




# Compressing Subject-specific Brain-Computer Interface Models into One Model by Superposition in Hyperdimensional Space

**Conference Paper****Author(s):**

[Hersche, Michael](#) ; [Rupp, Philipp](#); [Benini, Luca](#) ; [Rahimi, Abbas](#) 

**Publication date:**

2020

**Permanent link:**

<https://doi.org/10.3929/ethz-b-000387117>

**Rights / license:**

[In Copyright - Non-Commercial Use Permitted](#)

**Originally published in:**

<https://doi.org/10.23919/DATE48585.2020.9116447>

**Funding acknowledgement:**

780215 - Computation-in-memory architecture based on resistive devices (EC)

# Compressing Subject-specific Brain–Computer Interface Models into One Model by Superposition in Hyperdimensional Space

Michael Hersche, Philipp Rupp, Luca Benini, Abbas Rahimi  
Integrated Systems Laboratory, ETH Zurich, Switzerland  
Emails: rupp@student.ethz.ch, {hersche, benini, abbas}@iis.ee.ethz.ch

**Abstract**—Accurate multiclass classification of electroencephalography (EEG) signals is still a challenging task towards the development of reliable motor imagery brain–computer interfaces (MI-BCIs). Deep learning algorithms have been recently used in this area to deliver a compact and accurate model. Reaching high-level of accuracy requires to store subjects-specific trained models that cannot be achieved with an otherwise compact model trained globally across all subjects. In this paper, we propose a new methodology that closes the gap between these two extreme modeling approaches: we reduce the overall storage requirements by superimposing many subject-specific models into one single model such that it can be reliably decomposed, after retraining, to its constituent models while providing a trade-off between compression ratio and accuracy. Our method makes the use of unexploited capacity of trained models by orthogonalizing parameters in a hyperdimensional space, followed by iterative retraining to compensate noisy decomposition. This method can be applied to various layers of deep inference models. Experimental results on the 4-class BCI competition IV-2a dataset show that our method exploits unutilized capacity for compression and surpasses the accuracy of two state-of-the-art networks: (1) it compresses the smallest network, EEGNet [1], by  $1.9\times$ , and increases its accuracy by 2.41% (74.73% vs. 72.32%); (2) using a relatively larger Shallow ConvNet [2], our method achieves  $2.95\times$  compression as well as 1.4% higher accuracy (75.05% vs. 73.59%).

## I. INTRODUCTION

Brain–computer interfaces (BCIs) aim to provide a communication and control channel based on the recognition of the subjects intentions from neural activity typically recorded by noninvasive electroencephalogram (EEG) electrodes [3]. What makes it particularly challenging, however, is its susceptibility to errors in the recognition of human intentions, especially during motor imagery (MI) [4], [5]. MI-BCIs strive to decode the cognitive process of thinking of a motion, e.g., the left or the right hand movement, without actually performing it. Recently, deep learning algorithms strive to decode EEG signals to deliver compact yet accurate MI-BCI models [1], [2], [6], [7].

To further improve classification accuracy and deal with the high variability of EEG signals between subjects, most approaches [2], [5], [6], [8], [9] train a *personalized model* per subject. This subject-specific model training requires a calibration session for each subject, which can be exhausting for clinical users with impaired cognitive abilities, or tedious for healthy users [10]. On other hand, one may train a global model for all subjects eliminating the calibration procedure. However, the global model often comes with a significant

degradation in classification accuracy. For instance, an accuracy loss of 9.22% is observed when using a global model instead of subject-specific models [6].

Further, subject-specific models demand separate memory to store the trained model per subject. Each model size could be as large as 103k parameters for execution on a neuromorphic TrueNorth chip [11]. The Shallow ConvNet [2], one of the smaller convolutional neural networks (CNN) for MI-BCIs, stores 47k parameters per subject. For a multi-subject BCI device, this memory requirement increases linearly with the number of subjects and poses limitations on resource-limited edge devices. This memory may be compressed to some degree by quantizing or sparsifying the parameters [12]. Hence a complementary approach that produces a single model to combine the best of both worlds (i.e., the compactness of a global model, and the accuracy of subject-specific models) is highly desirable.

One viable option is to exploit the hyperdimensional space [13] by *holistically* taking into account the parameters available in the model. Such large number of parameters can naturally form a holographic representation in which randomly drawn  $d$ -dimensional vectors ( $d > 1,000$ ) can be reliably superimposed and decomposed [14]. In this hyperdimensional space, we can learn a separate set of parameters for each subject, these parameters—arranged in a set of  $d$ -dimensional vectors—can be stored in superposition with each other, resulting in a composite  $d$ -dimensional vector that requires almost the same number of parameters as a model for a single subject [15].

We apply the concept of superposition in hyperdimensional space (Section II-B) to compress the weights of MI-BCI deep models (Section II-A) from multiple subjects into a single model. This paper makes the following contributions:

- We present a universal compression approach that applies to any layer in the network, including the fully connected, and convolutional layers. We assign to each subject a pseudorandom hyperdimensional vector that nearly orthogonalizes the subject-specific model after a binding operation, allowing for superposition of many such models into a single model without interference. Moreover, our method is non-intrusive to the operation of the network because retrieval of subject-specific weights is done by unbinding the single model with the

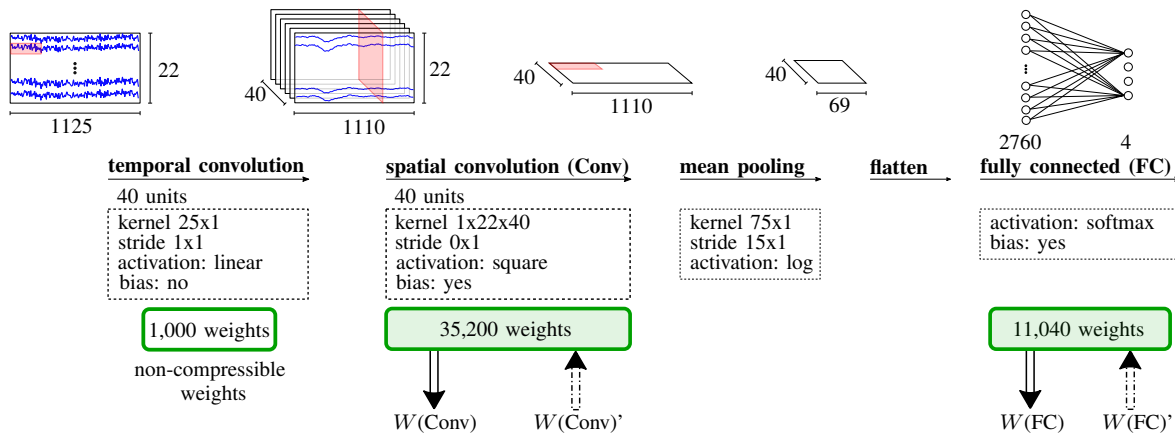


Fig. 1: Shallow ConvNet [2] with detailed kernel description and number of weights applied on 4-class BCI IV-2a dataset. Weights of spatial convolution (Conv) and fully connected layer (FC) are compression friendly and can be extracted and vectorized to a *hypervisor*, or inserted into the network again.

subject key at loading stage of the model. Section III describes our method in details.

- We propose an iterative retraining procedure to compensate for the accuracy loss of decomposed model due to noisy retrieval (Section III-B). The procedure adapts the pre-trained superimposed weights as well as non-compressed weights of every subject, followed by an altering random subject sequence selection.
- We evaluate our method on two widely-used networks using the 4-class BCI competition IV2a dataset [16] in Section IV. Our experimental results show that for both networks, our method finds a smaller set of parameters that is more accurate. Subject-specific EEGNet [1] models, as the smallest network, are compressed by  $1.9\times$  into a single—still subject-specific—model with 9,620 parameters, delivering 2.41% higher accuracy than the uncompressed models. This is achieved by superposition of the fully connected layers. Using a relatively larger Shallow ConvNet [2], our method finds an optimal set of parameters resulting in  $2.95\times$  compression and 1.4% higher accuracy by superimposing the spatial convolutional layers. This leads to a single model with 144,316 parameters achieving 75.05% accuracy (vs. 73.59% of uncompressed models). Our code is available.<sup>1</sup>

## II. RELATED WORK AND BACKGROUND

### A. Shallow ConvNet and EEGNet in MI-BCI

Motor imagery (MI) is still one of the most challenging paradigms to connect the brain with an external device. The main challenge in MI is the high variance in data between different subjects as well as between different recording sessions of the same subject.

Traditional MI-BCI architectures are divided into feature extraction and a subsequent classifier. EEG signals are typically pre-processed using tunable spectral and spatial filters followed by log-energy feature extraction, with filter bank common spatial pattern (FBCSP) [5] and Riemannian covariance matrices [9] being the most popular feature extrac-

tors. The multi-spectral features are classified using a linear discriminant analysis (LDA), regularized LDA, or support vector machines (SVMs) [10]. A linear SVM on more than 32k Riemannian features, leading to overall 1.751M trainable parameters, has achieved so far the highest classification accuracy of 75.47% [8] on the 4-class MI-BCI competition IV-2a dataset [16].

Alternatively, the feature extractor and classifier can be combined and trained simultaneously with a convolutional neural network (CNN). While being successful in image classification, CNNs are gaining attention in MI-BCIs as well [10]. Schirrmeister et al. [2] provides an elaborate study on CNN architectures for MI-BCI, where the small Shallow ConvNet achieves an accuracy of 73.59% on the 4-class dataset.

Fig. 1 shows the architecture of the Shallow ConvNet, which is inspired by the classic spectral and spatial filtering with log-energy features [2]. The input feature map represents the EEG signal in the time domain with 1125 samples ( $4.5\text{ s} \times 250\text{ Hz}$ ) and 22 EEG channels. The samples are filtered in time-domain (1,000 weights), spatial domain (35,200 weights) with square activation, pooled with log activation, and classified with fully connected layer (11,400 weights). Overall, this CNN architecture has 47,324 parameters including weights and biases, where spatial convolution and fully connected layers are the largest, making them most appealing for compression.

Yet another smaller network is EEGNet [1] with 1,716 trainable parameters. The main difference to Shallow ConvNet is that EEGNet uses spatial separable convolutions and more pooling layers, which reduces the number of weights of the convolutional layer and the size of the fully connected layer, respectively. EEGNet enables not only classification of MI, but also of P300 event-related potential, feedback error-related negativity, and movement-related cortical potential. Its flexibility and small size, however, comes at the cost of significantly lower accuracy, e.g., 67% for MI. Effort has been done to modify EEGNet by changing the pooling layers and expanding the network to 2,036 trainable parameters for achieving 72% accuracy with subject-specific models [17].

<sup>1</sup><https://github.com/MHersche/bci-model-superpos>

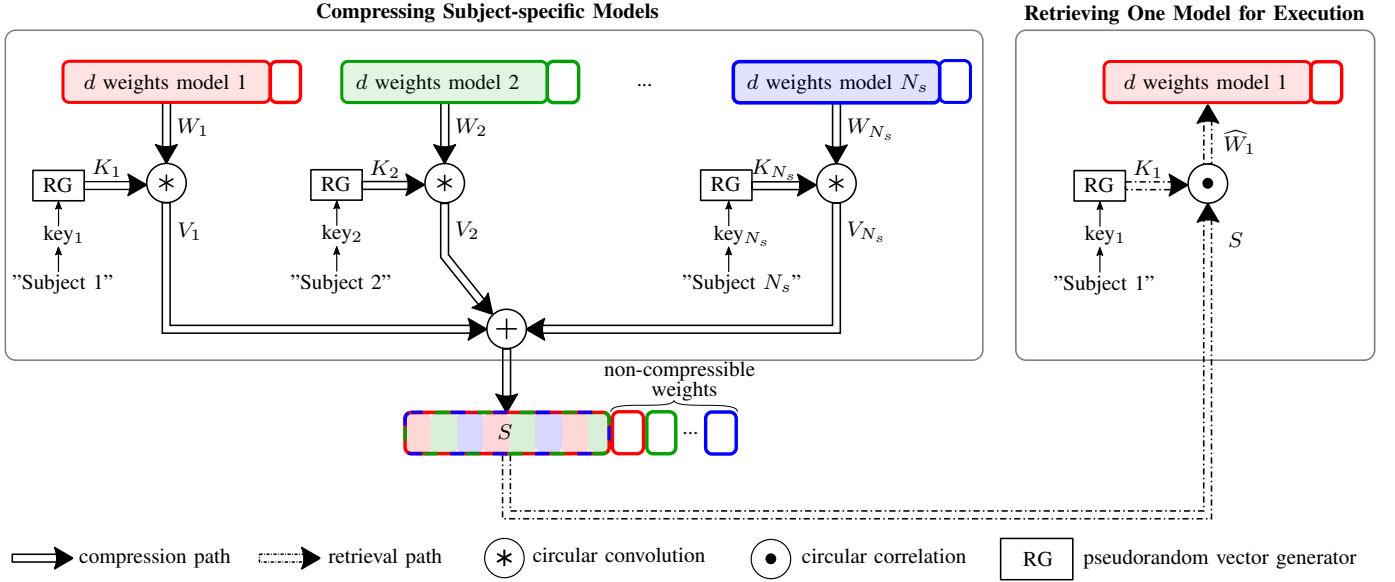


Fig. 2: Compression of  $N_s$  models and retrieval to execute model 1. The compressible weights of model 1 are extracted from the network and vectorized to a  $d$ -dimensional hypervector  $W_1$ , bound with a key hypervector  $K_1$  and superimposed with the bound weights of all other models to form a single model  $S$ . The superposition  $S$  is the compressed version of  $W_1, W_2, \dots, W_{N_s}$ . To execute model 1, its weights are retrieved by unbinding  $S$  with  $K_1$  and reloaded into the network. All double lines are connections of dimension  $d$ .

### B. Superposition in Hyperdimensional Space

Computing in hyperdimensional space [13] is all about manipulation and comparison of large patterns with a well-defined set of operations that provide multifaceted features. For instance, hyperdimensional computing constructs fast learning and low-energy models for ExG biosignal classification tasks [18]. Hyperdimensional computing goes beyond designing efficient classifiers by applying its rich operations to encapsulate deep learning models in a compressed representation. Here, we present the concept of hyperdimensional superposition for model compression [15] using Plate’s holographic reduced representation [14]. The basic idea is to extract parameters, or weights, from a pre-trained model, vectorize them to a hyperdimensional vector, bind the vector with a random key, and superimpose multiple bound vectors into a single vector. Retrieval of individual weights is guaranteed, if the dimension of the vectors is high, i.e.,  $d > 1,000$ , denoted as *hypervectors*.

We define the *binding* of a *weight* hypervector  $W$  with a *key* hypervector  $K$  as

$$V = K \otimes W, \quad (1)$$

where  $\otimes$  is the binding operator and  $V, W, K \in \mathbb{R}^d$  are  $d$ -dimensional hypervectors. There exists a vast variety of binding operators; we use *circular convolution* used in original holographic representation [14]. Circular convolution can be understood as a compression of the outer-product between  $K$  and  $W$  and is efficiently implemented with Latin squares [19]. This binding operator generates a key-value pair. The approximate inverse is the *unbinding* operation defined as

$$W \approx K \odot V, \quad (2)$$

where  $\odot$  corresponds to the *circular correlation*. A key hypervector is generated by sampling from a normal distribution with variance  $1/d$  yielding a random hypervector with unit norm.

Binding a hypervector  $W$  with a random key hypervector  $K$  generates  $V$ , which has small cosine similarity to  $W$  with very high probability if the dimension  $d$  is reasonably high [14]. After binding,  $V$  and  $W$  become *quasiorthogonal* [13]. This concept is leveraged to generate quasiorthogonal hypervectors even though they may be arbitrarily close in their original space. For instance,  $V_1 = W_1 \otimes K_1$  and  $V_2 = W_2 \otimes K_2$  have small cosine similarity, independent of their original relation if  $K_1$  and  $K_2$  are two randomly drawn key hypervectors.

Quasiorthogonality allows *superposition* of multiple key-value pairs:

$$S = \sum_i K_i \otimes W_i, \quad (3)$$

where  $S \in \mathbb{R}^d$  represents the composition of all  $W_i$ . A specific weight hypervector  $W_j$  is *retrieved* by unbinding  $S$  with the corresponding key

$$\widehat{W}_j = S \odot K_j. \quad (4)$$

The retrieval is *noisy* due to two error sources, namely the terms originating from other key-value pairs as well as the error from the unbinding operation itself:

$$\widehat{W}_j = K_j \odot (K_j \otimes W_j) + \sum_{j \neq i} K_j \odot (K_j \otimes W_j). \quad (5)$$

An auto-associative memory would allow for a perfect retrieval using a *clean-up* procedure, where  $\widehat{W}_j$  is compared against

all possible value hypervectors stored in the associative memory [13]. The hypervector with the highest cosine similarity to  $\widehat{W}_j$ , i.e.,  $W_j$  is finally returned.

### III. SUPERPOSITION AND RETRIEVAL IN HYPERDIMENSIONAL SPACE

This section presents the main contribution of this paper. We first describe how hyperdimensional superposition is used to compress CNN weights of multiple pre-trained MI-BCI networks. We then propose a retraining method to further improve the accuracy of compressed models.

#### A. Model Compression with Superposition

We exploit hyperdimensional superposition and noisy retrieval to *superimpose weights* of neural network models without clean-up. This eliminates the need of an associative memory, similar to [15], but imposes noisy retrieval that can be improved by an iterative retraining (Section III-B). Fig. 2 illustrates our proposed superposition and retrieval process for  $N_s$  separate models. From every model, we extract  $d$  weights and concatenate them to build a  $d$ -dimensional hypervector. Then, for every model, we generate a  $d$ -dimensional hypervector  $K_i$  using a pseudorandom number generator (RG) with seed corresponding to  $key_i$ . We need to store only the seed  $key_i$  instead of the actual key hypervector. This  $key_i$  needs a negligible 32-bit storage per model since  $K_i$  can be reproduced from the key by RG. The weights  $W_i$  are bound with the related hypervector  $K_i$ , resulting in  $V_i$  with weights quasi-orthogonal with respect to each other. Next, the hypervector  $S$  is computed by superposition of  $V_1, V_2, \dots, V_{N_s}$  which is a composite representation of  $W_1, W_2, \dots, W_{N_s}$ . Notably, the binding, unbinding, and superposition operators do not change the dimensionality of the hypervectors at any point, thus keeping the space *closed*. For inference, or executing model 1, the weights are retrieved by unbinding  $S$  with  $K_1$ . Finally, the retrieved weights  $\widehat{W}_1$  are inserted into the network. During execution of the model, no additional binding/unbinding operations are required; therefore, our method is *non-intrusive* to the operation of the network.

The presented procedure is used to *compress*  $N_s$  subject-specific CNN models. First, we train a specific CNN per subject, which gives  $N_s$  different sets of weights. We extract the weights from one or multiple layers per subject and superimpose the vectorized weights among all subjects. Any layer can be added or removed from compression, resulting in the dimension of the weight hypervector to change. The weights of the remaining layers as well as all biases are left non-compressed. To store all models, we need to keep the non-compressed weights as well as the hyperdimensional superposition  $S$  (see Fig. 2). For executing the MI-BCI on subject  $i$ , we locally generate the key hypervector  $K_i$  using the same RG with subject-dependent seed and use it for retrieval of the compressed weights. We then reload the CNN consisting of layers with non-compressed and retrieved weights.

We define the compression ratio as

$$CR = \frac{N_s \cdot \text{model size}}{N_s \cdot (\text{non-compressed weights}) + d} \quad (6)$$

$$= \frac{N_s}{N_s(1-r) + r} \quad (7)$$

where  $N_s$  is the number of models and  $r = \frac{d}{\text{model size}}$  is the ratio between the number of compressed weights, and all the weights available in one model.

Noisy retrieval of hypervectors introduces errors on the weights, which influence the accuracy of the CNN. For example, our experiments have shown that accuracy drops by 21.15% when compressing the fully connected layer in the 4-class MI dataset. The error of retrieval gets smaller when increasing the dimension of the hypervector. On other hand, increasing the dimension means compressing more weights and therefore introducing more errors into the network. Moreover, the network is highly sensitive to *which* layers are introduced with errors. Thus, a sensitivity analysis is required to find a trade-off between compression ratio and performance degradation of the model. We find that the sweet spot lies in the middle of the network, leaving input layer as well as the last classification layer untouched. This is aligned with multiple studies which quantize neural network models and explained in [20]. Nevertheless, we show that even the fully connected classification layer can be compressed without any loss in accuracy using retraining.

#### B. Retraining to Compensate Noisy Retrieval

We propose an iterative retraining procedure to recover, and also enhance, the performance of the compressed model. The idea is to add a retraining procedure which includes noisy retrieval, therefore making the network *robust* against introduced errors in weights. The retraining is done offline and does not have any affect on the execution of the model.

Algorithm 1 describes the retraining procedure. The innermost loop shows one retraining iteration for all subjects, which are retrained sequentially. First, the compressed weights are retrieved and inserted to the network of subject  $s$ , which already has the non-compressed weights of subject  $s$  included. The model is retrained, where hyperparameters such as learning rate, batch size, and number of epochs are determined by a grid search in 5-fold cross-validation on the training set based on the highest validation accuracy. It is essential to retrain *all* layers in the model: when adjusting only the compressed layers and freezing the non-compressed ones, experiments showed that the performance does not recover. Finally, the superposition  $S$  is updated with the retrained weights, and the uncompressed weights of network  $s$  are saved. The retraining is repeated for  $N_I$  iterations. In every iteration, the order of the subjects is shuffled randomly, which increases the accuracy of retraining by 2%.

Fig. 3 shows the training and validation misclassification rate during retraining on the BCI competition IV-2a training set for subject 1 as well as for all subjects on average. Before starting with retraining at iteration 0, the validation misclassification rate is 60% on average, which is close to chance level at 75%. This supports the necessity of retraining. In the first couple of iterations, training misclassification rate is above 25% but drops to zero quickly. The validation misclassification, however, decreases only slowly but gets close to the baseline validation misclassification without compression. In this example, retraining of subject 1 decreases validation misclassification beyond the baseline and therefore even improves the performance compared to no compression.

**Algorithm 1:** Iterative retrieval and retraining of CNN weights to improve accuracy of compressed models.

**input :**  $N_s$  - number of subjects (models superimposed)  
 $N_I$  - number of retraining iterations  
 $S$  - initial superimposed pre-trained weights  
 $K_s, s \in \{1, \dots, N_s\}$  - key hypervectors  
 $\mathfrak{N}$  - set of all pre-trained CNN models

**output:**  $S$  - retrained superimposed weights  
 $\mathfrak{N}$  - set of all retrained models

```

1 for ( $i = 1 : N_I$ ) do
2   for ( $s = 1 : N_s$ ) do
3      $\widehat{W}_s[i] = S \odot K_s$ 
4     network =  $\mathfrak{N}$ .getNetwork( $s$ )
5     network.setWeights( $\widehat{W}_s[i]$ )
6     network.retrain()
7      $W_s[i] =$  network.getWeights()
8      $S += (W_s[i] - \widehat{W}_s[i]) \otimes K_s$ 
9   end
10 end

```

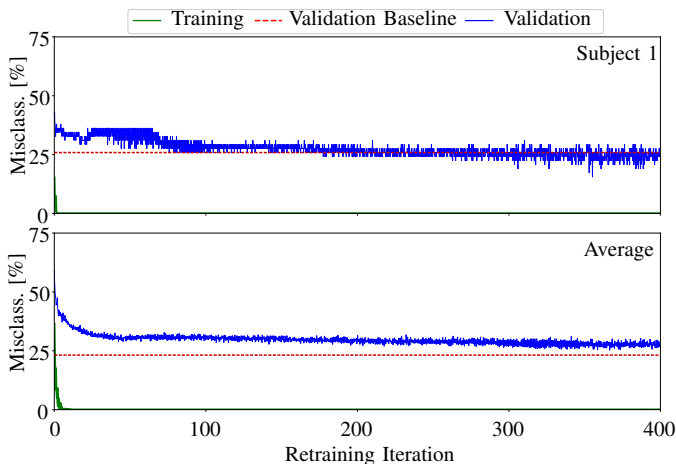


Fig. 3: Misclassification rate in % during retraining of compressed Shallow ConvNet in cross-validation of BCI competition IV-2a training set of subject 1 and average over all subjects. The validation baseline shows the validation performance of subject-specific model. With retraining of compressed models the training misclassification drops to zero and the validation misclassification converges close to the baseline.

## IV. EXPERIMENTAL RESULTS

### A. BCI Competition IV-2a dataset

We use the BCI Competition IV-2a dataset [16] consisting of EEG recordings from 9 different subjects. The subjects were requested to carry out four different MI tasks, namely the imagination of the movement of the left hand, right hand, both feet, and tongue. Two sessions were recorded on two different days. For each subject a session consists of 72 trials per class yielding 288 trials in total. One session is used for training and the other for testing exclusively. The signal was recorded using 22 EEG electrodes according to the 10-20

TABLE I: Classification accuracy in %, and compression ratio (CR) on 4-class MI-BCI competition IV-2a testset by compressing fully connected (FC) and/or spatial convolutional (Conv) layer of Shallow ConvNet and EEGNet. p-value reports the significance of Wilcoxon signed-rank test relative to subject-specific model. Retraining hyperparameters such as number of retraining iterations (Niter), learning rate (LR), batch size (BS), and number of epochs per iteration (Epochs) where determined with cross-validation on the training set.

Comp. layer	Shallow ConvNet				EEGNet	
	-	FC	Conv	FC+Conv	-	FC
CR	<b>1</b>	<b>1.26</b>	<b>2.95</b>	<b>7.61</b>	<b>1</b>	<b>1.90</b>
Niter	N/A	1000	1000	1000	N/A	1000
LR	1e-3	2.5e-5	1.25e-4	2.5e-5	1e-3	1e-4
BS	64	64	32	128	64	64
Epochs	N/A	5	20	5	N/A	5
S1	78.81	87.51	78.81	64.58	84.36	85.77
S2	55.90	60.37	56.22	40.63	54.06	60.07
S3	90.28	85.40	89.57	75.35	87.91	85.71
S4	82.64	79.84	78.42	54.17	63.16	66.23
S5	56.60	63.92	62.83	43.75	67.39	73.19
S6	50.69	55.95	52.35	44.44	54.88	53.49
S7	91.17	93.38	93.40	78.47	88.09	87.36
S8	77.78	80.18	81.61	67.71	76.75	80.81
S9	78.47	69.73	82.27	75.00	74.24	79.92
<b>Mean</b>	<b>73.59</b>	<b>75.14</b>	<b>75.05</b>	<b>60.46</b>	<b>72.32</b>	<b>74.73</b>
Std	15.26	13.13	14.52	14.98	13.26	12.28
p-value	-	0.515	0.208	0.008	-	0.066

system. It is bandpass filtered between 0.5 Hz and 100 Hz and sampled with 250 Hz. In addition, three electrooculography (EOG) channels give information about the eye movement. An expert marked trials containing artifacts based on the EOG signal. This way, 9.41% of the trials were excluded from the dataset. The number of trials per class remains balanced. We measure the classification accuracy as the ratio between correct classified trials over the total number of trials.

### B. Compression vs. Accuracy Loss

Table I shows the classification accuracy with different compressed layers applied to Shallow ConvNet and EEGNet. The hyperparameters for retraining are determined with a grid search in 5-fold cross-validation on the training set. For sake of completeness, the training parameters of non-compressed original models are added as well, even though we did not apply any retraining there.

In Shallow ConvNet, the compression of the fully connected layer without retraining results in an accuracy of 52.44%, which is a significant loss of 21.15% relative to no compression. Our retraining procedure is capable of recovering the accuracy to 75.14%, therefore even improving the accuracy of subject-specific models by 1.55%. This is the highest accuracy in this state-of-the-art deep network so far, and puts it on par with the 75.47% using Riemannian+SVM [8], while our model requires 5× fewer parameters. When replacing the spatial convolution layer with the fully connected layer for compression, we can achieve 2.95× compression with almost the same accuracy (i.e., 75.05%). As shown in Table I, the



compression ratio is increased from 1.26 to 2.95 when compressing the spatial convolution instead of the fully connected layer. The compression of both layers together yields the highest compression ratio of 7.69 but the accuracy significantly degrades to 60%. This degradation can be interpreted in two ways: 1) we reach a fundamental limit of how much the model can be compressed, or 2) the number of non-erroneous weights is too little in order to counteract the errors introduced by lossy compression. Our method is still effective on the much smaller EEGNet by reducing the number of parameters by  $1.9\times$  and achieving 2.41% higher accuracy relative to uncompressed EEGNet (74.73% vs. 72.32%).

The comparison between total number of parameters for all subjects and classification accuracy is summarized in Fig. 4. Subject-specific EEGNet models require an order of magnitude lower number of parameters than Shallow ConvNet, while achieving slightly lower accuracy. Training one global EEGNet model for all subjects further reduces the number of parameters, but comes at the cost of significantly lower accuracy of 70%. Our method, applicable to both networks, reduces the number of parameters *and* improves the accuracy at the same time.

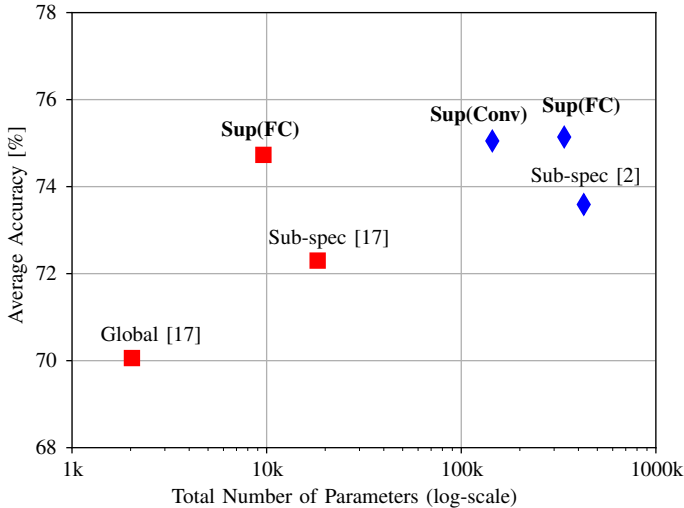


Fig. 4: Number of parameters to store CNN model of 9 subjects vs. average classification accuracy with Shallow ConvNet (◆) and EEGNet (■). Model weights are either global (Global), subject-specific (Sub-spec), superimposed at fully connected classification layer (Sup(FC)) or at spatial convolutional layer (Sup(Conv)).

## V. CONCLUSION

This paper demonstrates that hyperdimensional superposition can be used to compress already compact CNN models of MI-BCIs while improving classification accuracy at the same time. The noisy retrieval of weights introduces errors into the CNN causing severe degradation in accuracy, however, our proposed iterative retraining procedure recovers the performance with a compression ratio of  $\approx 3\times$ : nine subject-specific EEGNet models at 72.32% accuracy are compressed by  $1.9\times$  to a single model (9,620 parameters) at 74.73% accuracy; similarly, nine Shallow ConvNet models at 73.59%

are compressed by  $2.95\times$  to one model (144,316 parameters) at 75.05%. Moreover, hyperdimensional superposition can be used as a tool to identify compression friendly vs. non-compressible layers in larger CNNs. Ongoing work is focused on capacity analysis for a larger number of subjects.

## ACKNOWLEDGMENT

This project was supported in part by ETH Research Grant 09 18-2, by the Hasler Foundation under project no. 18082, and by EU's H2020 under grant no. 780215.

## REFERENCES

- [1] V. J. Lawhern, A. J. Solon *et al.*, "EEGNet: a compact convolutional neural network for EEG-based braincomputer interfaces," *Journal of Neural Engineering*, vol. 15, no. 5, p. 056013, 2018.
- [2] R. T. Schirrmester, J. T. Springenberg *et al.*, "Deep learning with convolutional neural networks for EEG decoding and visualization," *Human Brain Mapping*, vol. 38, no. 11, pp. 5391–5420, 2017.
- [3] B. Graimann, B. Allison *et al.*, "BrainComputer Interfaces: A Gentle Introduction." Springer, Berlin, Heidelberg, 2009, pp. 1–27.
- [4] M. Tangermann, K.-R. Müller *et al.*, "Review of the BCI Competition IV," *Frontiers in neuroscience*, vol. 6, p. 55, 2012.
- [5] Kai Keng Ang, Zhang Yang Chin *et al.*, "Filter Bank Common Spatial Pattern (FBCSP) in Brain-Computer Interface," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 2390–2397.
- [6] H. Dose, J. S. Moller *et al.*, "A Deep Learning MI - EEG Classification Model for BCIs," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 1676–1679.
- [7] Y. R. Tabar and U. Halici, "A novel deep learning approach for classification of EEG motor imagery signals," *Journal of Neural Engineering*, vol. 14, no. 1, p. 016003, 2017.
- [8] M. Hersche, T. Rellstab *et al.*, "Fast and Accurate Multiclass Inference for MI-BCIs Using Large Multiscale Temporal and Spectral Features," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 1690–1694.
- [9] A. Barachant, S. Bonnet *et al.*, "Classification of covariance matrices using a Riemannian-based kernel for BCI applications," *Neurocomputing*, vol. 112, pp. 172–178, 2013.
- [10] F. Lotte, L. Bougrain *et al.*, "A review of classification algorithms for EEG-based braincomputer interfaces: a 10 year update," *Journal of Neural Engineering*, vol. 15, no. 3, p. 031005, 2018.
- [11] E. Nurse, B. S. Mashford *et al.*, "Decoding EEG and LFP signals using deep learning," in *Proceedings of the ACM International Conference on Computing Frontiers - CF '16*. New York, New York, USA: ACM Press, 2016, pp. 259–266.
- [12] C. Sakr and N. Shanbhag, "Minimum Precision Requirements for Deep Learning with Biomedical Datasets," in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2018, pp. 1–4.
- [13] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation*, vol. 1, no. 2, 2009.
- [14] T. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, 1995.
- [15] B. Cheung, A. Terekhov *et al.*, "Superposition of many models into one," *arXiv:1902.05522 [cs]*, no. 1, 2019.
- [16] C. Brunner, R. Leeb *et al.*, "BCI competition 2008 - Graz data set A," doi: 10.1007/BF00994018.
- [17] A. Uran, C. van Gemeren *et al.*, "Applying Transfer Learning To Deep Learned Models For EEG Analysis," *arXiv:1907.01332 [cs, eess, stat]*, 2019.
- [18] A. Rahimi, P. Kanerva *et al.*, "Efficient biosignal processing using hyperdimensional computing: Network templates for combined learning and classification of exg signals," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 123–143, Jan 2019.
- [19] M. A. Kelly, D. Blostein *et al.*, "Encoding structure in holographic reduced representations," *Canadian Journal of Experimental Psychology*, vol. 67, no. 2, pp. 79–93, 2013.
- [20] A. G. Anderson and C. P. Berg, "The High-Dimensional Geometry of Binary Neural Networks," *International Conference on Learning Representations*, 2017.