A Bayesian-Optimized Convolutional Neural Network to Decode Reach-to-Grasp from Macaque Dorsomedial Visual Stream

A Bayesian-optimized Convolutional Neural Network to decode reach-to-grasp from macaque dorsomedial visual stream

Davide Borra[1[0000−0003−3791−8555]], Matteo Filippini[2[0000-0002-0730-4088]], Mauro Ursino[1,3[0000-0002-0911-0308]], Patrizia Fattori[2,3[0000-0002-0079-3755]] and Elisa Magosso[1,3[0000−0002−4673−2974]]

[1]Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi" (DEI), University of Bologna, Cesena Campus, Cesena, Italy
[2]Department of Biomedical and Neuromotor Sciences (DIBINEM), University of Bologna, Bologna, Italy
[3]Alma Mater Research Institute for Human-Centered Artificial Intelligence, University of Bologna, Bologna, Italy
davide.borra2@unibo.it

Abstract. Neural decoding is crucial to translate the neural activity for Brain-Computer Interfaces (BCIs) and provides information on how external variables (e.g., movement) are represented and encoded in the neural system. Convolutional neural networks (CNNs) are emerging as neural decoders for their high predictive power and are largely applied with electroencephalographic signals; these algorithms, by automatically learning the more relevant class-discriminative features, improve decoding performance over classic decoders based on handcrafted features. However, applications of CNNs for single-neuron decoding are still scarce and require further validation. In this study, a CNN architecture was designed via Bayesian optimization and was applied to decode different grip types from the activity of single neurons of the posterior parietal cortex of macaque (area V6A). The Bayesian-optimized CNN significantly outperformed a naïve Bayes classifier, commonly used for neural decoding, and proved to be robust to a reduction of the number of cells and of training trials. Adopting a sliding window decoding approach with a high time resolution (5 ms), the CNN was able to capture grip-discriminant features early after cuing the animal, i.e., when the animal was only attending the object to grasp, further supporting that grip-related neural signatures are strongly encoded in V6A already during movement preparation. The proposed approach may have practical implications in invasive BCIs to realize accurate and robust decoders, and may be used together with explanation techniques to design a general tool for neural decoding and analysis, boosting our comprehension of neural encoding.

Keywords: Neural decoding, Convolutional neural networks, Dorsomedial visual stream, V6A, Bayesian optimization.

1 Introduction

Neural decoding (i.e., prediction of observable output variables, such as movements or stimuli, from neural time series) is a central aim in neuroscience and in neural

engineering, for its practical and theoretical implications. Indeed, neural decoding is at the core of brain-computer interfaces (BCIs) where neural activity is translated into output commands for assistive and therapeutic purposes [1]. Moreover, neural decoding may advance our understanding of how information is represented and encoded in the neural system, as the accuracy of neural decoding reveals the amount of information the neural signals contain about an external variable (e.g., movement or sensation) and this can be evaluated across different brain regions and/or across different time intervals over the course of the sensory or sensorimotor task [2–8].

Currently, recent advances in machine learning, such as deep learning algorithms, are receiving growing attention for their predictive power in neural decoding applications [9]. Among deep learning architectures, Convolutional Neural Networks (CNNs) are preferentially adopted over other architectures (such as recurrent neural networks or fully-connected neural networks) for the classification of electroencephalographic signals (EEG) [10]. CNNs are feed-forward neural networks that learn convolutional filters to identify the features in the input signal that better discriminate among the predicted conditions. Attractive characteristics of these networks are a lower number of trainable parameters (e.g., vs. recurrent architectures), without hampering decoding performance, and an easier interpretability of the learned features in specific domains (e.g., temporal, spatial, and frequency domains) by using explanation techniques [6, 8]. CNNs have been successfully applied to EEG in a large spectrum of classification problems, such as emotion classification [11], event related potential detection and analysis [7, 8, 12, 13], motor execution/imagery classification [6, 14]. While CNN-based decoding of non-invasive neural recordings (EEG) have been widely investigated, CNNs are scarcely applied to single-neuron recordings acquired invasively in non-human mammals (in particular, non-human primates) and human patients. To overcome this limitation and to explore the potentialities of single-neuron decoding via CNNs, a recent study by Filippini et al. [2] designed and applied a CNN to decode the activity of neurons recorded from the posterior parietal cortex (PPC) of macaque (areas V6A, PEc, PE) while the animal performed a delayed reaching task towards 9 positions in the 3D space. PPC host areas integrating sensorimotor stimuli to dynamically guide the interaction with the surrounding environment and neurons in these areas are known to encode information regarding reaching endpoints, goals and trajectories [15–17]. In that study, Filippini et al. [2], employed a CNN whose configuration was optimized in its hyper-parameters (i.e., the parameters defining the functional form of the learning system, e.g., the number of convolutional kernels) via Bayesian optimization (BO), an efficient automatic hyper-parameter search algorithm. Results proved that the CNN was able to accurately decode the position of the reached points over the entire time course of the task, from target presentation to the end of reaching movement, with modulation across task phases and recording areas. Furthermore, the CNN outperformed a linear Naïve Bayes (NB) classifier suggesting that the CNN may represent a better framework to analyze how sensorimotor information are temporally encoded in neural representations. However, despite these promising results, the design and application of CNNs to motor decoding require a further validation on different motor tasks.

In this study, we aim to further validate the design and the application of CNNs to single-cell recordings, using a similar decoding workflow to the one adopted in [2]

(CNN architecture and BO for hyper-parameter search), while addressing a different motor decoding problem. Specifically, a CNN was used here to decode different grip types from the activity of V6A neurons, recorded while monkeys performed a delayed reach-to-grasp task towards objects of different shapes. V6A is a visuomotor area involved in the transformation of sensory information to guide prehension movements [18]. Its neurons are not only modulated by reaching targets located in different spatial positions but also by grasping information (e.g., grip type) and here we derived a Bayesian-optimized CNN configuration to decode such information and tested its decoding accuracy over the entire time course of the task, with a high time resolution (5 ms). Furthermore, we also evaluated the CNN performance while progressively using a reduced number of cells or a reduced number of training trials inside the recorded dataset, to test the robustness of the decoder by simulating practical scenarios where smaller datasets are available.

2 Methods

2.1 Dataset and pre-processing

This study reanalyzes the data obtained in [3], where single-cell activity was recorded from V6A area in two male Macaca fascicularis monkeys, using invasive electrode penetrations. The activity of 93 and 75 cells was recorded from the two monkeys (more details about the recording procedure can be found in [3]). Action potentials (spikes) were isolated and sampled at 100 KHz. Monkey sat on a primate chair with its head fixed in front of a rotating panel containing 5 different objects. The objects were chosen to evoke reach-to-grasp with different hand configurations and are illustrated in Fig. 1. These were: a ball (l_0: whole-hand prehension), ring (l_1: hook), plate (l_2: primitive precision grip), stick-in-groove (l_3: advanced precision grip), handle (l_4: finger prehension). Objects were presented to the monkey one at a time, in a randomized order. For each object, 10 trials were recorded, overall resulting in 50 trials per monkey. Each trial consisted of different phases, hereafter referred as 'epochs'. The trial started when the monkey pressed a 'home button' in complete darkness and then, the animal waited for instructions in darkness for 1 s (*free epoch*, epoch 0).

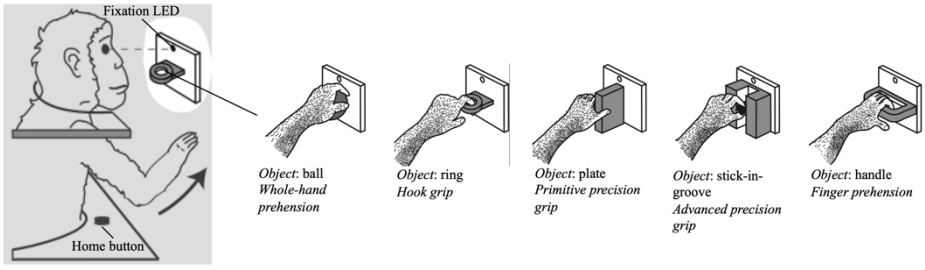

Fig. 1. Schematic representations of grip types and objects.

Subsequently, the fixation LED turned green, and the monkey had to wait for the LED to change its color to red, without performing any movement. After a fixation period of 0.5-1.0 s, LEDs surrounding the object to grasp turned on, illuminating the object. The monkey maintained the fixation on the LED without releasing the home button for a period of 1.5-2.0 s. The first 0.5 s portion of this period (corresponding to a first object visualization interval [3]) was excluded from the analysis. The remaining 1.0-1.5 s portion of this period formed the *delay epoch*, subdivided into the *early delay* (epoch 1) and *late delay* (epoch 2), by extracting 1 s epoch after the start of the delay epoch and before the end of the delay epoch, respectively. Then, the LED turned red, representing the go-signal for the reach-to-grasp movement (*reaction time epoch* and *reach-to-grasp epoch,* epochs 3,4, respectively). Once performed the movement, the monkey had to keep holding (*hold epoch*, epoch 5) the grasped object until the LED switched off (0.8-1.2 s). The LED switch-off cued the monkey to release the object and press the home button again, starting a new trial with a different object to reach and grasp.

For each neuron and each trial, spikes were binned at 5 ms. As trials and epochs may have a different duration across different neurons and trials, to obtain the same number of bins across neurons and trials, the average number of bins for each epoch was computed, then, the activity of each neuron and trial was re-binned using that number of average bins per epoch (thus, slightly changing respect to the original 5 ms binning). Then, firing rates were computed; thus, in this study the multi-variate neural activity was described by means of neuron firing rate. Firing rates recorded during the early delay, late delay, reaction time, reach-to-grasp, and hold epochs were collected and used in this study. Ten-fold stratified cross-validation was applied to partition the dataset of each monkey. Therefore, within each fold, 5 trials (one for each grip) were used as test set; then, 5 (one for each grip) and 40 of the remaining trials were used as validation and training sets, respectively.

2.2 Sliding window neural decoding

To analyze the temporal dynamics of reach-to-grasp encoding inV6A, a sliding window decoding approach was applied, as performed in previous studies [2–5] that analyzed the neural activity (neuron firing rates or EEG) by using the prediction of a machine learning algorithm as measure of neural encoding of motor- or cognitive-related brain states. This approach consists in decoding small portions of neurons' signals (hereafter referred as 'chunks') within each single recorded trial, enabling the study of the time course of neural encoding across all the phases of the task. To this aim, the neurons' firing rates were processed as follows, for each monkey (see upper panel of Fig. 2).

Let X_t be the t-th trial of shape (N, T), representing the multi-variate neural activity, where N is the number of recorded neurons (different across animals, here $N = 93$ and $N = 75$) and T is the number of time samples in the trial ($T = 812$ and $T = 816$ for the two monkeys, having 5 ms resolution). Overlapped chunks $X_{t,i}$ of shape (N, T_z) were extracted with a stride of T_s, where T_z is the number of time samples of each chunk. Each sampled chunk ($X_{t,i}$) was fed as input to the neural decoder.

$$X_{t,i} = X_t[:, iT_s : iT_s + T_z - 1], 0 \leq i \leq M - 1, \quad (1)$$

where i is the chunk index and M is the total number of chunks that can be extracted using T_z and T_s as chunk size and stride, respectively, i.e., $M = (T - T_z)/T_s + 1$. T_z and T_s are hyper-parameters of the decoding approach and were set to $T_z = 60$ (=300 ms) and $T_s = 10$ (=50 ms) during the training phase of the decoder (as in [2]), while $T_s = 1$ (=5 ms) during the testing phase. That is, during training a higher stride was used to speed up the computation, while during testing chunks were extracted with the maximum overlap, producing an inference with a high time resolution of 5 ms-step.

The addressed decoding problem was the classification of 5 different grip types from neuron firing rates (5-way classification). Each trial X_t was associated to a single label corresponding to the specific shape of the object the monkey had to grasp in that trial, i.e., $y_t \in L = \{l_k\}, 0 \leq k \leq 4$ (see Section 2.1 for the association label ID-grip type). Therefore, while performing sliding window decoding, the label associated to each sampled chunk ($y_{t,i}$) was the one associated to the trial the chunk was extracted from.

$$y_{t,i} = y_t, 0 \leq i \leq M - 1. \tag{2}$$

The CNN can be described by a probabilistic model $f(X_{t,i}; \vartheta, h)$: $\mathbb{R}^{N \times T_z} \rightarrow L$ parametrized in the trainable parameters and hyper-parameters contained in the arrays ϑ, h, respectively. In this study, monkey-specific decoders were designed, by using monkey-specific datasets to tune trainable parameters and hyper-parameters of CNNs. Hyper-parameters must be set before the model training starts; these parameters are optimized on the validation set via the hyper-parameter search procedure. Trainable parameters are the collection of weights and biases that model connections across the artificial neurons included in the network; these are learned on the training set during the network training.

2.3 Architecture and parameter tuning of the CNN

Architecture. The adopted CNN topology is inspired from the architecture recently proposed for the decoding of reaching targets from V6A neuron activity [2]. The CNN is composed by two modules and a schematization is reported in Fig. 2 (lower panel).

The first module (*convolutional module*) includes only trainable sparse connections across artificial neurons. This is composed by N_b blocks; each block in turn is composed by the sequence of N_c 2-D convolutional layers. The very first convolutional layer of the architecture performed convolutions in both space and time domains (mixed spatio-temporal convolutions) using kernels of size (N, F), while all other convolutional layer performed convolutions in the time domain using kernels of size $(1, F)$. Each layer learned K kernels using unitary stride and a padding of $(0, F//2)$, where // is the floor division operator. After each layer, batch normalization [19] is optionally included and, then, Exponential Linear Unit (ELU) non-linearity [20], i.e., $f(x) = x, x > 0$ and $f(x) = \exp(x) - 1, x \leq 0$, is applied. Between each block, average pooling with a pool size of $(0, 2)$ is performed, halving the temporal dimension, and then dropout [21] with dropout rate p_{drop} is included. Over the network, batch normalization

and dropout act as regularizers to improve model generalization. The main hyper-parameters of this module were searched via BO and are detailed in Table 1.

The second module (*fully-connected module*) includes only trainable dense connections across neurons. In particular, feature maps provided by the convolutional module are flattened and given as input to a fully-connected layer with 5 neurons, corresponding to the output layer. This layer is activated with a softmax activation function to convert neuron outputs into the conditional probabilities $p(l_k|X_{t,i}), 0 \leq k \leq 4$, and then, the predicted class is the one with the highest probability.

Table 1. Hyper-parameters of the convolutional module searched with Bayesian optimization: distributions and admitted values.

Hyper-parameter	Distribution	Values
No. of blocks (N_b)	uniform	[1,2]
No. of conv. layers (N_c)	uniform	[1,2,3]
No. of kernels (K)	uniform	[4,8,16,32]
Kernel size (F)	uniform	[(N,11), (N,21), (N,31), (N,41)]
Dropout rate (p_{drop})	uniform	[0, 0.25, 0.5]
Use batch norm.	uniform	[False, True]
Learning rate (lr)	log-uniform	[1e-4, 5·1e-4, 1e-3, 5·1e-3, 1e-2]

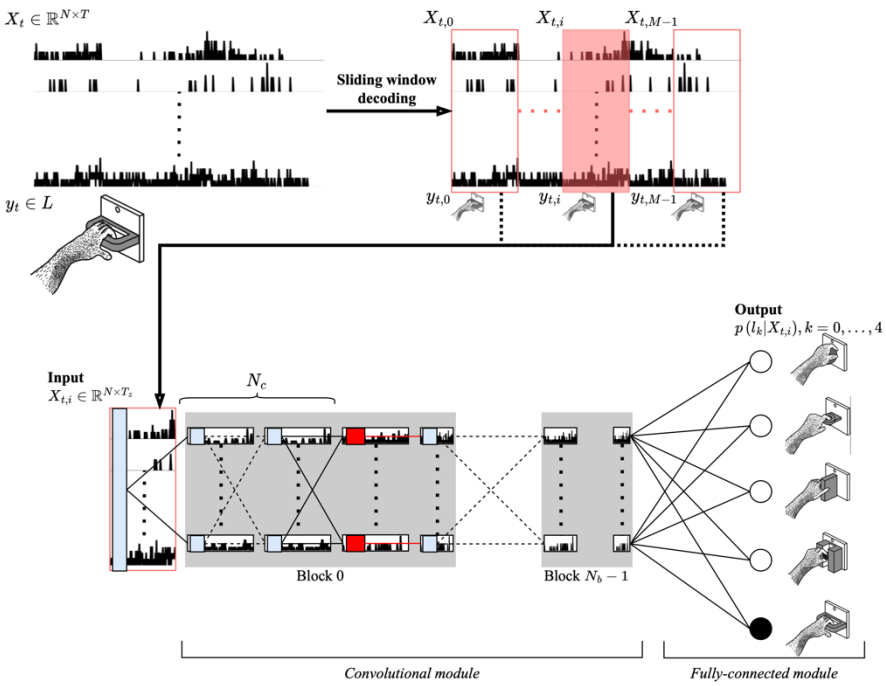

Fig. 2. Sliding window decoding approach and CNN architecture. In the CNN structure, only the main layers are displayed (i.e., only convolutional, pooling and fully-connected layers) with their output feature maps. Convolutional kernels are displayed with light-blue boxes, while pooling kernels are displayed with red boxes. Convolutional blocks are displayed as grey boxes; note that for brevity, only the first convolutional block (block 0) is detailed in its layers, while in the last block (block $N_b - 1$) only the first and last feature maps are displayed.

Hyper-parameter optimization. Hyper-parameter optimization is devoted to find the optimal hyper-parameters of a learning system on a validation set (different from the training and test sets). Denoting with h the array containing the hyper-parameters to search, hyper-parameter optimization finds the h^* that minimizes an objective function $k(h)$ on the validation set, i.e., $h^* = argmin_h k(h)$. In this study, we used $k(h) = 1 - acc(h)$ as objective function, where $acc(h)$ is the accuracy (averaged across all time samples and across epochs) obtained with a specific hyper-parameter configuration; for each configuration, a new training (as specified in Section Trainable parameter optimization) and evaluation stages must be performed, increasing the computational cost.

Hyper-parameter search algorithms (e.g., grid search and random search) generally search h^* without exploiting results from past iterations to select the array h to be evaluated in the next iteration (uninformed algorithms), often wasting time on unpromising h values. BO [22] overcomes this limitation, by suggesting, in an informed way, the next hyper-parameters h to be evaluated using a selection criterion, and thus investigating only hyper-parameters that seem promising based on past evaluations. In particular, a Bayesian statistical model $p(k|h)$ of the objective function (surrogate model) is used and it is updated after each iteration by keeping track of past evaluation results (i.e., each pair $h, k(h)$). Crucially, this surrogate model is easier to optimize than the objective function $k(h)$ [22].Thus, the next set of hyper-parameters to be evaluated on the actual objective function is chosen by selecting the hyperparameters that perform best on the current surrogate model. The criterion used to optimize the surrogate is called 'selection function'. BO was performed for 100 iterations by using tree-structured Parzen estimator as surrogate model and expected improvement as selection function. A more complete description of BO used to tune hyper-parameters of a CNN for neural decoding can be found in [2]. BO was performed for each cross-validation fold (10 in total) and each monkey (2 in total), leading to 20 optimal hyper-parameter configurations.

Trainable parameter optimization. The cross-entropy between the predicted probability distribution (provided by the probabilistic learning system) and the empirical distribution (provided by the labelled dataset) was used as loss function ($j(\vartheta)$) while learning the trainable parameters. Adam [23] was used as optimizer, searching for $\vartheta^* = argmin_\vartheta j(\vartheta)$. The learning rate was selected via BO (see Table 1) together with the other searched hyper-parameters. Furthermore, the mini-batch size was set to 64 and the maximum number of training epochs to 250. The optimization stopped when the validation accuracy did not decrease after 50 consecutive epochs (early stopping).

2.4 Analysis of decoding performance

For each hyper-parameter, the most frequent value across the configurations selected via BO was derived; thus, we identified a single hyper-parameter configuration where each hyper-parameter was set to the value occurring more frequently during hyper-parameter search. This hyper-parameter configuration was adopted for the CNN; then, for each monkey and each fold, the so designed CNN was trained in 3 different conditions and the corresponding decoding accuracy was analyzed. Besides analyzing the performance when the entire dataset was used for training, we analyzed the performance of reduced datasets obtained in two different ways, by dropping cells or by dropping training trials. This was accomplished to understand whether the CNN abilities of reach-to-grasp decoding from V6A still persist when the dataset is artificially reduced, simulating scenarios where less cells or training trials are available. Thus, these analyses may serve to further validate the proposed CNN-based framework as a decoding and analysis tool of single-neuron time series, by artificially generating variable-sized datasets. For each monkey and each fold, the training was performed in the following conditions:

a) *No dropping*. The CNN was trained using all the recorded cells and training trials. Thus, only one CNN training was performed for each fold and monkey.
b) *Cell dropping*. The CNN was trained using a subset of $N' \in \{10, 20, 30, 40, 50, 60, 70\}$ cells randomly sampled (10 times) from the entire population. That is, instead of using an input feature map consisting of (N, T_z) spatio-temporal samples, a reduced input feature map of shape $(N' < N, T_z)$ was used. Thus, a total of 10·7 CNN trainings was performed for each fold and each monkey (10 trainings for each of the seven values of N').
c) *Training example dropping*. The CNN was trained using a subset of training examples corresponding to the 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5% randomly sampled (10 times) from the entire training set. Thus, in this case too, a total of 10·7 CNN trainings was performed for each fold and each monkey (10 trainings for each of the seven percentages).

For each CNN training in the previous conditions, the trained CNN was tested within each fold and monkey, computing the accuracy chunk by chunk. Note that in this way, for each fold, we obtained a temporal pattern of decoding accuracy thanks to the sliding window decoding approach, that enabled to highlight the dynamics of the reach-to-grasp task encoded in V6A with a high time resolution (5 ms). Furthermore, in condition b) and c), the accuracy was averaged, chunk by chunk, across the 10 random extractions for each dropping value. Therefore, one temporal pattern of accuracy per fold and monkey was obtained in the condition a), while 7 averaged temporal patterns of accuracy were obtained per fold and monkey in conditions b) and c), each pattern corresponding to a different dropping value.

In each condition, the performance of the proposed approach was compared with the one obtained with a NB classifier (the same used in [2–4]). This decoder takes as input the multi-variate neural activity and it linearly combines inputs assuming independences between time samples. Permutation cluster tests (1000 iterations) with threshold-free cluster enhancement (TFCE) [24] were performed to test for differences between

the accuracy temporal dynamics obtained with the CNN and with NB for each analyzed condition (conditions a-c).

3 Results

Fig. 3 reports the hyper-parameter distributions resulting from BO. The most frequent configuration was a shallow CNN with one convolutional layer ($N_b = 1$, $N_c = 1$), learning the maximum number of allowed feature maps, i.e., kernels ($K = 32$) thus resulting in a shallow but wide network, with a kernel size of $(N, 21)$, corresponding to learning temporal features within approximately 100 ms of the input multi-variate neural activity. Furthermore, BO selected more frequently dropout ($p_{drop} = 0.5$) as regularizer, instead of batch normalization.

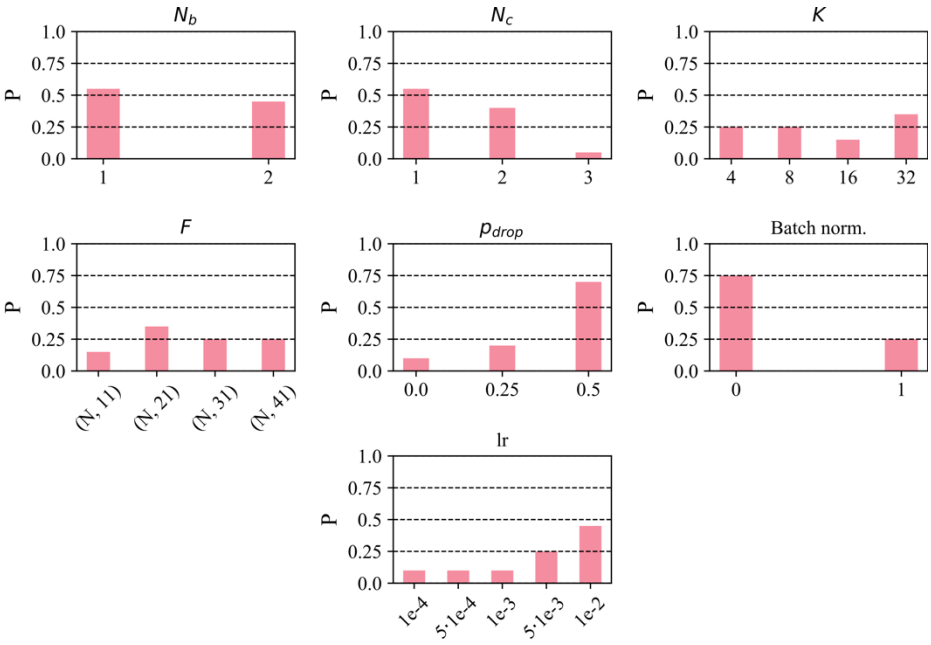

Fig. 3. Hyper-parameter probability distributions. Each bar plot shows the frequency of occurrence of a specific hyper-parameter value among the admitted ones (see Table 1) across the 20 BO configurations (one per fold and monkey).

Fig. 4 reports the dynamic of the prediction (panel A) and of the accuracy (panel B) over the trial course of the proposed Bayesian-optimized CNN, when no dropping was applied (see Section 2.4). Specifically, Fig. 4A displays the output probabilities for the correct class, separately for each object to be reached and grasped, while Fig. 4B reports the decoding accuracy compared with the NB algorithm inspired from [2–4]. The CNN significantly outperformed ($p < 0.05$) the NB algorithm, across the entire time course, already from the last portion of the early delay epoch (epoch 1) up to the hold epoch

(epoch 5). Notably, in Fig. 4B the accuracy is reported also in the free interval (epoch 0), in which the animal was not engaged in the motor task. As expected, here both algorithms performed at the chance level (20%), thus, algorithms proved to be able also to detect the absence of motor-related signatures from the neural activity.

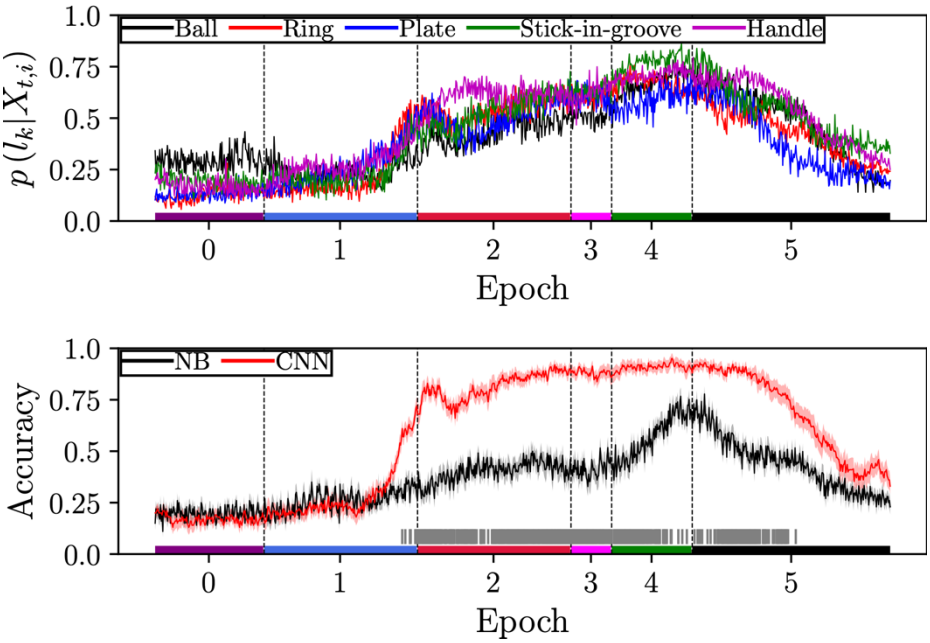

Fig. 4. Panel A - Output probabilities associated to the correct labels for the Bayesian-optimized CNN, separately for the 5 object to reach and grasp (associated to 5 different grip types). Probabilities were averaged across monkeys and cross-validation folds. Panel B - Decoding performance of the proposed Bayesian-optimized CNN (red) and of the NB algorithm (black). Decoding accuracies are reported in their mean value (thick lines) and standard error of the mean (overlayed area) across monkeys and cross-validation folds. The epochs outlining the time sequence of the task are color-coded as: purple-free; blue-early delay; red: late delay; magenta: reaction time; green: reach-to-grasp; black: hold. The vertical dashed lines denote the separation between the epochs. Grey vertical strips reported on the bottom denote time intervals where the two decoding algorithms are significantly ($p < 0.05$) different, as resulting from the performed permutation cluster test.

Finally, Fig. 5 reports the time course of the decoding accuracy scored with NB and the Bayesian-optimized CNN when reduced-size datasets were simulated by sampling a variable number of cells (Fig. 5A) and training examples (Fig. 5B) from the entire dataset (see Section 2.4). Some considerations can be drawn. First, the proposed CNN significantly outperformed the NB baseline (even up to 50% of accuracy) across the entire ranges of cell and training example dropped out, especially in the second part of the delay (epoch 2 and last portion of epoch 1), during the reaction time and the first part of reach-to-grasp and hold epochs. Second, the CNN accuracy was more susceptible to deterioration when few cells were available, in particular below 40 cells, while a

reduction in the number of training examples affected less the CNN performance, as only the lowest percentage of training examples produced an evident performance decrease.

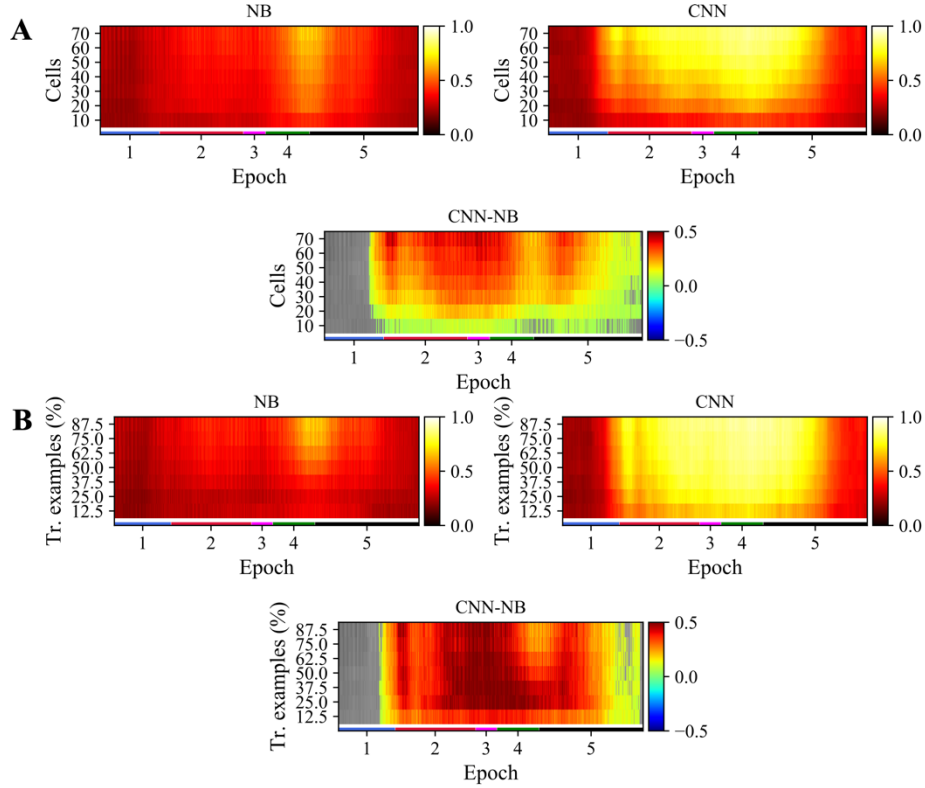

Fig. 5. Time course of the decoding accuracy scored by NB and Bayesian-optimized CNN while randomly dropping out cells (Fig. 5A) and training trials (Fig. 5B) from the dataset. In both Fig. 5A and 5B, the decoding accuracy of each algorithm (NB, CNN) was averaged across folds and monkeys for each dropping value and displayed as an heatmap (upper panels). In addition, the difference between the average accuracy scored with the Bayesian-optimized CNN and with NB is reported in the bottom panel (CNN-NB), where the significant ($p < 0.05$) differences resulting from the performed permutation cluster test were colored (leaving in grey the unsignificant ones). The epochs outlining the time sequence of the task are color-coded as: blue-early delay; red: late delay; magenta: reaction time; green: reach-to-grasp; black: hold.

4 Discussion and conclusions

In this study, a Bayesian-optimized CNN was designed and applied to decode reach-to-grasp from the activity of neurons in V6A, a pivotal parietal area of the dorsomedial visual stream of macaque brain. The decoding capabilities of the learning system were analyzed under different conditions, i.e., while using the entire datasets available and while using reduced datasets, both in terms of number of cells and of training trials.

By using a similar methodology as the one adopted in [2] (similar CNN structure coupled with BO), the present study serves to further validate a CNN-based decoding workflow for general motor decoding, instead of proposing novel CNN workflows, often tailored to one specific application. Notably, the optimal CNN configuration for decoding different grip types obtained here had a similar topology as the optimal CNN configuration obtained in Filippini et al. [2] for decoding the location of reaching targets from PPC neurons. This suggests that a model based on a single-layer CNN performing mixed spatio-temporal convolutions may have enough capacity for general upper-limb motor decoding, decoding both reaching endpoints in space (as obtained in [2]) and grip types (as obtained here).

Despite the adopted sliding approach with short 300 ms windows, CNN predictions resulted consistent through the trial course, especially from the late delay epoch, as shown in Fig. 4A. The CNN significantly outperformed NB, thus, was able to capture more relevant features to discriminate between different grip types compared to NB, over the entire task course and across the different analyses conducted (Fig. 4B and Fig. 5). It is important and fair to note that a previous study [3], using a NB classifier with a sliding window decoding approach on the same dataset used here, scored accuracies up to and above 80% over time. While that study was of great importance to evidence the possibility of decoding grasping information from V6A neurons, the procedure adopted in [3] involves profound differences compared to the one adopted here, where NB exhibited lower performance. Indeed, in the previous study [3], only a subset of cells of the entire dataset (79 cells across the two monkeys), previously identified as being grip-modulated via ANOVA, were used for decoding. In addition, NB was separately trained and tested for each step of the sliding window approach (i.e., $\forall i$, see Eq. 1 and Eq. 2), thus generating different decoders over time, i.e., a different decoder for each analyzed 300 ms-window (overall, M decoders for each cross-validation fold, see Section 2.1). Lastly, in [3] the sliding window decoding was designed with a step of 20 ms and the dataset was binned at 20 ms. Here, all the cells of the dataset were used, without any a priori selection, thus, leaving the learning system the ability to explore all the available information for decoding and to learn to discard task-unrelated information. Furthermore, in this study, the decoder was trained and tested considering all chunks at once, a procedure more parsimonious in terms of number of trained decoders (one decoder per fold, instead of M decoders per fold), but more challenging, as being time-aspecific. In addition, here, signals were binned within a shorter window (5 ms vs 20 ms), enabling to take advantage also of high frequency components (e.g., high gamma band) for decoding, which may play a key role during movement [25]. Furthermore, the decoding and analysis of reach-to-grasp was performed with a finer time resolution of 5 ms. All these points represent more challenging conditions for the neural decoder and the CNN proved to outperform NB with these settings.

Besides the performance difference between the CNN and NB decoders, they showed different time patterns of decoding accuracy (Fig. 4B). NB exhibited only a small increase in performance from early to late delay epoch, increasing rapidly only during the reach-to-grasp epoch, and peaking at the end of movement. Conversely, the CNN showed a strong increase in performance already from the last portion of early delay, then slowly increased over time and peaked between the reach-to-grasp and hold

epochs as NB. That is, the CNN was able to capture grip-discriminant features already while the animal was attending the object to grasp and waiting for the go-signal, further supporting that grip-related neural signatures are progressively encoded in V6A during movement preparation, as known in the literature [3], reflecting a visuomotor processing of the information. These differences between the CNN and NB may be associated to the ability of the CNN to better capture temporal dynamics and non-linearities encoded in neural activity [2], better extracting relevant features from the input signal and discarding task-unrelated information. Lastly, as expected, accuracies gradually deteriorated as the number of cells and training example was artificially reduced. However, the CNN exhibited good robustness, as its accuracy markedly deteriorated only in an extreme low-data regime, suggesting its usefulness also when less cells are recorded or few training examples are available. This result support the potentialities of CNNs as decoders in BCIs, as while transposing decoders from the monkeys to humans, performing a BCI calibration as short as possible and designing a decoder robust to signal degradation over time (thus, reducing the number of recorded cells) are desired aspects.

In conclusion, this study further validates the design and application of CNNs for motor decoding from neurons' recordings. A single-layer wide CNN resulted the optimal design for reach-to-grasp decoding, matching the findings of [2] where a similar structure resulted optimal for reach decoding, and was able to accurately decode grip type early from the start of the task, i.e., from the beginning of movement preparation. However, the present study is affected by the following limitations that will be addressed in the future. First, a more complete performance comparison with also other deep neural networks is needed. Second, selecting a model design by taking the most frequent optimal value for each hyper-parameter may have neglected potential correlations between hyper-parameters. Lastly, the comparison with traditional decoders (e.g., NB) should be further validated as a function of the selection of modulated neurons performed before decoding (as adopted in [3]). Despite the previous limitations, the results obtained in this study, together with the ones obtained in [2], may have some relevant perspectives. Indeed, CNNs compared to other deep learning approaches are more easily interpretable in their learned features. Thus, future studies, by using explanation techniques (e.g., saliency maps [26]) to interpret the features, can exploit CNNs to analyze neural signatures in spatial, temporal and frequency domains [6, 8], identifying differences in neural motor encoding not only across time samples but also, for example, across subpopulations of neurons (at different spatial locations) within area V6A. Finally, the proposed CNN resulted an accurate and light non-linear decoder that, in prospective, may find applicability in BCIs.

Fundings

This study was supported by PRIN 2017 – Prot. 2017KZNZLN and MAIA project. MAIA project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 951910. This article reflects only the author's view, and the Agency is not responsible for any use that may be made of the information it contains.

References

1. Wolpaw, J., Wolpaw, E.W.: Brain-Computer Interfaces: Principles and Practice. Oxford University Press, USA (2012).
2. Filippini, M., Borra, D., Ursino, M., Magosso, E., Fattori, P.: Decoding sensorimotor information from superior parietal lobule of macaque via Convolutional Neural Networks. Neural Networks. 151, 276–294 (2022). https://doi.org/10.1016/j.neunet.2022.03.044.
3. Filippini, M., Breveglieri, R., Akhras, M.A., Bosco, A., Chinellato, E., Fattori, P.: Decoding Information for Grasping from the Macaque Dorsomedial Visual Stream. J. Neurosci. 37, 4311–4322 (2017). https://doi.org/10.1523/JNEUROSCI.3077-16.2017.
4. Filippini, M., Breveglieri, R., Hadjidimitrakis, K., Bosco, A., Fattori, P.: Prediction of Reach Goals in Depth and Direction from the Parietal Cortex. Cell Reports. 23, 725–732 (2018). https://doi.org/10.1016/j.celrep.2018.03.090.
5. Solon, A.J., Lawhern, V.J., Touryan, J., McDaniel, J.R., Ries, A.J., Gordon, S.M.: Decoding P300 Variability Using Convolutional Neural Networks. Front. Hum. Neurosci. 13, 201 (2019). https://doi.org/10.3389/fnhum.2019.00201.
6. Borra, D., Fantozzi, S., Magosso, E.: Interpretable and lightweight convolutional neural network for EEG decoding: Application to movement execution and imagination. Neural Networks. 129, 55–74 (2020). https://doi.org/10.1016/j.neunet.2020.05.032.
7. Borra, D., Fantozzi, S., Magosso, E.: A Lightweight Multi-Scale Convolutional Neural Network for P300 Decoding: Analysis of Training Strategies and Uncovering of Network Decision. Frontiers in Human Neuroscience. 15, 655840 (2021). https://doi.org/10.3389/fnhum.2021.655840.
8. Borra, D., Magosso, E.: Deep learning-based EEG analysis: investigating P3 ERP components. Journal of Integrative Neuroscience. 20, 791–811 (2021). https://doi.org/10.31083/j.jin2004083.
9. Livezey, J.A., Glaser, J.I.: Deep learning approaches for neural decoding across architectures and recording modalities. Briefings in Bioinformatics. 22, 1577–1591 (2021). https://doi.org/10.1093/bib/bbaa355.
10. Craik, A., He, Y., Contreras-Vidal, J.L.: Deep learning for electroencephalogram (EEG) classification tasks: a review. J. Neural Eng. 16, 031001 (2019). https://doi.org/10.1088/1741-2552/ab0ab5.
11. Suhaimi, N.S., Mountstephens, J., Teo, J.: EEG-Based Emotion Recognition: A State-of-the-Art Review of Current Trends and Opportunities. Computational Intelligence and Neuroscience. 2020, 1–19 (2020). https://doi.org/10.1155/2020/8875426.
12. Simões, M., Borra, D., Santamaría-Vázquez, E., GBT-UPM, Bittencourt-Villalpando, M., Krzemiński, D., Miladinović, A., Neural_Engineering_Group, Schmid, T., Zhao, H., Amaral, C., Direito, B., Henriques, J., Carvalho, P., Castelo-Branco, M.: BCIAUT-P300: A Multi-Session and Multi-Subject Benchmark Dataset on Autism for P300-Based Brain-Computer-Interfaces. Front. Neurosci. 14, 568104 (2020). https://doi.org/10.3389/fnins.2020.568104.

13. Borra, D., Magosso, E., Castelo-Branco, M., Simoes, M.: A Bayesian-optimized design for an interpretable convolutional neural network to decode and analyze the P300 response in autism. J. Neural Eng. 19, (2022). https://doi.org/10.1088/1741-2552/ac7908.
14. Schirrmeister, R.T., Springenberg, J.T., Fiederer, L.D.J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W., Ball, T.: Deep learning with convolutional neural networks for EEG decoding and visualization. Human brain mapping. 38, 5391–5420 (2017).
15. Mulliken, G.H., Musallam, S., Andersen, R.A.: Decoding Trajectories from Posterior Parietal Cortex Ensembles. Journal of Neuroscience. 28, 12913–12926 (2008). https://doi.org/10.1523/JNEUROSCI.1463-08.2008.
16. Aflalo, T., Kellis, S., Klaes, C., Lee, B., Shi, Y., Pejsa, K., Shanfield, K., Hayes-Jackson, S., Aisen, M., Heck, C., Liu, C., Andersen, R.A.: Decoding motor imagery from the posterior parietal cortex of a tetraplegic human. Science. 348, 906–910 (2015). https://doi.org/10.1126/science.aaa5417.
17. Chinellato, E., Grzyb, B.J., Marzocchi, N., Bosco, A., Fattori, P., del Pobil, A.P.: The Dorso-medial visual stream: From neural activation to sensorimotor interaction. Neurocomputing. 74, 1203–1212 (2011). https://doi.org/10.1016/j.neucom.2010.07.029.
18. Fattori, P., Breveglieri, R., Bosco, A., Gamberini, M., Galletti, C.: Vision for Prehension in the Medial Parietal Cortex. Cereb. Cortex. bhv302 (2015). https://doi.org/10.1093/cercor/bhv302.
19. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: Bach, F. and Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. pp. 448–456. PMLR, Lille, France (2015).
20. Clevert, D.-A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint. (2015).
21. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research. 15, 1929–1958 (2014).
22. Frazier, P.I.: A Tutorial on Bayesian Optimization, http://arxiv.org/abs/1807.02811, (2018).
23. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs]. (2017).
24. Smith, S., Nichols, T.: Threshold-free cluster enhancement: Addressing problems of smoothing, threshold dependence and localisation in cluster inference. NeuroImage. 44, 83–98 (2009). https://doi.org/10.1016/j.neuroimage.2008.03.061.
25. Nowak, M., Zich, C., Stagg, C.J.: Motor Cortical Gamma Oscillations: What Have We Learnt and Where Are We Headed? Curr Behav Neurosci Rep. 5, 136–142 (2018). https://doi.org/10.1007/s40473-018-0151-z.
26. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034 [cs]. (2014).