Learning Temporal Information for Brain-Computer Interface Using Convolutional Neural Networks

Siavash Sakhavi, *Student Member, IEEE*, Cuntai Guan, *Fellow, IEEE*, and Shuicheng Yan, *Fellow, IEEE*

***Abstract*— Deep learning (DL) methods and architectures have been the state-of-the-art classification algorithms for computer vision and natural language processing problems. However, the successful application of these methods in motor imagery (MI) brain–computer interfaces (BCIs), in order to boost classification performance, is still limited. In this paper, we propose a classification framework for MI data by introducing a new temporal representation of the data and also utilizing a convolutional neural network (CNN) architecture for classification. The new representation is generated from modifying the filter-bank common spatial patterns method, and the CNN is designed and optimized accordingly for the representation. Our framework outperforms the best classification method in the literature on the BCI competition IV-2a 4-class MI data set by 7% increase in average subject accuracy. Furthermore, by studying the convolutional weights of the trained networks, we gain an insight into the temporal characteristics of EEG.**

***Index Terms*— Brain–computer interface (BCI), convolutional neural network (CNN), deep learning (DL), machine learning, motor imagery (MI), signal processing.**

I. Introduction

DEEP learning (DL), as a subcategory of machine learning, is currently the state-of-the-art method in computer vision and natural language processing applications. This phenomenon is loosely associated with the revolutionary paper by Hinton and Salakhutdinov [1], which reignited the interest in neural networks–the building block of modern deep architectures. In 2012, the convolutional neural network (CNN) won the ImageNet competition [2] with better learning algorithms, faster computational resources, and large annotated data sets. This architecture renewed the interest of many computer vision researchers in neural networks and deep architectures. The concept of hierarchical convolutional data processing (used in CNN) was initially introduced by Fukushima [3] in the neocognition framework in 1980 and further developed by LeCun *et al.* [4] in the LeNet-5 architecture in 1998. Since the 2012 success of CNN, convolutional architectures have become an important tool in vision-related applications, and have been rapidly modified and improved, including modification in network architecture design (network in network [5], inception [6], identity mapping [7]) or network activation functions (ELU [8], PReLU [9]), and improvements in optimization algorithms (ADAM [10]), or network regularization (batch normalization [11], weight normalization [12]).

Besides computer vision tasks, DL and CNNs have also been utilized in other fields such as speech recognition, text understanding, and more recently, brain–computer interfaces (BCIs), which is the main focus of this paper.

Motor imagery (MI) BCI systems base their framework on the fact that there will be a change of activation in certain areas of the brain when a patient/subject imagines moving any part of their body. For example, when a person imagines moving his/her right arm, there will be a desynchronization of neural activities in the primary motor cortex of the left brain. This desynchronization, called event-related desynchronization (ERD) in neuroscience literature, can be seen in the EEG signal as a transition from a resting state energy level to a lower energy level. The spatial location, temporal onset, amount of decrease, and stability of the ERD are all subject-dependent factors, which pose a challenge for designing a single framework to detect the changes in the neural activities that can be accurate and functional for a wide variety of users.

Inspired by MI-based ERD, computer scientists and BCI researchers have proposed some classification methods based on the common spatial patterns (CSPs) algorithm [13]. The CSP algorithm finds a set of linear transformations (i.e., spatial filters) that maximize the distance of multiple classes (i.e., right hand, left hand, and feet) of data recorded during an MI-EEG task. After estimating the spatial filters, the relative energy of the filtered channels is computed as the representation of the data. This representation of the high-dimensional EEG data can be easily fed into a linear classifier, such as support vector machine (SVM), leading to good performance [14]. However, in this representation, since the signal is reduced from a series to a single value, the temporal information is destroyed and the dynamics of the signals are neglected, which may contain valuable information regarding EEG.

The filter-bank CSP (FBCSP) algorithm [15], [16] extends the CSP algorithm by considering that not all frequency bands contain discriminative information, and therefore, by passing the signal through a filter bank, it computes the CSP energy

Manuscript received August 8, 2016; revised January 24, 2017 and June 6, 2017; accepted January 2, 2018. *(Corresponding author: Siavash Sakhavi.)*

S. Sakhavi is with the Department of Electrical and Computer Engineering, National University of Singapore (NUS), Singapore, and also with the Institute for Infocomm Research (I^2R), Agency for Science, Technology, and Research (A*STAR), Singapore (e-mail: s.sakhavi@gmail.com).

C. Guan is with the Institute for Infocomm Research (I^2R), Agency for Science, Technology, and Research (A*STAR), Singapore, and also with the Nanyang Technological University, Singapore.

S. Yan is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2789927

features for each of the temporally filtered signal outputs. The features are then selected and classified. The extra step of performing CSP on each filtered input helps boost the classifier performance and shows the benefits of signal decomposition before spatial filter estimation.

There have also been some other successful attempts to improve the CSP algorithm. To name a few, Sparse CSP [17] adds a regularization factor on the spatial filter estimation, imposing sparsity on the weights; stationary CSP [18], divergence-CSP [19], and probabilistic CSP [20], each try to solve the CSP problem either by changing the objective function or by defining a more generalized computational framework around the problem. Some algorithms simultaneously try to optimize frequency and spatial filters such as [21] and [22]. Although these methods improve the CSP algorithm and increase the classification accuracy, they still suffer from the same caveat of the original CSP method: the negation of temporal dynamics.

The utilization of DL methods in EEG-based BCI has been relatively scarce. The high dimensionality (multichannel and sampling rate) of EEG data, channel correlation and presence of artifacts (i.e., movement), and noise make it challenging to design an ideal framework for EEG classification using DL. Such a framework must first include a data preparation stage, in which the signal is reduced to lower dimensionality and transformed into a new representation without any significant loss in information. Based on this representation, the next stage of the framework (i.e., the network architecture) must be designed to extract meaningful features from the input. With the challenges in mind, DL methods have been successfully implemented for EEG classification.

Cecotti and Graser [23] introduced a CNN classifier for classification of a P300 (a positive peak seen in the EEG signal 300 ms after presenting a stimulus to the subject) speller task [23]. The CNN in this paper is used in both temporal and spatial manner: a convolution is first performed on the spatial EEG channels, thus mixing them, and in the next layer, a convolution is performed in time along the temporal samples of the EEG signal. Filtered EEG signals are used as inputs. The authors also used a similar architecture in another paper for steady-state visual evoked potentials [24] and rapid serial visual presentation task [25].

In another work, Stober *et al.* [26] used two representations (raw signal and spectral features) for classifying music imagery EEG signals using CNNs. The results verify the capacity of CNNs in classifying imagery-based EEG. In a follow-up paper, Stober *et al.* [27] used a convolutional autoencoder to pretrain a CNN on the same data set in a unique fashion using cross-trial encoding and similarity-constraint encoding. These two techniques increase the number of samples for the network to learn from and can be used as a solution in problems with a low number of data samples in the future studies.

In a more recent paper, Bashivan *et al.* [28] introduced a novel representation for EEG Signals by using an image of the topological map of the EEG signal's fast Fourier transform (FFT) on the scalp, in a specific time interval. In this way, a sequence of images is generated for the whole EEG trial and then fed into a combined CNN and long short-term memory for classification.

Hajinoroozi *et al.* [29] and [30] applied DL techniques for driver cognitive performance with deep belief networks (DBNs) and restricted Boltzmann machine on raw representations of the EEG

Literature on application of DL in MI EEG (MI-EEG) is also limited. An *et al.* [31] proposed to use manually extracted features from the channels based on FFT and then feed them into a DBN. Although it can be considered as a demonstration of DL, it only uses a DBN as a classifier and does not interpret the network as a feature learning algorithm.

In the paper by Yang *et al.* [32], building upon the success of FBCSP, an augmented-CSP algorithm was proposed by using overlapping frequency bands. The log-energy features are extracted for each frequency band and arranged on a 2-D matrix. By training a convolutional network on the frequency-energy matrix, the network learns to discriminate the features. Furthermore, a map selection algorithm is used to select specific feature maps after the convolution operation. The interpretation of the weights in the network is unknown and the features selected neglect the time dynamics.

Sakhavi *et al.* [33] proposed a parallel MLP and CNN architecture from which the predictions of the networks are joined via averaging. The MLP architecture receives the log-energy features of the FBCSP algorithm and the CNN receives a temporal representation of the selected EEG channels and frequencies from the FBCSP algorithm. The temporal representation is the channel-relative instantaneous energy of the envelope of the EEG signal, extracted using the Hilbert transform. In other words, instead of compressing the signal into a single value, a new temporal representation of the signal is utilized in which the temporal dynamics are intact. The CNN network is then applied in a channel-wise fashion on each class independently and combined before the fully connected layer, yielding a significant, but small, increase in classification accuracy.

In this paper, we propose a new architecture built upon the idea of our previous paper [33]. By using the FBCSP algorithm as our data preparation method, we propose a novel envelope representation for the MI-EEG. This representation of the data is important in several ways. First, it is a temporal representation of the EEG, preserving the information relative to the dynamics of the EEG signal throughout recording, and thus is more valuable than a single value energy representation. Second, in terms of information content, it lowers the dimension without distorting the signal, as shown in Section III-B. Third, by combining this envelope representation with a CNN, the kernels learned in the network give an insight into the morphology and pattern in the input, which gives rise to class discrimination. This representation and the proposed architecture have achieved state-of-the-art performance on the BCI competition IV-2a data set, significantly increasing the average accuracy.

II. Data

In this paper, we focus on the 2008 BCI competition IV-2a EEG data set [34], which is a four-class MI (left, right, feet, and tongue) data set recorded from 22 Ag/AgCl electrodes with a 250-Hz sampling rate in two sessions from nine subjects. Each session has 72 trials per class resulting in 288 samples per session. The timing scheme consists of a fixation of 2 s, cue time of 1.25 s, followed by a period of a MI of 4 s. Previous attempts on classification of the data show the data set consists of various types of subjects, measured based on their performance in terms of accuracy score and Cohen's kappa.

III. Methods

The main process of feature extraction is based on FBCSP. In Section III-A, we will review the algorithm and then in Section III-B discuss on extracting temporal features from EEG based on FBCSP. Note that our temporal feature extraction method *modifies and borrows* the results from the original FBCSP algorithm.

A. Filter-Bank Common Spatial Patterns

FBCSP was first introduced as an extension of the original CSP algorithm and gained attention by winning the 2008 BCI Competition IV-2a [15], [16]. CSP is viewed as a data-driven spatial filtering algorithm. Spatial filtering algorithms aim to decorrelate EEG channels individually or find a combination of the channels that have valuable information regarding the task at hand. CSP and FBCSP try to find a linear combination of channels (i.e., spatial filter) that discriminates two classes of data. The procedure for FBCSP is as follows.

1) The EEG signals from all recorded channels are filtered using a filter bank with nine subsequent bandpass filters, starting at 4 Hz and with a bandwidth of 4 Hz (4–8 Hz, 8–12 Hz, ...). All filters are type II Chebyshev filters.
2) Spatial filters for each output of the filter bank are computed using CSP. This is done by maximizing the following objective function:
$$w^* = \arg\max_{w} \frac{w^T \Sigma_{c_1} w}{w^T(\Sigma_{c_1} + \Sigma_{c_2})w} \tag{1}$$
where Σ_{c_1} and Σ_{c_2} correspond to the channel covariance matrix for classes c_1 and c_2, respectively, in a specified time segment and w is the spatial filter. This objective function, also called the Rayleigh Quotient, has an analytical solution, which is equivalent to solving a generalized eigenvalue decomposition problem.
3) Spatial filters corresponding to the $2 \times N_W$ extreme eigenvalues (N_W largest and N_W smallest eigenvalues) are selected. Each of the extreme spatial filters is then paired with each other correspondingly (spatially filtered channel pairs).
4) Energy (variance) of the spatially filtered channels is calculated (E_C) and normalized to the total energy of the channels in a given frequency band ($\tilde{E}_C = (E_C/(\Sigma_{i=1}^{2\times N_W} E_C))$). The logarithm of energy is computed as the final feature.
5) Features coming from all nine filter bands are concatenated and a mutual information-based feature selection is performed on $2 \times N_W \times 9$ spatially filtered channels, where N_S filtered channels and their pairs are chosen. Depending on whether the selected features are already pairs with each other or not, a maximum of $2 \times N_S$ features may be selected.
6) Because CSP is designed for a two-class problem, in the case of multiclass tasks, a one-versus-rest or one-versus-one strategy must be appointed. In FBCSP, the former is chosen and it will lead to a maximum of classNumber $\times$ $2 \times N_S$ features.

The values $N_W = 2$ and $N_S = 4$ are selected using cross validation. With the competition data containing four classes, the maximum number of features used for classification will be 32. It should be noted that the features can be handled in two ways: concatenation of all features into one large vector, or using features extracted by class-specific spatial features individually. In [16], the latter has been used. Because of the success of FBCSP in classification frameworks, we decide to build a feature extraction procedure for temporal features utilizing a modified version of the FBCSP algorithm, which will be described in Section III-B.

B. Extracting Temporal Features

After performing the FBCSP process as described in III-A, we extract the temporal features as the following procedure.

1) After FBCSP, we have indices of the selected filtered channels for each frequency band and each class. We use these indices to extract the corresponding EEG signals. Note that, in contrast with FBCSP, which has a variable feature dimension output, we force the selection algorithm to select $2 \times N_S$ spatially filtered channel pairs. This ensures that the dimension of the input, and therefore, the structure of the designed network are consistent between subjects. Furthermore, we use the whole period of MI (0 to 4 s) as the time segment to estimate the covariance matrices for the CSP algorithm.
2) The signal envelope of each signal is extracted using the Hilbert transform [35], which gives the analytic form of a signal that is complex-valued and interpreted as the one-sided version of the original signal's frequency spectrum. Taking the amplitude of the analytic form gives an estimate of the envelope.
3) After extracting the envelope, we consider three possible representations for the EEG (details regarding how to choose the representations will be given in Section IV-B):
   a) using the raw (or a smoothed version of) EEG envelope (R1);
   b) taking the power of the envelope, which can be interpreted as instantaneous energy (R2);
   c) dividing the envelope power by the total energy of each of the channels in each trial, similar to Step 4 in the FBCSP algorithm described in III-A (R3).
4) Since the spectral nature of the envelope is low frequency, we can down-sample the signal without information loss. This comes in handy to lower the dimension of the data, especially when the number of samples is limited. This is a natural benefit of using the FBCSP algorithm: the filter bank intrinsically reduces

the number of samples needed to represent the original data.

5) After extracting the envelope signals for each class, instead of taking a one-vs-rest strategy for classification, we concatenate all four classes forming a single matrix of signals.

With the same values of $N_W = 2$, $N_S = 4$ and with four classes, the number of channels will be 32. As for the dimension of the features in time, the original sampling frequency of the data is 250 Hz and by choosing 4 s interval of data, we will have 1000 samples. Note that the envelope will have a cutoff frequency of 4 Hz, which means a sampling frequency of 8 Hz is sufficient for the signal (Nyquist rate), but we choose to reduce the frequency to 10 Hz, yielding 40 time points for the 4 s interval. Overall, the dimension of the data fed into the network will be 32×40. From here on we will refer to the channel dimension as feature channels.

C. Designing the CNN Architecture

When the CNN LeNet5 was first introduced by LeCun *et al.* [4], it consisted of a sequence of convolutions and subsampling layers mainly containing max pooling. Since then, most architectures follow such a procedure with some additions such as dropout [36], batch normalization [11], inception [6], and/or identity mapping [7]. These additions, based on their natures, can lead to faster training of the network, better conservation of information throughout the hierarchical process, and/or avoid overfitting of the network.

For designing the network, the nature of the input (in our case the envelope representation), should be taken into consideration. Each of the 32 feature channels may be from different frequency bands (based on the feature selection algorithm). In terms of spatial filters, each feature channel has a unique spatial filter (based on the selected eigenvalues in the CSP algorithm), which is designed for discriminating one class against the other classes. Furthermore, it is possible for the spatial filters to be correlated since a spatial pattern can be discriminative for two classes or more. Therefore, we will consider different scenarios and their interpretations of utilizing convolution for our EEG representation.

Scenario 1 :Convolution Only Across Time With a Common Kernel Shape for All Feature Channels: In this type of convolution, the assumption is that feature channels selected for classification, although intrinsically different and independent, share a common morphology. This morphology can be captured from each channel using a common kernel, which learns the morphology leading to the discrimination of classes. The choice of such a convolutional kernel will result in preserving the channels during the convolutional layers of a network (channel-wise convolution) and reduction of the temporal dimension. After convolution, the fully-connected layers will mix all the channels and temporal values and then classify them. We will call the network architecture utilizing this convolution “Channel-wise CNN” (CW-CNN).

Scenario 2: Convolution Only Across Channels: This operation can be interpreted as mixing the channel signals with each other. For example, if the size of the convolutional kernel for this layer is the same as the number of channels, the output of the convolution operation is a new signal, which is the linear combination of all the given channels. A kernel size smaller than the number of channels is not ideal because it implies that a common linear combination can be shared amongst the channels, but usually the EEG channels in this stage are independent and their order in the input matrix is not important. We call this scenario “Channel mixing CNN” (CM-CNN). This type of convolutional layer is better used with CW-CNN because the FBCSP input is already a linear mixture of the original EEG channels and an extra channel mixing is redundant. Only after some processing can a new mixture of the channels make sense.

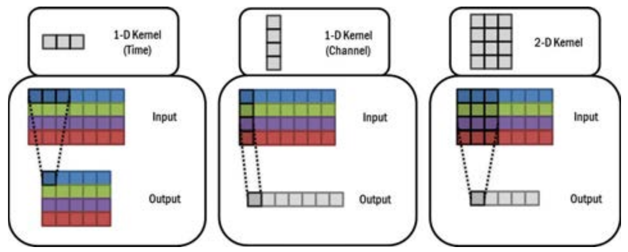

Fig. 1. Three types of convolution possible to be implemented on any feature map (from left to right): convolution in time, convolution in channel, and 2-D convolution. A mapping from multiple squares to one square is a linear combination of the multiple squares into a single value. The number of squares in the output of each convolution corresponds with the actual effect of convolution. Color shows an independent time series. The gray values in the output means the channels values are mixed.

Scenario 3: Convolution Across Both Time and Channels Using a Two-Dimensional Kernel: The evident result of this type of convolution, in addition to convolution in time, is the mixing of the feature channels after they are convoluted. This scenario produces a new time series, which captures information from all the channels simultaneously. The receptive field of the kernel in the channel dimension determines which channels should be mixed with each other. For example, if all 32 channels of the input are mixed, the output will be a single channel feature, which is the summation of the convolution of all the other feature channels with their own unique filter. Or, in another case, if the feature channels related to the class-specific spatial filters are mixed, the output is a channel summing the information of a class. We will call this type of architecture 2-D convolution scenario (2D-CNN), because of its similarity to most 2-D CNN architectures.

Another way of performing 2-D convolution is by breaking up the 2-D convolution into two 1-D convolutions in two separate layers. This method of implementing the 2-D convolution makes the convolution in time and space independent of each other and increases the flexibility of the network but with the cost of increasing the number of parameters due to the introduction of a new computational layer. It can be viewed as adding a channel mixing to the CW-CNN network. Therefore, this architecture will be called “Channel-wise Convolution with Channel Mixing” (C2CM).

Fig. 1 illustrates the convolution in time, convolution in channel, and 2-D convolution in detail.

Amongst all the models, 2D-CNN has the smallest number of parameters, followed by C2CM. The main contributor to the

TABLE I
CROSS-VALIDATION PARAMETERS

Cross Validation Parameter	Initial value for other parameters	Cross Validation Values
Convolutional kernel sizes (convParams)	MLP Nodes = none, Conv. Nodes = $\{32, 32\}$	$\{\{4,1,3,1\},\{3,1,2,1\}\}, \{\{7,1,3,1\},\{3,1,3,1\}\}, \{\{8,1,2,1\},\{5,1,3,1\}\}, \{\{10,1,3,1\},\{3,1,2,1\}\}, \{\{10,1,2,1\},\{4,1,2,1\}\}, \{\{16,1,3,1\},\{3,1,2,1\}\}, \{\{20,1,2,1\},\{5,1,2,1\}\}$
Number of convolutional nodes (hodNodes)	Chosen values for convParams, MLP Nodes = none	$\{8,8\}, \{16,16\}, \{32,32\}, \{64,64\}, \{128,128\}, \{256,256\}$

number of parameters trained in a CNN is the connection of the last convolutional layer to the fully connected layers. In all CNN architectures, the last convolutional feature maps are vectorized and stacked into one large vector and fed into fully connected layers. The smaller the dimension of the feature map of the last layer is, the lower the number of linear units used will be. In the case of both 2D-CNN and C2CM, because the feature map size is reduced due to convolution in both dimensions, the number of parameters is significantly lower compared to CW-CNN. A lower number of parameters is desired especially when the number of training samples is relatively low to avoid overfitting and allow for better training of the network.

We will present the results for architectures C2CM and CW-CNN in Section IV.

D. Parameter Selection via Cross Validation

For each of the scenarios in Section III-C, we need to choose the network parameters; number of layers (convolutional, fully connected), kernel size, number of hidden nodes, convolution stride, pooling method, regularization methods (batch normalization, dropout) and other network related parameters are considered as hyperparameters and can be optimized. These hyperparameters, in addition to different representations for the EEG brought in Section III-B, must be correctly chosen based on cross-validation. Practically, it is not feasible to search through the parameter space due to time and computation limitations. Instead, we use coordinate descent as a suboptimal method to perform cross validation for the network parameters [37]. In coordinate descent, a set of parameters, $\Theta = [\theta_1, \theta_2, \ldots, \theta_N]$, is initialized and then the objective function or score function is optimized for each θ_i $(i = 1, \ldots, N)$ independently, while updating the values of the initial Θ with the newly optimized parameters. After N optimizations, the Θ vector will be completely updated and a new iteration of optimization can be initiated. For better results, the algorithm can be repeated for several iterations.

For this paper, two values are selected via cross-validation: size of the kernel and number of convolutional nodes. Table I shows the values chosen for each of the two values plus the other parameters' initialization values. Multiple values in curly brackets show the number of layers used during cross validation. For convolution kernels, the values in the bracket are as follows: kernel width, kernel height, stride in width, and stride in height. For example, $\{\{4, 1, 3, 1\}, \{3, 1, 2, 1\}\}$ indicates that the convolution has two layers where the first layer has a kernel size of 1×4 with a stride of 3 and the second layer has a kernel size of 1×3 with a stride of 2. We perform a ten-fold cross validation only once in order to select the parameters. The convolutional layer parameters (*convParams*) are first selected using cross-validation and then the selected values are used for cross validation for selecting the number of convolutional nodes (*hidNodes*). For C2CM structure, which has an additional computational layer, every value is the same as the CW-CNN structure. The channel mixing layer is positioned after the channel-wise convolutions and has the same number of hidden nodes as the previous convolutional layers.

TABLE II
SAMPLE ARCHITECTURE

Layer Type	Patch size / Stride	Input Size	Hidden Unit	Parameters
Convolution	$1 \times 7/1 \times 3$	32×40	32	256
ReLU	-	32×12	-	0
Convolution	$1 \times 3/1 \times 3$	32×12	32	3104
ReLU	-	32×4	-	0
Linear	-	4096	512	2,097,664
ReLU	-	512	-	0
Linear	-	512	4	2052
LogSoftMax	-	4	4	0

The kernel size and stride combination for the first and second layer are chosen based on the input size and in a way that the output of the CNN is an integer value. It should be noted that in the conventional CNN architecture used in computer vision, the kernel size is high for the first layer, but is reduced in the subsequent layers. In order to find the rule of the kernel, we give different values for the first layer from a size 4 corresponding to an interval of 400 ms to a size of 20, which corresponds to an interval of 2 s. Furthermore, compared with conventional architectures, we choose not to use any subsampling method and instead, rely solely on changing the stride.

The training of the networks is performed with the following configurations.

1) ADAM [10] is used as the optimization method. The parameters are set to default values as [10].
2) Negative log-likelihood is taken as the optimization criterion.
3) In all layers, we insert a batch normalization [11] layer before the activation layer and a dropout layer after the activation with a probability of 50%.

As an example, considering the input size calculated in Section III-B, Table II shows the input size at each layer for a sample two-layer CW-CNN architecture, while Fig. 2 shows a sample visualization of both CW-CNN and C2CM networks.

IV. RESULTS

A. Baseline Method

In our paper, we use both Cohen's kappa and accuracy to evaluate our method. The FBCSP feature extraction algorithm in combination with a linear C-SVM classifier is used as

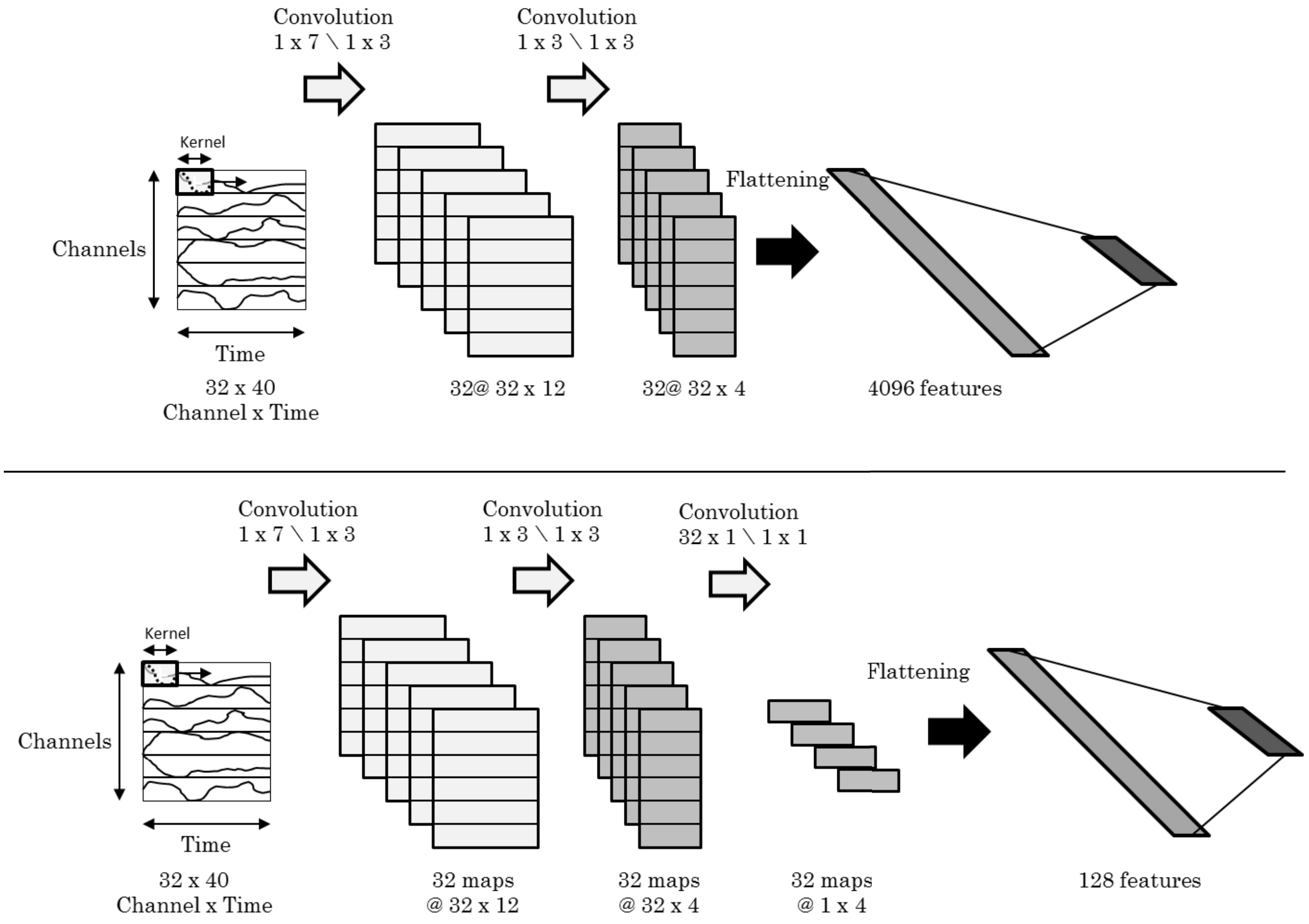

Fig. 2. Visualization of sample architecture CW-CNN (top) and C2CM (bottom). After the convolution layer, the feature maps are flattened into a single vector and fed to the fully connected network. It can be seen that the more the convolutional layers, the lower the dimension of the last convolution operation, and therefore, less parameters needed to connect to the fully connected layers.

the baseline. Features are extracted from the 0.5 to 2.5 s after the cue for both train and test data sets. Kappa values are from the original FBCSP paper. We also include the results from a paper of Bashashati *et al.* [38], which used Bayesian optimization to find the best parameters for FBCSP and achieved the best results to our knowledge on the BCI competition IV-2a data set in terms of accuracy. In terms of kappa, we use the values in [39], which is shown to be the highest amongst many methods (accuracy was not reported in this paper). The baseline results can be seen in Table VII.

In order to verify the significance for increase/decrease of the accuracy, a one-sided Wilcoxin signed-rank test [40] is used. This test is appropriate in conditions where the number of paired samples to compare is relatively small and non-Gaussian.

B. EEG Representation and Architecture Comparison

In order to select one of the three EEG representations described in Section III-B, we use a simple architecture with a set of chosen parameters and perform a ten-fold cross validation over each of the representations. The architecture is similar to that in Table II, with two convolutional layers (32 nodes each) and without the fully connected layer. The average cross-validation values are obtained by repeating the measurements for 10 networks (10 networks $\times$ 10 folds). Our assumption is that with a common architecture, the better representation would have a higher cross-validation average. The results can be seen in Table III. Table III shows that the CNN architecture selects the $R1$ representation for all subjects and the cross-validation average accuracy is higher than the other two representations.

For classifier comparison, we perform cross validation of the given representation on two additional classifiers: linear SVM and MLP (two hidden layers with 32 hidden nodes, 10 networks). The cross-validation results are given in Table IV. It can be seen that on average, most subjects have a higher cross-validation accuracy for the CNN architecture, thus CNN is selected.

C. Architecture Parameter Selection

With the chosen representation, we conduct cross validation over the network parameters based on the values in Table I. Parameters are selected by averaging over the results of each fold and multiple network initializations. Here, a ten-fold

TABLE III
CROSS-VALIDATION ACCURACY RESULTS FOR FEATURE REPRESENTATION GIVEN THE C2CNN ARCHITECTURE IN TABLE II

Feature	Classifier	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Average
Envelope (R1)	CNN	85.23	69.73	90.15	65.57	77.42	52.41	93.68	90.04	84.75	**78.78**
Instantaneous Energy (R2)		83.38	69.42	87.38	65.27	75.43	47.71	92.38	87.45	82.40	76.76
Relative Instantaneous Energy (R3)		85.04	65.67	87.07	64.28	73.72	48.59	91.97	87.80	83.55	76.41

TABLE IV
CROSS VALIDATION ACCURACY RESULTS FOR CLASSIFIERS GIVEN THE $R1$ FEATURE

Feature	Classifier	Parameters	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Average
Envelope (R1)	CNN	19748	85.23	69.73	90.15	65.57	77.42	52.41	93.68	90.04	84.75	**78.78**
	MLP	42180	85.20	68.16	91.04	60.86	73.68	52.34	90.20	88.69	85.83	77.33
	SVM	5120	82.36	69.52	90.84	62.09	74.32	52.82	91.91	86.57	83.89	77.15

TABLE V
ACCURACIES USING THE SELECTED PARAMETERS FROM CROSS VALIDATION FOR CW-CNN

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Average
	Test Results (50 Ensemble)									
	86.11	60.76	86.81	67.36	62.50	45.14	90.63	81.25	77.08	73.07
Selected Kernels Size	{{20,1,2,1}, {5,1,2,1}}	{{7,1,3,1}, {3,1,3,1}}	{{20,1,2,1}, {5,1,2,1}}	{{20,1,2,1}, {5,1,2,1}}	{{10,1,2,1}, {4,1,2,1}}	{{20,1,2,1}, {5,1,2,1}}	{{20,1,2,1}, {5,1,2,1}}	{{7,1,3,1}, {3,1,3,1}}	{{20,1,2,1}, {5,1,2,1}}	
Selected Hidden Nodes	{64,64}	{32,32}	{32,32}	{32,32}	{32,32}	{8,8}	{8,8}	{32,32}	{32,32}	

TABLE VI
ACCURACIES USING THE SELECTED PARAMETERS FROM CROSS-VALIDATION FOR C2CM

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Average
	Test Results (50 Ensemble)									
	87.50	65.28	90.28	66.67	62.50	45.49	89.58	83.33	79.51	74.46
Selected Kernels Size	{{10,1,2,1}, {4,1,2,1}, {1,32,1,1}}	{{4,1,3,1}, {3,1,2,1}, {1,32,1,1}}	{{10,1,2,1}, {4,1,2,1}, {1,32,1,1}}	{{20,1,2,1}, {5,1,2,1}, {1,32,1,1}}	{{20,1,2,1}, {5,1,2,1}, {1,32,1,1}}	{{20,1,2,1}, {5,1,2,1}, {1,32,1,1}}	{{4,1,3,1}, {3,1,2,1}, {1,32,1,1}}	{{4,1,3,1}, {3,1,2,1}, {1,32,1,1}}	{{10,1,3,1}, {3,1,2,1}, {1,32,1,1}}	
Selected Hidden Nodes	{256,256}	{32,32}	{256,256}	{256,256}	{128,128}	{32,32}	{256,256}	{8,8}	{256,256}	

TABLE VII
TABLE OF ACCURACY AND KAPPA FOR BASELINE METHODS AND OUR METHOD. VALUES IN THE PARENTHESIS ARE KAPPA VALUES AND THE NONPARENTHESIS ARE ACCURACY

	FBCSP [16]	FBCSP [38]	BO [38]	TSSM+SVM [39]	SVM	CW-CNN	C2CM
Subject 1	(0.676)	76.00	82.12	(0.77)	82.29 (0.764)	86.11 (0.815)	**87.50** (**0.833**)
Subject 2	(0.417)	56.5	44.86	(0.33)	60.42 (0.472)	60.76 (0.477)	**65.28** (**0.537**)
Subject 3	(0.745)	81.25	86.60	(0.77)	82.99 (0.773)	86.81 (0.824)	**90.28** (**0.870**)
Subject 4	(0.481)	61	66.28	(0.51)	**72.57** (**0.634**)	67.36 (0.565)	66.67 (0.556)
Subject 5	(0.398)	55	48.72	(0.35)	60.07 (0.468)	**62.50** (**0.500**)	**62.50** (**0.500**)
Subject 6	(0.273)	45.25	**53.30**	(**0.36**)	44.10 (0.255)	45.14 (0.269)	45.49 (0.273)
Subject 7	(0.773)	82.75	72.64	(0.71)	86.11 (0.815)	**90.63** (**0.875**)	89.58 (0.861)
Subject 8	(0.755)	81.25	82.33	(0.72)	77.08 (0.694)	81.25 (0.750)	**83.33** (**0.778**)
Subject 9	(0.606)	70.75	76.35	(**0.83**)	75.00 (0.667)	77.08 (0.694)	**79.51** (0.727)
Average	(0.569)	67.75	68.13	(0.593)	71.18 (0.616)	73.07 (0.641)	**74.46** (**0.659**)

cross validation is performed on 10 network initializations. The average accuracy of these 100 networks is used for classification. The selected parameters for architectures CW-CNN and C2CM can be seen in Tables V and VI, respectively. The final test accuracy reported in both tables is obtained by averaging and ensemble of 50 model initializations trained on the training data using the selected parameters.

Tables VI and VII show interesting results. For the CW-CNN architecture, most of the subjects select a larger kernel size with a smaller number of hidden nodes, whereas for the C2CM architecture, most subjects select lower kernel sizes but with higher hidden nodes, with the channel mixing layer being the only difference between the two architectures. This means that the channel mixing makes the network wider, and thereby increases the number of features in the output of the convolution. In contrast, without the channel mixing, the network gives more emphasis on the receptive field of the network rather than widening the network.

Table VII shows the results for multiple classification methods including our proposed method using C2CM. The values in parenthesis are Cohen's kappa. As shown in Table VII, our method has superior performance in both kappa and accuracy. The first "FBCSP" uses the SVM classifier on the FBCSP features and is reported from [16]. The second "FBCSP" is based on the FBCSP algorithm results in [38] and "BO" is the Bayesian optimization method proposed in the same paper.

"SVM" means the use of SVM on the $R1$ envelope representation features proposed in III-B. Based on the Wilcoxon signed-rank test, the increase in the mean accuracy for C2CM is significantly higher with a bound of $p < 0.05$ relative to BO, SVM, and CW-CNN. In terms of average kappa, our method is not significantly higher than the method in [39] based on the Wilcoxon test due to the ranking of the differences between the kappa values of subjects 4 and 6. But overall, there is an increase in kappa for the other subjects. For other methods, the increase in mean kappa is significant with a bound of $p < 0.05$.

V. Analysis

To further interpret the learned network weights, and verify whether the result obtained by the trained network is by random or the network actually learns important patterns in the data, we conduct some experiments.

A. Importance of the Kernel Morphology

As shown in the results of Section IV-C, each subject chooses a specific kernel length and a certain number of hidden nodes for the first two layers. Note that the selected parameter values for each subject are based on cross validation and ensembles. Therefore, the kernel sizes are dependent on each fold of the data used in cross validation and the initializations of the network for each model in the ensemble. This, in turn, makes it difficult to determine whether or not the kernel sizes have meaning from a neuroscientific perspective. Nevertheless, in this section, we seek to verify whether the convolutional kernels learned in the first two layers are truly important, or the shapes are random and do not contribute to the classification performance.

We modify the convolutional kernels such that each convolutional kernel is replaced by its mean value. We adopt the kernel mean mainly because we do not want to abrupt the scale of the network by changing the values of the kernel to nonrelated values. After changing the values, we perform statistical analysis on the accuracy with the following hypothesis: *the mean accuracy obtained by the modified network is lower than the original accuracy*. It is expected that the classification results will drop, but our hypothesis emphasizes whether this drop is significant or not.

While performing this analysis, we first modify the kernel values of each of the layers independently and eventually change the values for both layers simultaneously. Table VIII shows the accuracy values for each individual subject after the network modification from an ensemble of 50 networks, the same 50 networks used to derive the test accuracy. The W-score calculated between the original network and the change of the first layer parameters is 6, and based on the hypothesis, the maximum value at which the hypothesis is significant with a p-value of 0.05 is 8, showing that the network with the first layer modified performs worse than the original network. Modification of both layers simultaneously shows an even larger decrease relative to the original network, demonstrating that the morphology captured by the kernels of the first two layers is important.

TABLE VIII

Modification of Kernels. L1 and L2 Refer to the Kernels of Layer 1 and 2. Each Column Shows the Test Accuracy When the Kernel of Those Layers(s) Is Modified to Be the Mean Value of the Kernel

Subject	Trained CNN	Mod. (L1)	Mod. (L2)	Mod. (L1+L2)
Subject 1	87.50	82.99	82.29	82.64
Subject 2	65.28	60.42	61.11	57.99
Subject 3	90.28	84.38	85.42	83.33
Subject 4	66.67	58.33	61.81	58.33
Subject 5	62.50	54.17	58.33	53.47
Subject 6	45.49	48.26	43.40	48.26
Subject 7	89.58	90.63	90.28	92.01
Subject 8	83.33	84.03	80.21	82.29
Subject 9	79.51	61.11	60.76	59.38
Average	74.46	69.37	69.29	68.63

B. Qualitative Analysis of Kernel Shapes

After training the network, we also want to gain an understanding about what the network has learned and if possible, obtain a visualization of the learned kernels. In image processing, methods such as deconvolution [41], back-propagation-based visualization [42], [43], and layer-wise relevance propagation (LRP) [44] have been used to interpret trained networks with LRP recently being used for EEG analysis as well [45].

Here, we choose the back-propagation methods proposed in [42]. In this group of methods, an initial image is fed to the network and based on the desired activation of any specific node in the network, the initial image is recursively changed in order to match the activation at that node. This recursive algorithm can be accompanied by a smoothing function to remove high frequency noise during the recursion.

In our case, we initialize the input with the average of all training examples and then, recursively change the input in a way that its classification label vector (i.e., network output layer) corresponds to class C. The result of the recursion is an input that when fed to the network, results in class C. In other words, this perceived input is what the network recognizes as class C.

Fig. 3 shows a sample of the average signals for class 1 of subject 9 (solid blue line) and the perceived signals for class 1 for a number of channels that have high significant correlation. (The correlation r and p-values p have been provided.) This reconstruction is made based on the network that has the highest classification accuracy on the test data. By visually inspecting the graphs, in most cases, the perceived input signal follows the average signal. It should be noted that the algorithm converges and the perceived input correlates with each of the classes.

Fig. 4 shows the high positive correlation coefficient values ($r > 0.5$) with a p-value of lower than 0.05 for all feature channels of subject 9 in each of the four classes. The dashed vertical lines correspond to each of the one-versus-rest classes in the CSP algorithm. Fig. 4 illustrates high significant correlation only in certain channels for each individual class, implying that the perceived signals and average signals are only similar in these channels for these classes. Fig. 5 shows the signals sorted based on the p-value for the original average channel (left) and perceived input (right) based on the p-value

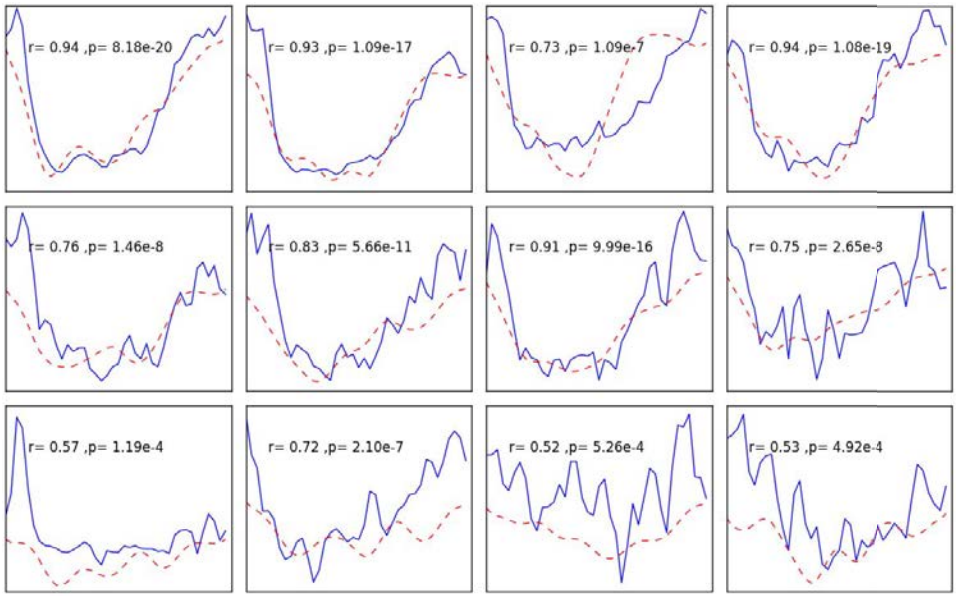

Fig. 3. Sample signal visualization for network perceived input for significantly correlated channels from Subject 9 and class 1. The blue (solid line) signal is the average signal and the red (dashed line) signal is the perceived signal from the network. Only signals with significant high positive correlation ($p < 0.05$, $r > 0.5$) have been shown. The correlation and p-value have been written in the image. The x-axis represents time and y-axis represents amplitude. The y-axis has been adjusted for each subplot for better visualization.

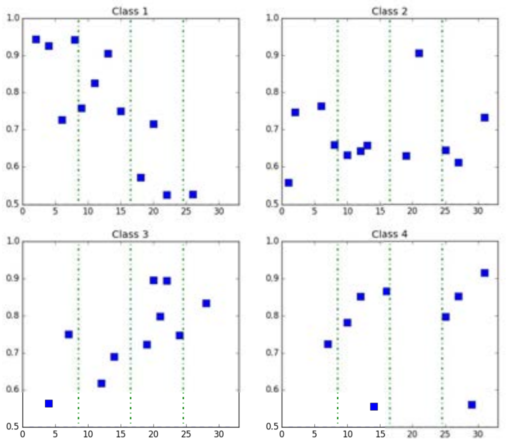

Fig. 4. Correlation between perceived signal and average signal in class 1 for subject 9. The vertical dashed lines show the channels belonging to the one-versus-rest CSP channels for each class. Only high positive correlations ($r > 0.5$) and significant correlations ($p < 0.05$) have been shown.

arranged vertically from class 1 to class 4, validating the similarity between the original data and the perceived input in some, but not all, channels. Further analysis must be conducted in order to verify whether the similarity between the perceived and average inputs in specific feature channels imply the importance of the channels or not.

VI. Discussion and Future Work

Deep neural networks and DL algorithms have opened a door to many applications, which were not possible before due to their compositional structure [46], expressivity [47], and their ability to be parallelized for speeding up the learning process. The results in this paper have identified two important facts regarding the application of DL in EEG. First, the representation of the signal fed into the DL framework is important: previous methods based solely on energy values neglect valuable temporal information. When considering the

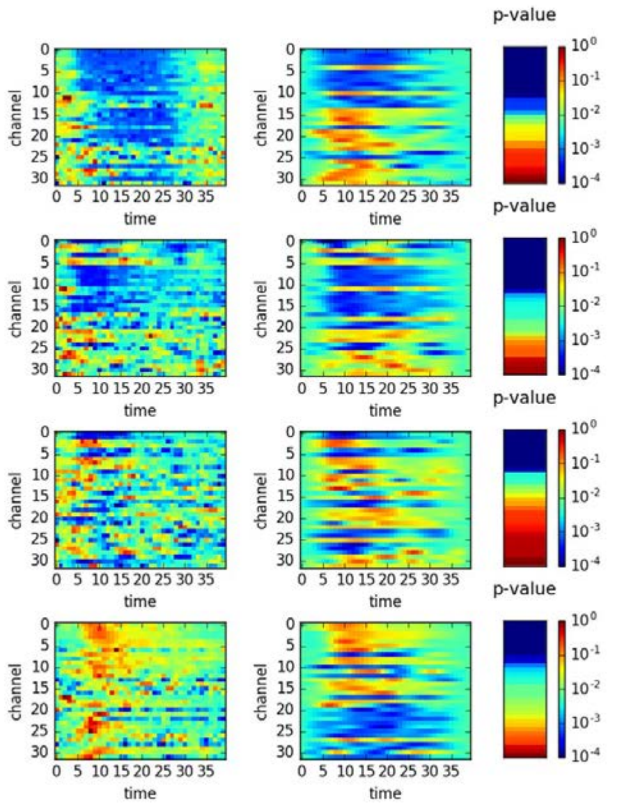

Fig. 5. Sample visualization for average class signals, and network perceived input for all channels sorted based on p-value for subject 9. Average input for each class (left column). Perceived input for each class (middle column). Correlation p-value between average and perceived classes (right column).

new representation, the way the information is processed must be updated as well because new representations need new processing methods. This demand for new processing and classification tools leads to the second important fact: DL methods can be used in the context of EEG signal classification and can yield superior results relative to other methods such as SVM and MLP. Our analysis shows that our results are not from random matrix multiplications, and the network is indeed learning something from the input EEG data, proving that the network's architecture is meaningful for EEG. Visualizing the architecture further verifies that the network has learned important relationships from the data and is able to construct a perceived input, which is similar to the original data.

One of the caveats of the current framework is the independence of the data representation algorithm from the classification network. In particular, the FBCSP method is not affected by the network optimization and in turn, the network is forced to work with an input it has no control over. We believe the combination of the two stages will yield a unified, end-to-end classification framework.

Furthermore, the training of the networks is time consuming. In BCI systems, it is desired to reduce the calibration time, i.e., the time necessary to record sufficient data and train a model. In the models that we have trained for this paper, training time can vary from 30 to 150 s per network for the C2CM model and 9 to 12 s per network for the CW-CNN. These values are multiplied when we consider training an ensemble and/or use cross validation for parameter selection. Note that using CW-CNN is much faster compared with C2CM because it has one less layer but a much larger number of parameters due to the fully connected layers. This problem of time consumption can potentially be solved by learning a generalized network from a large pool of subjects and transferring the knowledge to a new subject (transfer learning). By defining the correct representation for all subjects and choosing the parameters of the network carefully, a deep neural network will be able to learn from all subjects and use the learned information for new subjects without the requirement of being retrained (zero-shot learning) or by training on a small sample size from the new data (domain adaptation).

Also, we have not addressed the problem of how a small number of samples per class can be used to train high capacity networks without suffering from the overfitting issue. Although regularization methods such as dropout and batch normalization can help avoid overfitting, the small number of samples has pushed us to use ensemble methods. Pretraining the network via generative strategies (e.g., autoencoders) can lead to better kernel learning and lower ensemble sizes.

VII. Conclusion

Overall, the application of DL methods in EEG processing is promising and the method proposed above can be generalized to medical applications and/or different types of EEG recordings in the future.

Acknowledgment

The authors would like to thank all members of the BCI laboratory at A*STAR, Singapore, and the Learning and Vision (LV) laboratory at NUS, Singapore, for their discussions and inspiration. Furthermore, the maintenance and development of Torch on Github made the implementations of the algorithms much easier and faster.

References

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2012, pp. 1097–1105.

[3] K. Fukushima, "Neocognitron: A self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[5] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400

[6] C. Szegedy *et al.* (2014). "Going deeper with convolutions." [Online]. Available: https://arxiv.org/abs/1409.4842

[7] K. He, X. Zhang, S. Ren, and J. Sun. (Mar. 2016). "Identity mappings in deep residual networks." [Online]. Available: http://arxiv.org/abs/1603.05027

[8] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. (Nov. 2015). "Fast and accurate deep network learning by exponential linear units (ELUs)." [Online]. Available: http://arxiv.org/abs/1511.07289

[9] K. He, X. Zhang, S. Ren, and J. Sun. (Feb. 2015). "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification." [Online]. Available: http://arxiv.org/abs/1502.01852

[10] D. P. Kingma and J. Ba. (Dec. 2014). "Adam: A method for stochastic optimization." [Online]. Available: http://arxiv.org/abs/1412.6980

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[12] T. Salimans and D. P. Kingma. (Feb. 2016). "Weight normalization: A simple reparameterization to accelerate training of deep neural networks." [Online]. Available: http://arxiv.org/abs/1602.07868

[13] H. Ramoser, J. Müller-Gerking, and G. Pfurtscheller, "Optimal spatial filtering of single trial EEG during imagined hand movement," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 8, no. 4, pp. 441–446, Dec. 2000.

[14] F. Lotte and C. Guan, "Regularizing common spatial patterns to improve BCI designs: Unified theory and new algorithms," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 2, pp. 355–362, Feb. 2011.

[15] K. K. Ang, Z. Y. Chin, H. Zhang, and C. Guan, "Filter bank common spatial pattern (FBCSP) in brain-computer interface," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jun. 2008, pp. 2390–2397.

[16] K. K. Ang, Z. Y. Chin, C. Wang, C. Guan, and H. Zhang, "Filter bank common spatial pattern algorithm on BCI competition IV datasets 2a and 2b," *Front. Neurosci.*, vol. 6, p. 39, Mar. 2012.

[17] M. Arvaneh, C. Guan, K. K. Ang, and C. Quek, "Optimizing the channel selection and classification accuracy in EEG-based BCI," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 6, pp. 1865–1873, Jun. 2011.

[18] W. Samek, C. Vidaurre, K.-R. Müller, and M. Kawanabe, "Stationary common spatial patterns for brain-computer interfacing," *J. Neural Eng.*, vol. 9, no. 2, p. 26013, 2012.

[19] W. Samek, M. Kawanabe, and K.-R. Müller, "Divergence-based framework for common spatial patterns algorithms," *IEEE Rev. Biomed. Eng.*, vol. 7, pp. 50–72, 2014.

[20] W. Wu, Z. Chen, X. Gao, Y. Li, E. N. Brown, and S. Gao, "Probabilistic common spatial patterns for multichannel EEG analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 639–653, Mar. 2015.

[21] W. Wu, X. Gao, B. Hong, and S. Gao, "Classifying single-trial EEG during motor imagery by iterative spatio-spectral patterns learning (ISSPL)," *IEEE Trans. Biomed. Eng.*, vol. 55, no. 6, pp. 1733–1743, Jun. 2008.

[22] F. Qi, Y. Li, and W. Wu, "RSTFC: A novel algorithm for spatio-temporal filtering and classification of single-trial EEG," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3070–3082, Dec. 2015.

[23] H. Cecotti and A. Gräser, "Convolutional neural networks for P300 detection with application to brain-computer Interfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 433–445, Mar. 2011.

[24] H. Cecotti, "A time–frequency convolutional neural network for the offline classification of steady-state visual evoked potential responses," *Pattern Recognit. Lett.*, vol. 32, no. 8, pp. 1145–1153, 2011.

[25] H. Cecotti, M. P. Eckstein, and B. Giesbrecht, "Single-trial classification of event-related potentials in rapid serial visual presentation tasks using supervised spatial filtering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 11, pp. 2030–2042, Nov. 2014.

[26] S. Stober, D. J. Cameron, and J. A. Grahn, "Using convolutional neural networks to recognize rhythm stimuli from electroencephalography recordings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1449–1457.

[27] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahn. (Nov. 2015). "Deep feature learning for EEG recordings." [Online]. Available: http://arxiv.org/abs/1511.04306

[28] P. Bashivan, I. Rish, M. Yeasin, and N. Codella. (Nov. 2015). "Learning representations from EEG with deep recurrent-convolutional neural networks." [Online]. Available: http://arxiv.org/abs/1511.06448

[29] M. Hajinoroozi, Z. Mao, and Y. Huang, "Prediction of driver's drowsy and alert states from eeg signals with deep learning," in *Proc. IEEE 6th Int. Workshop Comput. Adv. Multi-Sensor Adapt. Process. (CAMSAP)*, Dec. 2015, pp. 493–496.

[30] M. Hajinoroozi, Z. Mao, T.-P. Jung, C.-T. Lin, and Y. Huang, "EEG-based prediction of driver's cognitive performance by deep convolutional neural network," *Signal Process., Image Commun.*, vol. 47, pp. 549–555, May 2016.

[31] X. An, D. Kuang, X. Guo, Y. Zhao, and L. He, "A deep learning method for classification of EEG data based on motor imagery," in *Intelligent Computing in Bioinformatics*. Springer, 2014, pp. 203–210.

[32] H. Yang, S. Sakhavi, K. K. Ang, and C. Guan, "On the use of convolutional neural networks and augmented CSP features for multi-class motor imagery of EEG signals classification," in *Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2015, pp. 2620–2623.

[33] S. Sakhavi, C. Guan, and S. Yan, "Parallel convolutional-linear neural network for motor imagery classification," in *Proc. 23rd Eur. Signal Process. Conf. (EUSIPCO)*, Aug./Sep. 2015, pp. 2736–2740.

[34] M. Tangermann *et al.*, "Review of the BCI Competition IV," *Frontiers Neurosci.*, vol. 6, p. 55, Jul. 2012.

[35] M. Le Van Quyen *et al.*, "Comparison of Hilbert transform and wavelet methods for the analysis of neuronal synchrony," *J. Neurosci. Methods*, vol. 111, no. 2, pp. 83–98, Sep. 2001.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[37] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, Jun. 2015.

[38] H. Bashashati, R. K. Ward, and A. Bashashati, "User-customized brain computer interfaces using Bayesian optimization," *J. Neural Eng.*, vol. 13, no. 2, p. 026001, Jan. 2016.

[39] X. Xie, Z. L. Yu, H. Lu, Z. Gu, and Y. Li, "Motor imagery classification based on bilinear sub-manifold learning of symmetric positive-definite matrices," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 6, pp. 504–516, Jun. 2016.

[40] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[41] M. D. Zeiler and R. Fergus. (2013). "Visualizing and understanding convolutional networks." [Online]. Available: https://arxiv.org/abs/1311.2901

[42] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *Proc. Int. Conf. Mach. Learn.-Deep Learn. Workshop*, Jun. 2015, p. 12. [Online]. Available: http://arxiv.org/abs/1506.06579

[43] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Proc. ICLR*, Dec. 2014, p. 1. [Online]. Available: http://arxiv.org/abs/1312.6034

[44] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, p. e0130140, Jul. 2015.

[45] I. Sturm, S. Bach, W. Samek, and K.-R. Müller. (Apr. 2016). "Interpretable deep neural networks for single-trial EEG classification." [Online]. Available: http://arxiv.org/abs/1604.08201

[46] H. Mhaskar, Q. Liao, and T. Poggio. (Mar. 2016). "Learning functions: When is deep better than shallow." [Online]. Available: http://arxiv.org/abs/1603.00988

[47] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. (Jun. 2016). "On the expressive power of deep neural networks." [Online]. Available: http://arxiv.org/abs/1606.05336

Siavash Sakhavi (S'13) received the master's degree from the Sharif University of Technology, Tehran, Iran, in 2012, and the Ph.D. degree from a joint program between the National University of Singapore, Singapore, and the Agency for Science, Technology and Research, Singapore, in 2018.

During his Ph.D. degree, he was with researchers at the Institute for Infocomm Research, Singapore, and the Neural and Biomedical Technology Department, involved in designing and utilizing deep learning methods for EEG signal processing and brain–computer interfaces. His research interests include biosignal processing, biomedical image processing, machine learning, and deep learning. His current research interests include applying new methodologies in machine learning to real-world problems.

Cuntai Guan (S'91–M'92–SM'03–F'18) received the Ph.D. degree from Southeast University, Nanjing, China, in 1993.

He has been a Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, since 2016. He is the Co-Director of the Rehabilitation Research Institute of Singapore, Singapore. He serves as a Chief Scientist with Neeuro Pte Ltd., Singapore. His current research interests include the fields of brain–computer interfaces, neural engineering, machine learning, data analytics, and neuro-technologies.

Dr. Guan is a member of the Singapore National Medical Research Council Open Fund IRG Review Panel. He was the founding Head of the Neural and Biomedical Technology Department and a Principal Scientist with the Institute for Inforcomm Research, Agency for Science, Technology and Research (A*STAR), Singapore. He was the Program Leader of neuro-technology at A*STAR. He had served as the Chairman of the IEEE Engineering in Medicine and Biology Chapter and also a President of the Pattern Recognition and Machine Intelligence Association. He received the Annual BCI Research Award, the IES Prestigious Engineering Achievement Award, the Achiever of the Year (Research) Award, and was a Finalist of the President Technology Award, and the National Infocomm Award. He is an Associate Editor for the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, IEEE ACCESS, *Frontiers in Neuroscience*, and *Brain–Computer Interfaces*, and a Guest Editor for the *IEEE Computational Intelligence Magazine*.

Shuicheng Yan (F'17) is the Vice President and a Chief Scientist with Qihoo 360 Technology Co., Ltd., and also the Head of the 360 Artificial Intelligence Institute, Beijing, China. He is also a tenured Associate Professor with the National University of Singapore, Singapore. He has authored/co-authored about 500 high quality technical papers, with a Google Scholar citation over 25 000 times and H-index 70. His current research interests include computer vision, machine learning, and multimedia analysis.

Mr. Yan is an IAPR Fellow and an ACM Distinguished Scientist. He was a TR Highly Cited Researcher of 2014, 2015, and 2016. His team received winner or honorable-mention prizes seven times in five years over PASCAL VOC and ILSVRC competitions, which are core competitions in the field of computer vision, along with more than 10 best (student) paper awards and especially a Grand Slam in the ACM MM, the top conference in the field of multimedia, including the Best Paper Award, the Best Student Paper Award, and the Best Demo Award.