
3	Convolution [3, 1] 32 ch BatchNormalization LeakyReLU (α = 0.2)	(244, 1, 32)
4	Convolution [3, 1] 32 ch BatchNormalization LeakyReLU (α = 0.2)	(242, 1, 32)
5	Global Average Pooling	(32)
6	Dense [4] Softmax	(4)

Table 3. The model structure of Time and EEG electrodes axis 2D CNN.

Layers	Time and EEG Electrodes 2D CNN Model Structure	Output Shape
Input	**-**	**(250, 22, 1)**
1	Convolution [3, 3] 8 ch BatchNormalization LeakyReLU (α = 0.2)	(248, 20, 8)
2	Convolution [3, 3] 16 ch BatchNormalization LeakyReLU (α = 0.2)	(246, 18, 16)
3	Convolution [3, 3] 32 ch BatchNormalization LeakyReLU (α = 0.2)	(244, 16, 32)
4	Convolution [3, 3] 32 ch BatchNormalization LeakyReLU (α = 0.2)	(242, 14, 32)
5	Global Average Pooling	(32)
6	Dense [4] Softmax	(4)

2.5.4. 2D CNN with STFT

To make comparisons with the existing two-step process using the two-dimensional expansion method, we extracted the features by STFT and classified them by CNN. STFT was applied to EEG, and the time and frequency domains were assigned to the spatial dimensions of the convolution. The classification was performed using a CNN with two spatial dimensions. The model structure is presented in Table 4. The STFT parameters are described below.

Table 4. The model structure 2D CNN after STFT.

Layers	2D CNN Model after STFT	
	Structure	**Output Shape**
Input	**-**	**(251, 80, 22)**
1	Convolution [3, 3] 8 ch BatchNormalization LeakyReLU (α = 0.2)	(249, 78, 8)
2	Convolution [3, 3] 16 ch BatchNormalization LeakyReLU (α = 0.2)	(247, 76, 16)
3	Convolution [3, 3] 32 ch BatchNormalization LeakyReLU (α = 0.2)	(245, 74, 32)
4	Convolution [3, 3] 32 ch BatchNormalization LeakyReLU (α = 0.2)	(243, 72, 32)
5	Global Average Pooling	(32)
6	Dense [4] Softmax	(4)

2.6. Dataset

For the purpose of evaluation, we utilized the BCI competition IV dataset 2a [20]. This dataset comprises EEG data measured from nine subjects while they performed four distinct types of motor imagery tasks involving the foot, tongue, left hand, and right hand. The recordings were taken using 22 EEG electrodes at a sampling frequency of 250 Hz. A total of 288 trials, including missing parts and excluded trials, were recorded for each subject for both training and testing purposes.

As the input, an EEG signal was used for 4 s (1000 samples) for the duration of "cue" to "motor imagery."

When inputting the EEG time signal, it was downsampled to 62.5 Hz and then normalized to a mean of 0 and variance of 1 for each trial and electrode before being inputted in to the network.

When inputting the power spectrum using STFT, STFT was applied with a window width of 250 samples and step width of 3 samples after normalization with a mean of 0 and variance of 1 for each trial and channel (without downsampling). A frequency range of $0 < f \leq 80$ was used as the input.

2.7. Training and Evaluation

For the purpose of classifier training, the cross-entropy loss function was employed. The optimization algorithm used was Adam [21], with a mini-batch size of 58 and a learning rate of 0.0001. The parameters were updated a total of 1000 times, without alteration to the learning rate or cessation of the training process.

We conducted two types of evaluations: cross-validation within the BCI competition dataset training data, and unseen test data evaluation. In the cross-validation within the training data, a 5-fold cross-validation was conducted 20 times for each individual. This evaluation examined the performance (accuracy) by eliminating the effect of dataset selection. Only the original training data of the BCI competition dataset (288 trials for each subject) were used. In the unseen test data evaluation, the classifier was trained using the training data (288 trials for each subject), and the classification accuracy of the test data (288 additional trials for the same subject) was determined. This procedure was repeated 100 times by changing the initial weights in the model, and the average accuracy was

calculated. The data used for the training and test data evaluations were unchanged. In addition to the five-fold cross-validation accuracy and unseen test evaluation data accuracy, we calculated the kappa values and compared the results of our proposed method with those reported in previous studies. Since the numbers of samples in this data set are equally distributed across classes, using the number of classlabels M, or the chance level p_e (reciprocal of M), the kappa value κ can be expressed by the following equation [22]:

$$\kappa = \frac{Accuracy - p_e}{1 - p_e} = \frac{M \times Accuracy - 1}{M - 1}. \quad (2)$$

To check the statistical significance of the accuracy improvement, we performed a Wilcoxon signed-rank test and calculated the p-values between the transposed convolution model and the other methods. A p-value of 0.05 was assumed to be the threshold for assessing statistical significance.

3. Results

3.1. Comparison among the Classification Methods

To verify the effects of transposed convolution, we compared four classification methods: 2D CNN with transposed convolution (proposed method), 1D CNN, 2D CNN for time-series using electrode data and 2D CNN for using STFT data.

Tables 5 and 6 display the comparisons among the results for the four network types. The p-value indicates the results of the Wilcoxon signed-rank test.

The means and standard deviations (SD) of the accuracies of 100 times in total (20 times 5-fold cross-validations) for each subject are displayed in Table 5. For cross-validation, the average accuracy across subjects was approximately 0.77 when transposed convolution was used, and 0.42 to 0.65 in the other cases. When calculating the information transfer rate (ITR) [8], the results were as follows: 17.30 for the model using transposed convolution, 10.36 for the 1D CNN, 5.50 for the 2D CNN using STFT, and 2.18 for the 2D CNN using time-series electrode data.

Table 6 displays the average and SD of the learning accuracy using all training data and evaluating the unseen test data 100 times. When the unseen data were classified, the average accuracy was approximately 0.71 when transposed convolution was used and 0.43 to 0.62 in the other cases. When calculating the information transfer rate (ITR), the results were as follows: 13.82 for the model using transposed convolution, 9.08 for the 1D CNN, 4.23 for the 2D CNN using STFT, and 2.27 for the 2D CNN using time-series electrode data. The classification accuracy of each subject was the highest when transposed convolution was used in both cases. The accuracy of the proposed method was significantly higher ($p < 0.05$) than that of the other methods.

These results indicate that by expanding the feature dimension with transposed convolution, improved features are automatically extracted.

3.2. Comparison with Previous Studies

Next, we compared the results of the proposed method with those obtained from previous studies.

Table 7 displays a comparison of the kappa values obtained by converting the average accuracy of the model using transposed convolution and those of past studies performing the classification using BCI competition IV dataset 2a. The mean kappa value for the proposed method across subjects was 0.62, which is the highest classification accuracy compared with that of the other studies. However, a statistically significant difference between the proposed and previous methods was only found for the method in [23] and not for the method in [7] and that in [6].

Table 5. Cross-validation accuracy.

Subject	Five-Fold Cross Validation Accuracy			
	Transposed Convolution	**1D CNN**	**2D by STFT**	**Time and ch**
	Mean ± SD	**Mean ± SD**	**Mean ± SD**	**Mean ± SD**
1	0.8314 ± 0.0489	0.7144 ± 0.0540	0.5770 ± 0.0631	0.5521 ± 0.0702
2	0.6407 ± 0.0708	0.4965 ± 0.0680	0.4516 ± 0.0649	0.3204 ± 0.0580
3	0.8942 ± 0.0396	0.8232 ± 0.0491	0.6863 ± 0.0608	0.5186 ± 0.0733
4	0.6618 ± 0.0699	0.4879 ± 0.0682	0.3693 ± 0.0657	0.2967 ± 0.0609
5	0.6500 ± 0.0651	0.5514 ± 0.0639	0.3674 ± 0.0589	0.2784 ± 0.0524
6	0.6440 ± 0.0687	0.4947 ± 0.0609	0.4123 ± 0.0631	0.3207 ± 0.0641
7	0.9367 ± 0.0338	0.7961 ± 0.0577	0.6596 ± 0.0640	0.5105 ± 0.0675
8	0.8965 ± 0.0357	0.8149 ± 0.0503	0.6942 ± 0.0571	0.5567 ± 0.0797
9	0.7947 ± 0.0457	0.6949 ± 0.0570	0.6388 ± 0.0622	0.5018 ± 0.0623
Average	0.7722	0.6527	0.5396	0.4284
p-value	-	<0.05	<0.05	<0.05

CNN: convolutional neural network, STFT: short-time Fourier transform, SD: standard deviations.

Table 6. Unseen evaluation data accuracy.

Subject	Evaluation Accuracy			
	Transposed Convolution	**1D CNN**	**2D by STFT**	**Time and ch**
	Mean ± SD	**Mean ± SD**	**Mean ± SD**	**Mean ± SD**
1	0.8654 ± 0.0216	0.7571 ± 0.0208	0.6039 ± 0.0278	0.6041 ± 0.0356
2	0.4757 ± 0.0309	0.3979 ± 0.0263	0.3419 ± 0.0185	0.3194 ± 0.0222
3	0.9061 ± 0.0116	0.8182 ± 0.0204	0.6474 ± 0.0354	0.5109 ± 0.0364
4	0.6715 ± 0.0446	0.4916 ± 0.0308	0.4174 ± 0.0234	0.3382 ± 0.0272
5	0.5468 ± 0.0392	0.4956 ± 0.0288	0.2789 ± 0.0239	0.2598 ± 0.0251
6	0.4987 ± 0.0444	0.4569 ± 0.0326	0.3451 ± 0.0295	0.3149 ± 0.0266
7	0.8769 ± 0.0227	0.7351 ± 0.0299	0.6519 ± 0.0304	0.4539 ± 0.0399
8	0.8202 ± 0.0149	0.7443 ± 0.0221	0.5998 ± 0.0334	0.5049 ± 0.0470
9	0.7886 ± 0.0227	0.7395 ± 0.0235	0.6362 ± 0.0279	0.5835 ± 0.0458
Average	0.7167	0.6262	0.5025	0.4322
p-value	-	<0.05	<0.05	<0.05

CNN: convolutional neural network, STFT: short-time Fourier transform, SD: standard deviations.

Table 7. Comparisons of the evaluated kappa values with previous studies.

Subject	Evaluation Kappa Value			
	Transposed Convolution	**Method-[7]**	**Method-[23]**	**Method-[6] PW**
1	0.821	0.86	0.75	0.782
2	0.301	0.24	0.37	0.407
3	0.875	0.70	0.66	0.755
4	0.562	0.68	0.53	0.528
5	0.396	0.36	0.29	0.417
6	0.332	0.34	0.27	0.185
7	0.836	0.66	0.56	0.796
8	0.760	0.75	0.58	0.741
9	0.718	0.82	0.68	0.537
Average	0.622	0.60	0.52	0.572
p-value	-	0.6523	<0.05	0.0977

3.3. Parameter Search of Transposed Convolution

We examined the effects of parameter selection in transposed convolution. The structure of the network performing the parameter search of the transposed convolution is presented in Table 8. W is the convolution width in the time direction, R is the dimension extended by transposed convolution, and Ch_{out} is the number of output channels after convolution. Five-fold cross-validation was performed using the training data (without repetition) where $W = [4, 8, 12, 16]$, $R = [50, 100, 150, 200, 250]$, and $Ch_{out} = [4, 8, 12, 16]$ (only combinations where $W \times R \times Ch_{out} \leq 16,000$ were tested because of the computational costs).

Tables 9–12 illustrate the mean accuracies of each parameter. The accuracy of the proposed method was higher than that of the other methods (Table 5), irrespective of the parameter values. When $W = 4$, the overall accuracy was low. In addition, when $R = 50$, the accuracy tended to be lower than that when R was higher. The maximum value occurred when $W = 16$, $R = 250$, and $Ch_{out} = 4$, but its value was high primarily near $W = 12$, $R = 150$, and $Ch_{out} = 4$. Therefore, the optimal parameters were considered to be approximately $W = 12$, $R = 150$, and $Ch_{out} = 4$.

Table 8. Model structure for parameter search.

Layers	**Transposed Convolution Model**
Input	**[250 (time), 1, 22 (Ch_{in})]**
1	Transposed Conv [W, R] Ch_{out} ch BatchNormalization Leaky ReLU (α = 0.2)
2	Convolution [3, 3] 8 ch BatchNormalization LeakyReLU (α = 0.2)
3	Convolution [3, 3] 16 ch BatchNormalization LeakyReLU (α = 0.2)
4	Convolution [3, 3] 32 ch BatchNormalization LeakyReLU (α = 0.2)
5	Convolution [3, 3] 32 ch BatchNormalization LeakyReLU (α = 0.2)
6	Global Average Pooling
7	Dense [4] Softmax

Table 9. Accuracy of parameter search at Ch_{out} = 4 ch.

Ch_{out} = 4 ch		**W**			
		4	**8**	**12**	**16**
R	50	0.7427	0.7719	0.7750	0.7770
	100	0.7466	0.7696	0.7766	0.7821
	150	0.7411	0.7782	0.7871	0.7875
	200	0.7481	0.7793	0.7836	0.7704
	250	0.7622	0.7813	0.7747	0.7891

Table 10. Accuracy of parameter search at Ch_{out} = 8 ch.

Ch_{out} = 8 ch		W			
		4	**8**	**12**	**16**
R	50	0.7326	0.7641	0.7571	0.7747
	100	0.7559	0.7587	0.7673	0.7606
	150	0.7513	0.7754	0.7762	
	200	0.7419	0.7704		
	250	0.7493	0.7774		

The diagonal line represents data not tested because of the computational costs.

Table 11. Accuracy of parameter search at Ch_{out} = 12 ch.

Ch_{out} = 12 ch		W			
		4	**8**	**12**	**16**
R	50	0.7575	0.7618	0.7587	0.7747
	100	0.7450	0.7731	0.7801	
	150	0.7392	0.7622		
	200	0.7528			
	250	0.7618			

The diagonal line represents data not tested because of the computational costs.

Table 12. Accuracy of parameter search at Ch_{out} = 16 ch.

Ch_{out} = 16 ch		W			
		4	**8**	**12**	**16**
R	50	0.7474	0.7556	0.7727	0.7684
	100	0.7466	0.7684		
	150	0.7505			
	200	0.7505			
	250	0.7485			

The diagonal line represents data not tested because of the computational costs.

4. Discussion

In this study, we evaluated the effectiveness of transposed convolution in classifying BCI data and compared the results with previous studies.

As a measure of accuracy, we employed Cohen's kappa, which has been widely used in prior studies as a simple 0–1 metric that provides a linear mapping of the classification accuracy from chance level to 100%. Thus, comparing kappa values is equivalent to comparing the classification accuracy. Although the EEGNet paper does not report a specific accuracy metric, our results suggest that the approximate accuracy of the readings, when expressed as a kappa value, is slightly lower than that of the conventional method. However, by transforming EEG data into a 2D map using transposed convolution, we achieved a high level of accuracy comparable to previous methods, demonstrating the efficacy of the transposed convolution approach.

The conventional method for transforming a signal into a 2D map is the Short-Time Fourier Transform (STFT). STFT is commonly employed in EEG analysis, and it extends the frequency axis by a number equal to the predetermined window width. However, STFT only extracts frequency information at equal intervals, with no guarantee of the validity of

the information in this range. The proposed method employs transposed convolution to increase the signal along a different axis direction with a predetermined window width. The integration of preprocessing into the neural network framework enables the determination of the window width and the number of extensions to be made arbitrarily, thus facilitating the learning of features that are instrumental for successful classification.

The reason we used transposed convolution is to extract features from EEG by unfolding it. Transposed convolution is designed to perform the inverse operation of normal convolution and is expected to decompose complexly convolved data. We found that CNN using transposed convolution can classify data more accurately than CNN without using it or using STFT data for classification. Therefore, we concluded that a kernel that unfolds EEG while capturing its features has been learned through the use of transposed convolution. In addition, a structural parameter search was performed to investigate how accuracy is affected by each parameter of transposed convolution. We found that the classification accuracy was lower when W and R were small, whereas Ch_{out} had a higher accuracy when its value was small (Tables 9–12). Here, we further discuss these parameter sets. First, R is the number of extended dimensions. When R was too small, the classification accuracy was believed to be low because the probability of appearance of the features necessary for classification was low. Generally, the weights of the convolutional layers tended to be regularized. Therefore, when the convolution width W was too small, the regularization effect was significant, and the accuracy was considered to have decreased. In the convolutional layer, the regularizing effect of the number of channels was larger than the convolutional width. When Ch_{out} was large, overfitting was considered to have occurred.

Certain issues exist in constructing a neural network model when incorporating transposed convolution into analysis. One crucial aspect is the selection of a proper learning rate, as batch normalization is utilized in each layer of the network. Despite batch normalization helping to prevent weight parameter divergence, an excessively high learning rate can result in excessive regularization, hindering the improvement of accuracy. Observing the trend of the validation loss during the learning process, it can be seen that it tends to reduce towards the end of learning. Hence, the accuracy can be optimized by adjusting the learning rate at each validation or adapting it during the learning epoch and selecting the best set of parameters for transposed convolution.

The discussion on how the feature extraction occurs and the types of features extracted by transposed convolution would be interesting; however, it is very difficult to observe and evaluate the inner workings of a neural network structure in our method because of its complexity. In a future work, visualizing the parameters and outputs of each layer can help us analyze the source components necessary for classification in the signal and improve our understanding of EEG.

5. Conclusions

We aimed to investigate the possibility of transforming preprocessing by utilizing transposed convolution on EEG signals with varying window widths and spatial sizes, followed by a network of simple structures suitable for a 2D CNN. Our results indicate that a relatively straightforward neural network can effectively replace the complex preprocessing of EEG data, leading to a high degree of accuracy. Furthermore, the use of transposed convolution in our approach is not limited to BCI data and has the potential to be applied to other future datasets.

Author Contributions: Conceptualization, K.M.; methodology, K.M.; software, K.M.; validation, K.M. and I.N.; formal analysis, K.M.; investigation, K.M.; resources, I.N. and Y.W.; data curation, K.M.; writing—original draft preparation, K.M.; writing—review and editing, I.N.; supervision, I.N. and Y.W.; project administration, I.N. and Y.W.; funding acquisition, I.N. and Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Japan Society for the Promotion of Science KAKENHI (grant numbers 21H03480, 21H18304, and 22H04772), KDDI Foundation, and Nagaoka University of Technology Presidential Research Grant.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: BCI Competition IV dataset 2a http://www.bbci.de/competition/iv/, accessed on 1 April 2019.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bonci, A.; Fiori, S.; Higashi, H.; Tanaka, T.; Verdini, F. An Introductory Tutorial on Brain–Computer Interfaces and Their Applications. *Electronics* **2021**, *10*, 560. [CrossRef]
2. Schalk, G.; McFarland, D.J.; Hinterberger, T.; Birbaumer, N.; Wolpaw, J.R. BCI2000: A general-purpose brain-computer interface (BCI) system. *IEEE Trans. Biomed. Eng.* **2004**, *51*, 1034–1043. [CrossRef] [PubMed]
3. Brunner, C.; Birbaumer, N.; Blankertz, B.; Guger, C.; Kübler, A.; Mattia, D.; del R. Millán, J.; Miralles, F.; Nijholt, A.; et al. BNCI Horizon 2020: Towards a roadmap for the BCI community. *Brain-Comput. Interfaces* **2015**, *2*, 1–10. [CrossRef]
4. Penaloza, C.I.; Nishio, S. BMI control of a third arm for multitasking. *Sci. Robot.* **2018**, *3*, eaat1228. [CrossRef] [PubMed]
5. Tangermann, M.; Müller, K.R.; Aertsen, A.; Birbaumer, N.; Braun, C.; Brunner, C.; Leeb, R.; Mehring, C.; Miller, K.; Mueller-Putz, G.; et al. Review of the BCI Competition IV. *Front. Neurosci.* **2012**, *6*, 55. [CrossRef]
6. Ang, K.K.; Chin, Z.Y.; Wang, C.; Guan, C.; Zhang, H. Filter Bank Common Spatial Pattern Algorithm on BCI Competition IV Datasets 2a and 2b. *Front. Neurosci.* **2012**, *6*, 39. [CrossRef] [PubMed]
7. Gaur, P.; Pachori, R.B.; Wang, H.; Prasad, G. A multi-class EEG-based BCI classification using multivariate empirical mode decomposition based filtering and Riemannian geometry. *Expert Syst. Appl.* **2018**, *95*, 201–211. [CrossRef]
8. Wolpaw, J.R.; Birbaumer, N.; McFarland, D.J.; Pfurtscheller, G.; Vaughan, T.M. Brain–computer interfaces for communication and control. *Clin. Neurophysiol.* **2002**, *113*, 767–791. [CrossRef] [PubMed]
9. Jouppi, N.P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. *SIGARCH Comput. Archit. News* **2017**, *45*, 1–12. [CrossRef]
10. Craik, A.; He, Y.; Contreras-Vidal, J.L. Deep learning for electroencephalogram (EEG) classification tasks: A review. *IOP Sci.* **2019**, *16*, 031001. [CrossRef] [PubMed]
11. Lawhern, V.J.; Solon, A.J.; Waytowich, N.R.; Gordon, S.M.; Hung, C.P.; Lance, B.J. EEGNet: A compact convolutional neural network for EEG-based brain–computer interfaces. *J. Neural Eng.* **2018**, *15*, 056013. [CrossRef] [PubMed]
12. Kwon, Y.H.; Shin, S.B.; Kim, S.D. Electroencephalography Based Fusion Two-Dimensional (2D)-Convolution Neural Networks (CNN) Model for Emotion Recognition System. *Sensors* **2018**, *18*, 1383. [CrossRef] [PubMed]
13. Herman, P.; Prasad, G.; McGinnity, T.M.; Coyle, D. Comparative Analysis of Spectral Approaches to Feature Extraction for EEG-Based Motor Imagery Classification. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2008**, *16*, 317–326. [CrossRef] [PubMed]
14. Machida, K.; Nambu, I.; Wada, Y. Neural Network Including Alternative Pre-processing for Electroencephalogram by Transposed Convolution. In Proceedings of the Neural Information Processing, New Delhi, India, 3 July 2022; Yang, H., Pasupa, K., Leung, A.C.S., Kwok, J.T., Chan, J.H., King, I., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 139–146.
15. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 1578–1585. [CrossRef]
16. Dumoulin, V.; Visin, F. A guide to convolution arithmetic for deep learning. *arXiv* **2016**, *arXiv:1603.07285*.
17. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Bach, F., Blei, D., Eds.; PMLR: Lille, France, 2015; Volume 37, pp. 448–456.
18. Bjorck, N.; Gomes, C.P.; Selman, B.; Weinberger, K.Q. Understanding Batch Normalization. In *Advances in Neural Information Processing Systems 31*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; pp. 7694–7705.
19. Lin, M.; Chen, Q.; Yan, S. Network In Network. *arXiv* **2013**, arxiv:1312.4400.
20. Brunner, C.; Leeb, R.; Müller-Putz, G.; Schlögl, A.; Pfurtscheller, G. *BCI Competition 2008-Graz Data Set A*; Institute for Knowledge Discovery (Laboratory of Brain–Computer Interfaces), Graz University of Technology: Graz, Austria, 2008.
21. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.

22. Schlögl, A.; Kronegg, J.; Huggins, J.; Mason, S. Evaluation criteria in BCI research. In *Toward Brain-Computer Interfacing*, 1st ed.; Neural Information Processing Series; MIT Press: Cambridge, MA, USA, 2007; pp. 327–342.
23. Barachant, A.; Bonnet, S.; Congedo, M.; Jutten, C. Multiclass Brain–Computer Interface Classification by Riemannian Geometry. *IEEE Trans. Biomed. Eng.* **2012**, *59*, 920–928. [CrossRef] [PubMed]