Motor decoding from the posterior parietal cortex using deep neural networks

Journal of Neural Engineering

PAPER

Motor decoding from the posterior parietal cortex using deep neural networks

Davide Borra[1,*], **Matteo Filippini**[3], **Mauro Ursino**[1,2], **Patrizia Fattori**[2,3,4] and **Elisa Magosso**[1,2,4,*]

[1] Department of Electrical, Electronic and Information Engineering 'Guglielmo Marconi' (DEI), University of Bologna, Cesena Campus, Cesena, Italy
[2] Alma Mater Research Institute for Human-Centered Artificial Intelligence, University of Bologna, Bologna, Italy
[3] Department of Biomedical and Neuromotor Sciences (DIBINEM), University of Bologna, Bologna, Italy
[4] Authors with shared senior authorship.
[*] Authors to whom any correspondence should be addressed.

E-mail: davide.borra2@unibo.it and elisa.magosso@unibo.it

Keywords: motor decoding, deep learning, macaque, single-neuron recordings, brain–computer interfaces (BCIs)

Supplementary material for this article is available online

OPEN ACCESS

RECEIVED
4 April 2023

ACCEPTED FOR PUBLICATION
2 May 2023

PUBLISHED
19 May 2023

Abstract

Objective. Motor decoding is crucial to translate the neural activity for brain-computer interfaces (BCIs) and provides information on how motor states are encoded in the brain. Deep neural networks (DNNs) are emerging as promising neural decoders. Nevertheless, it is still unclear how different DNNs perform in different motor decoding problems and scenarios, and which network could be a good candidate for invasive BCIs. *Approach.* Fully-connected, convolutional, and recurrent neural networks (FCNNs, CNNs, RNNs) were designed and applied to decode motor states from neurons recorded from V6A area in the posterior parietal cortex (PPC) of macaques. Three motor tasks were considered, involving reaching and reach-to-grasping (the latter under two illumination conditions). DNNs decoded nine reaching endpoints in 3D space or five grip types using a sliding window approach within the trial course. To evaluate decoders simulating a broad variety of scenarios, the performance was also analyzed while artificially reducing the number of recorded neurons and trials, and while performing transfer learning from one task to another. Finally, the accuracy time course was used to analyze V6A motor encoding. *Main results.* DNNs outperformed a classic Naïve Bayes classifier, and CNNs additionally outperformed XGBoost and Support Vector Machine classifiers across the motor decoding problems. CNNs resulted the top-performing DNNs when using less neurons and trials, and task-to-task transfer learning improved performance especially in the low data regime. Lastly, V6A neurons encoded reaching and reach-to-grasping properties even from action planning, with the encoding of grip properties occurring later, closer to movement execution, and appearing weaker in darkness. *Significance.* Results suggest that CNNs are effective candidates to realize neural decoders for invasive BCIs in humans from PPC recordings also reducing BCI calibration times (transfer learning), and that a CNN-based data-driven analysis may provide insights about the encoding properties and the functional roles of brain regions.

1. Introduction

Motor decoding from neural signals consists in finding the relationship that maps a multivariate neural time series to motor outputs. Depending on the nature of motor outputs to be decoded, e.g. reach goal positions in 3D space or hand gestures, the neural decoder searches for a particular input-output relationship. This represents a crucial stage for brain-computer interfaces (BCIs), which translate the neural activity into commands to drive external devices, e.g. for assistive or therapeutic purposes [1, 2]. Crucially, accurate predictions with a proper timing are desirable, to correctly translate the neural activity while guaranteeing contingency between the recorded neural activity and the control

of the external device/feedback provided to the user [3]. Moreover, in addition to practical implications for BCIs, neural decoders may advance our understanding of how the information is encoded in the brain: indeed, performance metrics of neural decoding unveil the amount of information the neural signals contain about the decoded motor outputs, and this can be evaluated across different brain regions and/or across different time intervals over the course of the sensorimotor task.

Voluntary movement has been decoded in human from invasively recorded signals (in patients), such as electrocorticographic [4–6] or intracortical [7–10] recordings, and from non-invasively recorded signals (in both healthy subjects and patients), such as magnetoencephalographic [11–14] and electroencephalographic (EEG) [11, 15–24] recordings. Specifically, simpler motor outputs, such as movements performed with different limbs, and more challenging motor outputs, such as fine movement trajectories or positions reached in the space with the same limb, were decoded. Due to the increased signal-to-noise ratio [25], neural decoding from invasive recordings is in general more accurate and might prevail in the future in most BCI applications, e.g. to restore motor functions [26]. Crucially, the research on invasive motor decoding takes advantage from comparative analyses on invasive procedures in animal models, such as non-human primates (NHP), to better comprehend the neural activity at the cellular level and, thus, better address the decoding of invasive recordings in humans.

Among the brain areas encoding reaching and grasping properties, the posterior parietal cortex (PPC) is a crucial node and has received much attention in the recent neurophysiological literature. Specifically, PPC in humans and NHPs hosts areas involved in the sensorimotor processing required to generate action plans [27–30]. The neural activity recorded from PPC areas was used to decode reach endpoints and trajectories, as well as grasping parameters, both in NHPs [31–34] and human patients [10, 35]. Among PPC areas, the V6A area (in the dorsomedial stream) encodes reaching goals and directions [36–39] in addition to grasp information [40], and it was recently proved that reaching and grasping properties can be reliably decoded from V6A in NHPs [41–44].

Machine learning approaches are widely used for neural decoding [45], based on a pipeline that processes the input neural activity via separate stages including feature extraction, feature selection and classification/regression. Deep neural networks (DNNs) are recent advances of machine learning that realize an end-to-end learning framework. That is, the classic machine learning pipeline, implementing separate stages, is replaced with a single learning system that learns to directly map the input neural activity to the desired output. Crucially, the DNN-based learning system automatically learns the most relevant neural features to realize the desired input-output mapping. Over the past decade, DNNs were successfully designed and applied for neural decoding [46], performing on par or even outperforming state-of-the-art machine learning approaches [16, 44–51]. Furthermore, DNNs facilitate the adoption of transfer learning. Transfer learning is inspired by the human capability to use the knowledge learned in a source domain/task to improve the performance and/or reduce learning time in a related target domain/task [52]. This technique is growing interest in neural time series decoding (e.g. see [49, 53, 54] in case of EEG decoding, where a user-to-user transfer is adopted), mainly with the aim of increasing the decoding performance with few training examples, in the perspective of reducing calibration time in BCI applications.

DNNs are composed by many layers of artificial neurons. By learning simple non-linear functions within each layer and by composing these functions, DNNs learn to approximate a complex and non-linear function which maps the input multivariate neural activity to the proper output. Depending on the connections established across artificial neurons, recurrent neural networks (RNNs), feed-forward and fully-connected neural networks (FCNNs), and feed-forward and convolutional neural networks (CNNs) can be designed. The weights defining the connections across artificial neurons are collected into a set of trainable parameters that must be optimized during a training process. In addition, the parameters defining the functional form of the decoder (e.g. the number of artificial neurons per layer, the numbers of layers, etc.), also called 'hyper-parameters', need to be set before the network training starts and should be optimized using hyper-parameter tuning algorithms, such as Bayesian optimization [55], as suggested by Glaser *et al* [45] for neural decoding. In particular, the interest on CNNs has rapidly increased in recent years, especially in EEG decoding in humans [47], and these networks represent the most frequently adopted DNNs for EEG, mainly for the following reasons: (i) they are lighter, that is, they introduce less trainable parameters to fit, proving to outperform significantly other DNNs [51] also in case of limited-sized datasets; (ii) they are faster to train, thus, they are particularly suitable to be used as decoders in BCI practical scenarios [49, 53]. Furthermore, the interest in CNNs has recently extended also to other recording modalities as well, e.g. the CNNs proposed to process EEG in humans have been modified and adapted to decode magnetoencephalography [56] and electrocorticography [57]. Conversely, decoding of invasive neural recordings from NHPs, particularly

motor decoding, widely relies on RNNs, based on long-short-term memories (LSTMs) or gated-recurrent units (GRUs), and on FCNNs, achieving significant performance improvements over other machine learning approaches, such as XGBoost, support vector machine (SVM), Kalman and Wiener filter, and Naïve Bayes (NB) classifiers [45, 46]. Only recently we investigated the design and application of CNNs for decoding the activity of single neurons in NHP brain [44]. Specifically, that study [44] was focused on motor decoding of reaching from PPC recordings (including V6A recordings); however, CNNs were only compared with a machine learning approach based on a NB classifier, without evaluating other DNNs and other successful traditional machine learning approaches (e.g. XGBoost and SVM), and addressed only the decoding of reaching endpoints. It remains still unclear how different DNNs perform in decoding motor states from single-neuron activity, also considering different types of motor outputs to be decoded, corresponding to different recording paradigms, e.g. involving reaching movements only or reach-to grasp movements.

In this study, we aim at contributing to single-neuron motor decoding from V6A by evaluating different families of DNNs (FCNNs, CNNs, and RNNs) and by comparing them with traditional machine learning approaches, including the NB classifier that we already adopted in our previous studies [41–44] and other two traditional approaches, XGBoost and SVM, that proved to perform well in Glaser *et al* [45]. All neural decoders were trained using a sliding window approach, useful to reveal the temporal dynamics of motor encoding in V6A. Neural network optimal hyper-parameters were searched by using Bayesian optimization. Analyses were conducted using three different datasets collected from four macaque monkeys: one dataset, collected on two monkeys, refers to a reaching task towards different endpoints in 3D space and the other datasets, collected on the other two monkeys, refer to a reach-to-grasping task of different grip shapes performed, by each of the two monkeys, in two different illuminance conditions (light vs. dark). This was performed to provide broad and more robust evaluation of neural decoders using different DNNs across different nature of recording paradigms (i.e. decoding tasks). Furthermore, decoders were evaluated by using a bank of experiments devoted to inspecting decoding capabilities in different conditions, that is, when using the entire dataset, when reduced datasets are simulated, by reducing either the number of recorded cells or recorded trials from the whole datasets, and when applying transfer learning from a task to another (task-to-task transfer learning within the same monkey). These experiments aim at providing a more complete evaluation of decoders, also as a function of recorded cells and trials, and to test the feasibility of transferring the knowledge from neural networks trained in one task to another.

2. Materials and methods

In this section, first, we describe the datasets and formalize sliding window decoding of single-neuron activity, introducing useful notations. Then, we describe the DNNs used for decoding, the hyper-parameter search approach, and the training strategies. Lastly, the performed statistical analyses are illustrated. DNNs were developed in Python (version 3.8.5) using the PyTorch library (version 1.9.0) [58] and the optimizations were performed on a workstation equipped with an AMD Threadripper 1900X, NVIDIA TITAN V and 48 GB of RAM. Codes are available at https://github.com/ddavidebb/macaque-single-neuron-decoding.git. Since many notations and equations will be introduced in the following subsections, all symbols are resumed in table 1 to ease the reading.

2.1. Data description

The learning systems were designed and applied on signals recorded during two experimental paradigms, involving a reaching task and a reach-to-grasping task, respectively. For each experimental condition, single-neuron activities were recorded extracellularly from the posterior parietal area V6A (figure 1(a), for more details on the location and reconstruction of electrode penetration see [59]) in male Macaca fascicularis monkeys. Two monkeys (m1, m2) were recorded in the reaching task and other two monkeys (m3, m4) in the reach-to-grasping task. Overall, from these recordings three datasets were considered in this study (one from the reaching task, two from the reach-to-grasping task), addressing three different decoding problems. These were reach decoding, reach-to-grasp decoding in good illumination condition, and reach-to-grasp decoding in darkness. The study was performed in accordance with the guidelines of EU Directives (86/609/EEC; 2010/63/EU) and Italian national law (D.L. 116-92, D.L. 26-2014) for the care and use of animals for scientific purposes. Experimental protocols have been approved by the Ethical Committee of the University of Bologna and by the Animal Welfare Body of the University of Bologna.

2.1.1. Reaching

Signals recorded in [41] were used. Specifically, the activity of 138 and 120 cells was recorded from the two monkeys (m1 and m2) respectively, while they performed a reaching task (more details about recording procedure can be found in [41]); action

Table 1. Summary of the introduced notations: symbols and descriptions.

Symbol	Description
m1–m4	Monkey identifiers
X_t	Multivariate neural activity recorded from the tth trial
N	No. of recorded cells
T	No. of time steps
$X_{t,i}$	Multivariate neural activity contained in the ith chunk extracted from X_t while performing sliding window decoding
T_z	No. of time steps defining $X_{t,i}$ for sliding window decoding
T_s	No. of time steps used as stride factor for sliding window decoding
M	No. of chunks that can be extracted from X_t using T_z and T_s
y_t	Class associated to the tth trial (X_t)
C	Set of classes
c_k	kth class, contained in C
N_c	No. of decoded classes
$y_{t,i}$	Class associated to the tth trial and ith chunk ($X_{t,i}$)
ϑ	Set of trainable parameters
η	Set of hyper-parameters
$f(X_{t,i};\vartheta,\eta)$	Output of a classifier parametrized in trainable parameters (ϑ) and hyper-parameters (η) when $X_{t,i}$ is provided as input
W	Weights: can have subscript superscript, also depending on the formalization (e.g. W_o weights of the output layer)
b	Biases: can have subscript superscript, also depending on the formalization (e.g. b_o biases of the output layer)
$yp_{t,i}$	Class predicted by the network when $X_{t,i}$ is provided as input
$y^{(l)}$	Output of the lth layer in FCNNs and CNNs
N_u	No. of artificial neurons used in each hidden layer in FCNNs
N_l	No. of hidden layers used in FCNNs and RNNs
p_{drop}	Dropout probability
N_b	No. of convolutional blocks in CNNs
N_{lb}	No. of hidden convolutional layers per block in CNNs
F	Size of convolutional kernels along the time axis
K	No. of convolutional kernels learned in each convolutional layer
$Y^{(l)}$	Multidimensional output of convolutional layers (once flattened, corresponds to $y^{(l)}$)
H	No. of features of the hidden state in a GRU
z_n	Output of the update gate at time step n in a GRU
r_n	Output of the reset gate at time step n in a GRU
σ	Sigmoid function
$\hat{h}_n$	New memory content at time step n in a GRU
$tanh$	Hyperbolic tangent function
$\odot$	Element-wise product
h_n	New state at time step n in a GRU
$k(\eta)$	Objective function used to optimize η
$j(\vartheta)$	Loss function used to optimize ϑ
lr	Learning rate used while optimizing the loss function $j(\vartheta)$

potentials (spikes) were isolated and sampled at 100 KHz.

Monkeys sat on a primate chair with their head restrained in front of a horizontal panel housing nine LEDs located at three different directions and distances with respect to the eyes (but all at the eye level, figure 1(b)). The directions were 0° (sagittal), −15° and +15° (respect the sagittal direction), while the depths were 10, 15, and 25 cm from monkey eyes; LEDs were placed in this way to enable the inclusion of most of the peripersonal space in front of the animals, from the nearest (10 cm) to the farthest (25 cm) depths reachable by monkeys. Animals were trained to perform reaching movements toward one of the nine LEDs at a time in a randomized order, using the arm contralateral to the recorded hemisphere.

For each position to reach, 10 recording trials were recorded, overall resulting in 90 trials for each monkey and neuron (randomized per position to reach). Each trial consisted of different phases (figure 1(c)) hereafter referred as 'epochs'.

Specifically, the trial started when the monkey pressed a 'home button' near to its chest in complete darkness and then the animal waited for instructions for 1 s (*free epoch*, epoch 0). One LED switched on (green) and the monkey started maintaining the fixation on the target LED for 0.5 s (*fixation epoch*), while keeping the home button pressed. Thus, during the

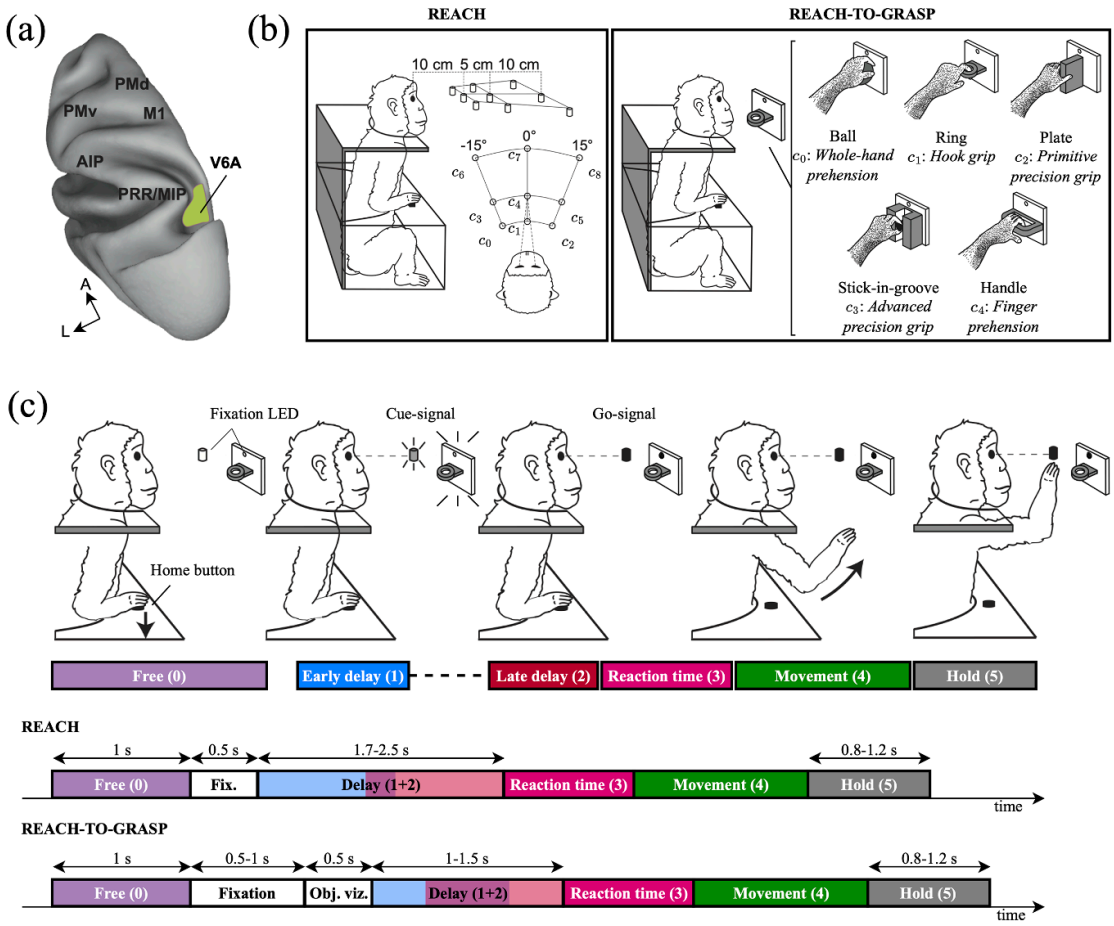

Figure 1. Dataset description. Panel (a)—Dorsal view of the left hemisphere of macaque brain. The green highlighted area represents V6A. The shown directions are A = anterior, L = lateral. Panel (b)—Reach and reach-to-grasp decoded classes. During reach decoding, the reached endpoint (one among 9 targets in space) was classified; during reach-to-grasp decoding, the grip type (one among 5 different grips) was classified. Panel (c)—Trial structure for both the reaching task and reach-to-grasping task. Top: The epochs defining each trial, which have the same definition across recording paradigms, are illustrated for both the reaching task and reach-to-grasping task (displaying both the LED to reach and the panel containing the object to reach and grasp). Bottom: The epoch timings are described, separately for the reaching task and reach-to-grasping task.

0.5 s-fixation epoch the monkey focused attention on the target LED to reach during the task. After 1.7–2.5 s of rest with no arm or eye movements (*delay epoch*), the target LED changed its color (from green to red, go-signal), cueing the animal to start reaching the foveated target LED, releasing the home button. The animal behavior in fixation epoch and delay epoch was the same (fixation of the LED to reach); the separation between these two epochs was kept to uniform reaching and reach-to-grasping tasks to ease the comparison between tasks. The delay epoch was subdivided into two parts: the *early delay epoch* (epoch 1) and the *late delay epoch* (epoch 2), by extracting a 1 s interval after the start of the delay epoch and a 1 s interval before the end of the delay epoch, respectively. Once the delay epoch ends, a *reaction time epoch* (epoch 3, between the onset of the go-signal and the release of the home button) and *movement epoch* (epoch 4, between the release of the home button and the touch of the target LED) can be identified. Once reached the target, the monkey had to hold the target LED for 0.8–1.2 s (*hold epoch*, epoch 5) and finally, the LED switched off, cueing the animal to return to the home button and ending the trial.

2.1.2. Reach-to-grasping

Signals recorded in [42] were used. Specifically, the activity of 93 and 75 cells was recorded from the two monkeys (m3 and m4) respectively, while monkeys performed a reach-to-grasping task (more details about recording procedure can be found in [42]); action potentials (spikes) were isolated and sampled at 100 kHz.

Monkeys sat on a primate chair with their head restrained in front of a rotating panel containing five different objects (figure 1(b)). The objects were chosen to evoke reach-to-grasp with different hand configurations. These were: a ball (c_0: whole-hand prehension), ring (c_1: hook), plate (c_2: primitive precision grip), stick-in-groove (c_3: advanced precision grip), handle (c_4: finger prehension). Objects were presented to the monkey one at a time, in

a randomized order. Animals were trained to perform reach-to-grasping movements toward the target object with the arm contralateral to the recorded hemisphere.

Each animal performed two sessions, differing for the illumination (present or absent) of the object to grasp (see below). In each session, for each object to grasp, 10 recording trials were recorded, overall resulting in 50 trials for each monkey and neuron (randomized per object to be reached and grasped). In the following, the different phases ('epochs') of the trial are descripted and are illustrated in figure 1(c).

The trial started when the monkey pressed a 'home button' near to its chest in complete darkness and then, the animal waited for instructions for 1 s (*free epoch*, epoch 0). Subsequently, the fixation LED mounted on top of the rotating panel turned on (green), and the monkey had to maintain the fixation on the LED without performing any movement. After a fixation period of 0.5–1.0 s (*fixation epoch*), the LEDs surrounding the object to grasp turned on, illuminating the object to reach and grasp for 0.5 s (*object visualization epoch*). Thus, during the 0.5 s-object visualization epoch the monkey focused attention on the specific shape to grasp during the task. Then, depending on the experimental condition, all the following trial phases were performed with (light condition) or without (dark condition) illumination around the object to grasp (i.e. in dark condition the LEDs surrounding the object turned off). These two experimental conditions were performed to also consider the effect of object visual information (present vs. absent) while performing the reach-to-grasp task. The monkey maintained the fixation on the LED without releasing the home button for a period of 1–1.5 s (*delay epoch*). This epoch was subdivided into the *early delay epoch* (epoch 1) and *late delay epoch* (epoch 2), as done in the reaching task. Then, the fixation LED changed its color (from green to red), representing the go-signal for the reach-to-grasp movement, and *reaction time epoch* and *movement epoch* can be identified (epochs 3,4, respectively). Once performed the movement, the monkey had to keep holding (*hold epoch*, epoch 5) the grasped object until the fixation 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.

Reach-to-grasping performed with or without illumination around the object to grasp were separately decoded in this study and are referred in the following as 'light' and 'dark' reach-to-grasp datasets, to also investigate modulations in decoding performance due to the presence or absence of object visual information immediately before (delay epochs) and during the reach-to-grasp movement.

2.1.3. Pre-processing and data splitting

All datasets were pre-processed as follows. For each trial and neuron, spikes were initially binned within a window of 5 ms (i.e. bin width of 5 ms). Generally, the bin width is chosen as a trade-off between noise reduction (the discharge activity of a biological network is subject to noise by definition), exploiting wide bin widths, and temporal resolution (depending on the speed of the neural processes under investigation), exploiting narrower bin widths. As we were interested into leaving decoders free to learn also motor features related to high-frequency dynamics [60, 61] which can help neural decoding of motor states [21], in our study the bin width was set to 5 ms. As epochs may have a different duration across trials and neurons (e.g. the movement epoch), epochs could differ in the number of resulting bins. To obtain the same number of bins across trials and neurons, for each epoch the average number of bins across trials and neurons was computed. Then, the activity of each trial and neuron was re-binned using a window that produces the so obtained average number of bins per epoch (thus, the window may slightly differ from the initial 5 ms-window). This procedure is the same adopted in [44]. Firing rates were then computed on the re-binned activity; thus, in this study the multivariate neural activity was described by means of neurons' firing rates. Firing rates recorded during the free, early delay, late delay, reaction time, movement (reach or reach-to-grasp), and hold epochs were collected and used in this study. Ten-fold cross-validation was applied to partition the dataset of each monkey into training and test set; then, a 10% of examples from the training set was held back as validation set. Therefore, in case of the reaching dataset, for each monkey and each fold, the data splits resulted: 9 trials (one per target position) in the test set, 9 trials (one per target position) in the validation set, and the remaining 72 trials (8 trials per target position) in the training set. In case of the reach-to-grasp datasets (both light and dark), for each monkey and each fold, the data splits resulted: 5 trials (one per grip type) in the test set, 5 trials (one per grip type) in the validation set, and the remaining 40 trials (8 trials per grip type) in the training set. Note that, the free epoch was included only in the trials belonging to the test set (i.e. epochs from 0 to 5 for testing, and epochs from 1 to 5 for training and validation), i.e. algorithms were only trained on the trial epochs in which the animal was engaged in the task. It is worth noticing that, in this study we did not apply a trial-level decoding, but we applied a sliding window decoding approach, which inherently implies data augmentation, thus overcoming the problem of

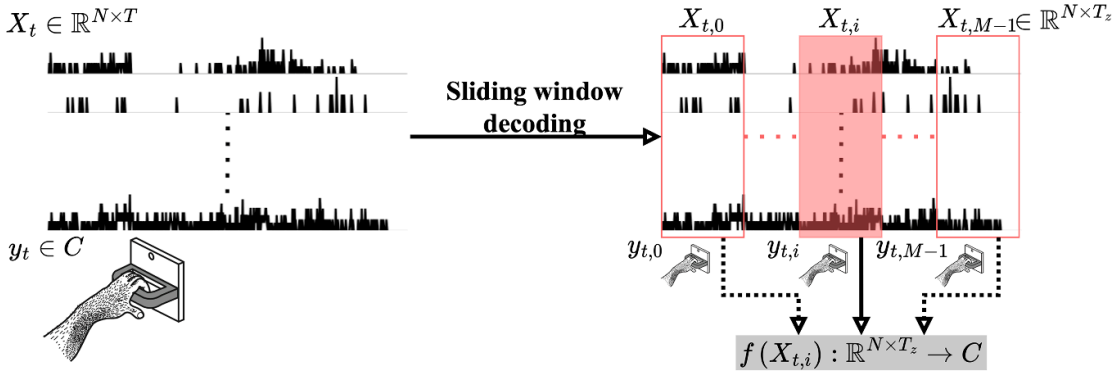

Figure 2. Sliding window decoding. Overlapped chunks $X_{t,i}$ of shape (N, T_z) were extracted from the multivariate neural activity X_t using a stride of T_s time samples, where T_z is the number of time samples of each chunk. Each trial was associated with a motor class $y_t \in C$, and all chunks extracted from that trial were associated to the motor class the trial belonged to, i.e. $y_{t,i} = y_t, \forall i$. Each sampled chunk $(X_{t,i})$ represented the input for the neural decoder, while its associated class $y_{t,i}$ represented the desired output for the neural decoder. Therefore, the neural decoder was trained to learn how to map each chunk $X_{t,i}$ to the proper motor class $y_{t,i}$, via the function $f(X_{t,i}; \vartheta, \eta) : \mathbb{R}^{N\times T_z} \rightarrow C$, parametrized in the trainable parameters (ϑ) and hyper-parameters (η).

datasets having a low number of trials (see section 2.2 below).

2.2. Sliding window neural decoding

Neural decoding was addressed adopting a sliding window approach. This enables to realize learning systems that decode small portions of neural signals within the trial course (i.e. 'chunks' of neural activity comprising a few hundreds of ms, chunk-level decoding), instead of the entire trial (i.e. trial-level decoding). The sliding window decoding procedure is often adopted in the literature [41–44, 62] since it exhibits three main advantages. First, learning systems are forced to associate the proper label (in case of classification tasks) to a few hundreds of milliseconds of signals instead of to the entire trial (single-trial decoding), thus, providing a faster and earlier inference over the trial course. Second, learning systems may be used to analyze the temporal dynamics of motor encoding (in this case, reach and reach-to-grasp encodings) by using chunk-by-chunk prediction performance as measure of neural encoding over time of the motor-related brain states. Third, compared to trial-level decoding, this decoding procedure is equivalent to augmenting data by applying a slicing technique, which is commonly adopted for time series classification [63] and is the most common data augmentation strategy adopted for neural time series (e.g. EEG [64]). The sliding window approach, applied with the parameters described in the following paragraph, allowed considerable augmentation of examples in the training, validation and test set (see the number of examples in table 2); for instance, the training data were augmented by approximately 75 times, resulting in about 600 training examples per condition (i.e. target position or grip type) on average across reaching and reach-to-grasping tasks, rather than 8 training examples per condition as it would be in case of trial-level decoding.

To perform sliding window neural decoding, neurons' firing rates were processed as follows, for each monkey (see figure 2).

Let X_t be the multivariate neural activity in the tth trial, having shape (N, T), where N is the number of recorded neurons ($N = 138, 120, 93, 75$, in m1-m4, respectively) and T is the number of time samples in the trial. 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, and were fed as input to the neural decoder:

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

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 this decoding approach and were set as in [44]. We set $T_z = 60$ (=300 ms) and $T_s = 10$ (=50 ms) during the training phase of the decoder, 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 steps. Furthermore, the inference was performed also on the free epoch (epoch 0), to check for random motor decoding when the monkey was not engaged in the task. Dataset details are summarized in table 2 together with the total number of training, validation, and test examples resulting for each dataset.

The addressed decoding problems were the classification of nine different target positions in space (reaching task) and five different grip types (reach-to-grasping task) from neuron firing rates. Each trial X_t was associated to a single class, corresponding to the target position to reach or the specific shape of the object the monkey had to grasp in that trial, i.e. $y_t \in C = \{c_k\}, 0 \leqslant k \leqslant N_c - 1$, where $N_c = 9$ or $N_c = 5$ is

Table 2. Description of the datasets.

		Reaching (m1/m2)	Reach-to-grasping: light (m3/m4)	Reach-to-grasping: dark (m3/m4)
No. of decoded classes (N_c)		9	5	5
No. of recorded cells (N)		138/120	93/75	93/75
Epochs	Train. and valid.	1–5	1–5	1–5
	Test	0–5	0–5	0–5
No. of time steps (T)	Train. and valid.	780/773	812/816	815/818
	Test	981/974	1013/1017	1016/1019
No. of examples	Train.	5256/5184	3040/3040	3040/3040
	Valid.	657/648	380/380	380/380
	Test	8298/8235	4770/4790	4785/4800

the number of motor classes (see section 2.1). Therefore, while performing sliding window decoding, the class associated to each chunk ($y_{t,i}$) was the one associated to the trial the chunk was extracted from:

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

DNNs were used as classifiers $f(X_{t,i};\vartheta,\eta): \mathbb{R}^{N\times T_z} \rightarrow C$, parametrized in the trainable parameters and hyper-parameters contained in the arrays ϑ, and η, respectively. Specifically, monkey-specific decoders were designed, by using monkey-specific datasets to tune trainable parameters and hyper-parameters of DNNs. Hyper-parameters must be set before the model training starts; these parameters can be optimized on a separate validation set via a hyper-parameter search procedure. Trainable parameters are the collection of weights (W) and biases (b) that model connections across the artificial neurons included in the network (see section 2.3.1); these are learned on the training set during the network training.

2.3. Neural decoders

2.3.1. Deep neural networks

In this study, three families of neural networks were investigated, differing mainly in the type of connections across artificial neurons; these included networks with feed-forward and dense connections (FCNNs), feed-forward and sparse connections (CNNs) and recurrent connections (RNNs). The designed neural networks are schematized in figure 3. In the following, these neural networks, including their main hyper-parameters, will be described. In the proposed formulations, W and b denote weight matrices and bias arrays introduced by a layer, respectively; these may include different subscripts and superscripts in the description of hidden connections depending on the specific formulation, while the connections with the output layer are denoted by W_o, b_o. Lastly, $yp_{t,i}$ denotes the predicted class associated to the ith chunk of the tth trial.

2.3.1.1. FCNN

In this network structure, layers of artificial neurons are stacked realizing all possible connections between the lth and (l-1)th layers. The input layer was composed by $N \cdot T_z$ artificial neurons and simply replicated the flattened (i.e. reshaped to 1-D) input $X_{t,i}$. The network included N_l hidden fully-connected layers, each composed by N_u artificial neurons, in addition to input and output layers. For each hidden layer, activations were normalized via batch normalization (BN) [65], passed through an exponential linear unit (ELU) non-linearity [66], i.e. $f(x) = x,\ x > 0$ and $f(x) = \exp(x) - 1,\ x \leqslant 0$ and dropout [67] was applied with a dropout probability p_{drop}. Lastly, the output layer was a fully-connected layer composed by N_c neurons, activated via softmax non-linearity to provide as output the conditional probabilities $p(c_k|X_{t,i})$. Finally, the predicted class $yp_{t,i}$ was computed as the most probable one among the N_c classes. The FCNN structure is schematized in figure 3(a).

A single forward-pass through this network (during inference, when dropout is not applied) can be formalized as:

$$y^{(0)} = flatten(X_{t,i}) \in \mathbb{R}^{N\cdot T_z} \tag{3}$$

$$y^{(l)} = ELU\left(BN\left(W^{(l)}y^{(l-1)} + b^{(l)}\right)\right),\ 1 \leqslant l \leqslant N_l, \tag{4}$$

having indicated with $y^{(l)}$ the lth layer output.

$$\begin{cases} f(X_{t,i};\vartheta,\eta) = softmax\left(W_o y^{(N_l)} + b_o\right) = p(c_k|X_{t,i}) \\ yp_{t,i} = \underset{k}{\text{argmax}}\left(p(c_k|X_{t,i})\right) \end{cases},\ 0 \leqslant k \leqslant N_c - 1 \tag{5}$$

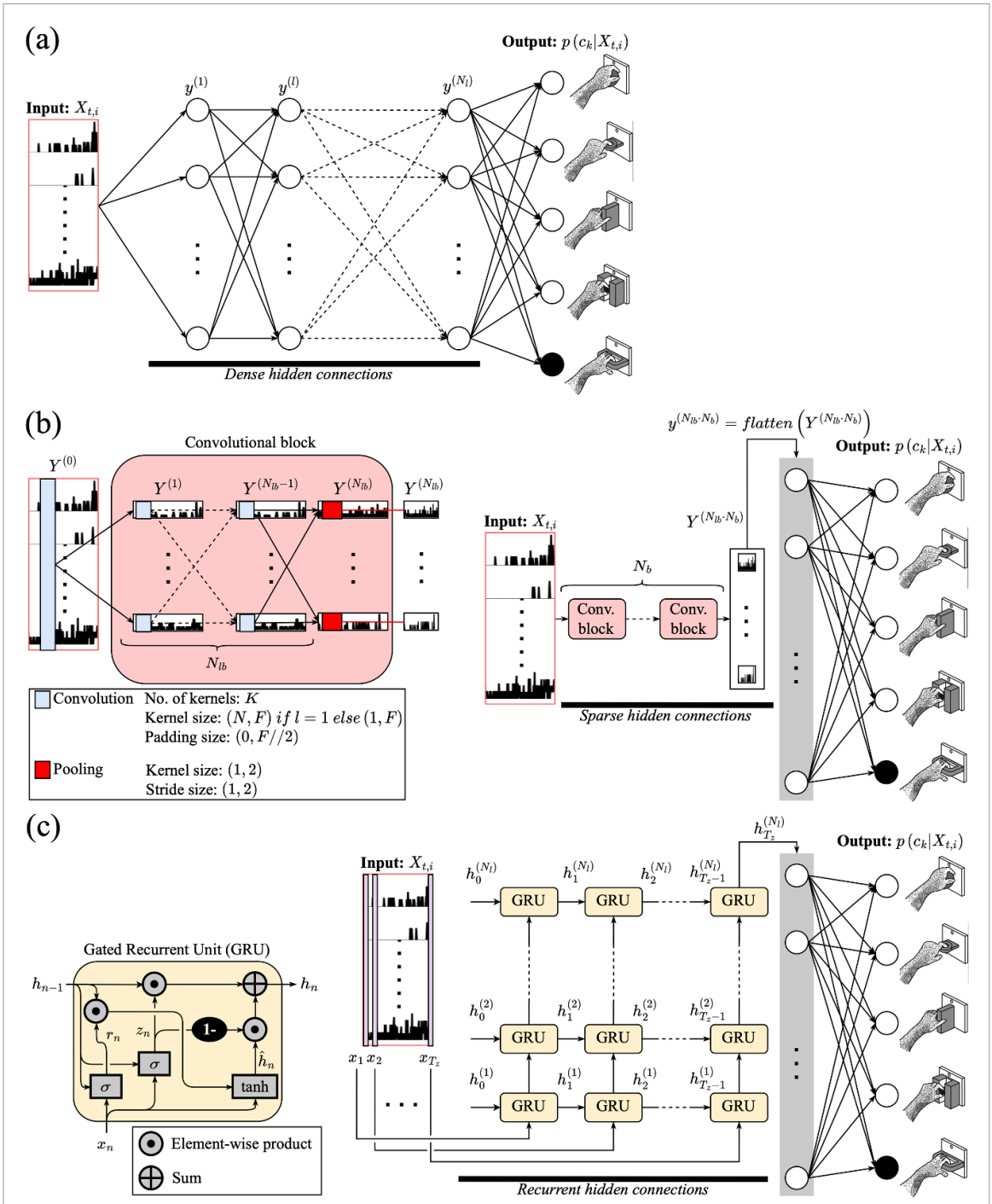

Figure 3. Architecture structures. For brevity, all schematizations refer only to reach-to-grasp decoding (decoding of 5 different grip types). Panel (a)—FCNNs. Panel (b)—CNNs. On the left, the fundamental convolutional block exploited to realize CNNs is schematized visualizing its main layers; here, convolutional, and pooling kernels are represented by blue and red boxes, respectively. Note that, for brevity only the first convolutional block is visualized; the subsequent ones only differ for the convolutional kernel sizes, see section 2.3.1. On the right, the overall CNN structure is represented. Panel (c)—RNNs. On the left, the fundamental Gated Recurrent Unit (GRU) used to realize RNNs is schematized. On the right, the overall RNN structure is represented.

2.3.1.2. CNN

Unlikely the networks densely connected across artificial neurons (as in FCNNs), in CNNs neurons could also be sparsely connected between the *l*th and (*l*-1)th layers. That is, each hidden layer compute convolutions between the local input and a set of kernels, realizing sparse connections between its local input and local outputs (also called 'feature maps').

The input layer was composed by artificial neurons arranged in 2D replicating the input $X_{t,i}$. Then, hidden layers performing 2D convolutions were included. These were grouped into N_b convolutional blocks (schematized on the left of figure 3(b)), each including the stack of N_{lb} hidden convolutional layers. The very first convolutional layer 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 time domain using kernels of size $(1, F)$. F denotes the kernel size along the time axis. Each convolutional layer learned K kernels and performed convolutions with unitary stride and with a padding of $(0, F//2)$, where // is the floor division operator. For each hidden layer, activations were normalized via BN [65] and passed through an ELU non-linearity [66]. Each block ended with an average pooling layer using a kernel size and stride of $(1, 2)$, halving the temporal dimension, and with dropout [67] using a dropout rate p_{drop}. After the sequence of the convolutional blocks, an output layer with N_c neurons, fully-connected to the feature maps provided by the last layer of the convolutional module, was included, and activated via softmax non-linearity to provide as output the conditional probabilities $p(c_k|X_{t,i})$. Finally, the predicted class $yp_{t,i}$ was computed as the most probable one among the N_c classes. The CNN structure is schematized on the right of figure 3(b).

The 2D convolution performed in the lth convolutional layer between a kernel $W_f^{(l)}$ $(0 \leqslant f \leqslant K-1)$ and its local input $Y^{(l-1)}$ can be seen as a matrix multiplication between a sparse matrix $W_f'^{(l)}$ and the flattened local input $y^{(l-1)} = flatten\left(Y^{(l-1)}\right)$. The sparse matrix $W_f'^{(l)}$ is obtained from Toeplitz matrices computed on the zero-padded kernel $W_f^{(l)}$ [68]. From this reformulation, sparse and shared interactions between artificial neurons are better highlighted, due to the sparsity of $W_f'^{(l)}$ and to the reuse of the same weights within $W_f'^{(l)}$ (i.e. many connections share the same weight).

A single forward-pass through this network (during inference, when dropout is not applied) can be formalized as:

$$y^{(0)} = flatten\left(X_{t,i}\right) \in \mathbb{R}^{N \cdot T_z} \tag{6}$$

$$\begin{cases} y_f^{(l)} = ELU\left(BN\left(W_f'^{(l)} y^{(l-1)} + b^{(l)}\right)\right) \\ y_f^{(l)} = pool\left(y_f^{(l)}\right), if\ mod\left(l, N_{lb}\right) = 0, \quad 1 \leqslant l \leqslant N_{lb} \cdot N_b, 0 \leqslant f \leqslant K-1, \\ y^{(l)} = \left[y_0^{(l)}, \ldots, y_f^{(l)}, \ldots, y_{K-1}^{(l)}\right] \end{cases} \tag{7}$$

where $y^{(l)}$ denotes the lth layer output (obtained by concatenating the flattened activations contained in $y_f^{(l)}, \forall f$) and $mod(x,y)$ represents the modulo operating division returning the remainder after division of x by y.

$$\begin{cases} f(X_{t,i}; \vartheta, \eta) = softmax\left(W_o y^{(N_{lb} \cdot N_b)} + b_o\right) = p\left(c_k|X_{t,i}\right) \\ yp_{t,i} = \underset{k}{\operatorname{argmax}}\left(p\left(c_k|X_{t,i}\right)\right), \end{cases} \quad 0 \leqslant k \leqslant N_c - 1 \tag{8}$$

2.3.1.3. RNN

This network structure exploits recurrent connections across artificial neurons. A GRU is a RNN (schematized on the left of figure 3(c)), which can be seen as a variant of LSTMs with comparable performance while being less complex, more efficient, and introducing less trainable parameters [69, 70]. A GRU keeps memory of the past using a hidden state h_n and exploits update gate z_n and reset gate r_n to control the propagation of the input information at the time step n, in our study represented by the input features x_n collected across V6A neurons $X_{t,i} = [x_1, \ldots, x_n, \ldots, x_{Tz}]$. During forward pass, these gates process the information as follows, indicating with σ the sigmoid function, and with i and h the local input and hidden state, respectively.

$$z_n = \sigma\left(W_{iz} x_n + b_{iz} + W_{hz} h_{n-1} + b_{hz}\right) \tag{9}$$

$$r_n = \sigma\left(W_{ir} x_n + b_{ir} + W_{hr} h_{n-1} + b_{hr}\right) \tag{10}$$

The update gate will determine how much of the past information needs to be passed, while the reset gate will determine how much of the past information to forget (with z_n and r_n varying between 0 and 1). The gates take connections with the same artificial neurons but differ in the weights and biases and in

how these gates are used within the GRU. Next, a new memory content $\hat{h}_n$ is computed based on the information passed through the reset gate:

$$\hat{h}_n = tanh\left(W_{i\hat{h}}x_n + b_{i\hat{h}} + r_n \odot \left(W_{h\hat{h}}h_{n-1} + b_{h\hat{h}}\right)\right), \tag{11}$$

where $\odot$ indicates the element-wise product. Lastly, the information passed through the update gate is used to determine what to collect from the new memory content $\hat{h}_n$ and from the previous step h_{n-1}:

$$h_n = (1 - z_n) \odot \hat{h}_n + z_n \odot h_{n-1}. \tag{12}$$

The input layer was composed by artificial neurons arranged in 2D replicating the input $X_{t,i}$. The network was composed by the sequence of N_l hidden GRU layers (i.e. multilayer GRU or stacked GRU), each having H features in the hidden state. After each layer, dropout [67] with a dropout rate p_{drop} was applied. Lastly, an output layer with N_c neurons, fully-connected to the hidden state of the last layer at the last time step ($h_{T_z}^{(N_l)}$) was included, and activated via softmax non-linearity to provide as output the conditional probabilities $p(c_k|X_{t,i})$. Finally, the predicted class $yp_{t,i}$ was computed as the most probable one among the N_c classes. The RNN structure is schematized on the right of figure 3(c).

A single forward-pass through this network (during inference, when dropout is not applied) can be formalized as:

$$\begin{aligned} h^{(0)} &= \left[h_1^{(0)}, \ldots, h_n^{(0)}, \ldots h_{Tz}^{(0)}\right] = X_{t,i} \\ &= [x_1, \ldots, x_n, \ldots, x_{Tz}] \end{aligned} \tag{13}$$

$$\begin{cases} \hat{h}_n^{(l)} = tanh\left(W_{i\hat{h}}^{(l)}h_n^{(l-1)} + b_{i\hat{h}}^{(l)} + r_n^{(l)} \odot \left(W_{h\hat{h}}^{(l)}h_{n-1}^{(l)} + b_{h\hat{h}}^{(l)}\right)\right) \\ h_n^{(l)} = \left(1 - z_n^{(l)}\right) \odot \hat{h}_n^{(l)} + z_n^{(l)} \odot h_{n-1}^{(l)} \end{cases}, \; 1 \leqslant l \leqslant N_l, 1 \leqslant n \leqslant T_z \tag{14}$$

$$\begin{cases} f(X_{t,i}; \vartheta, \eta) = softmax\left(W_o h_{T_z}^{(N_l)} + b_o\right) = p(c_k|X_{t,i}) \\ yp_{t,i} = \underset{k}{\text{argmax}}\left(p(c_k|X_{t,i})\right) \end{cases}, \; 0 \leqslant k \leqslant N_c - 1 \tag{15}$$

2.3.2. Automatic hyper-parameter search

Automatic hyper-parameter search is devoted to finding the optimal hyper-parameters of a learning system on a validation set (different from the training and test sets). The optimal hyper-parameters η^*, so that $\eta^* = \underset{\eta}{\text{argmin}}\, k(\eta)$, are searched automatically, where $k(\eta)$ is an objective function being evaluated on the validation set. As objective function, the same loss function used to optimize the trainable parameters could be used or the objective function can be based on a specific performance measure (e.g. the accuracy or F1 score) [55, 71]. In this study, we used $k(\eta) = 1 - acc(\eta)$ as objective function, where $acc(\eta)$ is the average accuracy across all time samples and across epochs, obtained with a specific hyper-parameter configuration η. Therefore, the optimal hyper-parameters that maximize the decoding accuracy were searched. The hyper-parameters automatically searched for each network architecture (FCNN, CNN, RNN) are described in table 3.

For each hyper-parameter configuration, a new training stage and a new evaluation stage must be performed. Thus, depending on the number of hyper-parameters to optimize and on the model complexity (in general both high in DNNs), automatic hyper-parameter search can be expensive. Hyper-parameter search algorithms (e.g. grid search and random search) generally look for η^* without exploiting results from past iterations to select the array η to be evaluated in the next iteration (uninformed algorithms), often wasting time on unpromising η values.

Bayesian optimization overcomes this limitation, by suggesting in an informed way, the next hyper-parameters η to be evaluated. By investigating promising hyper-parameter configurations based on past results, Bayesian optimization can find better configurations than other approaches within fewer iterations compared to uninformed algorithms (e.g. grid or random search) [55]. Specifically, a Bayesian statistical model $p(k|\eta)$ of the objective function (called 'surrogate model') is used and it is updated after each iteration, by keeping track of past evaluation results, i.e. each pair $(\eta, k(\eta))$. Crucially, this surrogate model is easier to optimize than the actual objective function $k(\eta)$, and its optimization after each iteration is based on a criterion called 'selection function'. Then, after each iteration the hyper-parameters that perform best on the surrogate are used to update the structure of the learning system, for the training and evaluation in the next iteration. A more detailed description of this automatic hyper-parameter search algorithm used for

Table 3. Searched hyper-parameters: distributions and values. For each hyper-parameter, the value was sampled from a set of discrete values during hyper-parameter search.

	Hyper-parameter	Distribution	Values
FCNNs	No. of hidden layers (N_l)	Uniform	[1, 2, 3, 4]
	No. of units (N_u)	Uniform	[16, 32, 64, 128]
	Dropout rate (p_{drop})	Uniform	[0, 0.25, 0.5]
	Use batch norm.	Uniform	[False, True]
	Learning rate (lr)	Log-uniform	[10^{-4}, $5{\cdot}10^{-4}$, 10^{-3}, $5{\cdot}10^{-3}$, 10^{-2}]
CNNs	No. of blocks (N_b)	Uniform	[1, 2]
	No. of hidden layers per block (N_{lb})	Uniform	[1, 2, 3]
	No. of kernels (K)	Uniform	[4, 8, 16, 32]
	Kernel size in time axis (F)	Uniform	[11, 21, 31, 41]
	Dropout rate (p_{drop})	Uniform	[0, 0.25, 0.5]
	Use batch norm.	Uniform	[False, True]
	Learning rate (lr)	Log-uniform	[10^{-4}, $5{\cdot}10^{-4}$, 10^{-3}, $5{\cdot}10^{-3}$, 10^{-2}]
RNNs	No. of features in hidden state (H)	Uniform	[16, 32, 64, 128]
	No. of hidden layers (N_l)	Uniform	[1, 2, 3, 4]
	Dropout rate (p_{drop})	uniform	[0, 0.25, 0.5]
	Learning rate (lr)	Log-uniform	[10^{-4}, $5{\cdot}10^{-4}$, 10^{-3}, $5{\cdot}10^{-3}$, 10^{-2}]

neural decoding can be found in our previous study [44].

In this study, Bayesian optimization was performed for 100 iterations by using tree-structured Parzen estimator [55] as surrogate model and expected improvement as selection function. Considering each specific network family (FCNN, CNN, RNN), Bayesian optimization was performed for each decoding problem, monkey, and cross-validation fold, leading to 60 (=3·2·10) optimal hyper-parameter configurations.

2.3.3. Training settings and strategies

The cross-entropy between the predicted probability distribution (provided by the learning system) and the empirical distribution (provided by the labelled dataset) was used as loss function $j(\vartheta)$ to learn the trainable parameters contained in ϑ. Adam [72] was used as optimizer, searching for $\vartheta^* = \underset{\vartheta}{\operatorname{argmin}}\, j(\vartheta)$. The learning rate ($lr$) was selected via Bayesian optimization (see table 3) together with the other searched hyper-parameters. The mini-batch size was set to 64 and the maximum number of training epochs to 250. Lastly, the optimization stopped when the validation accuracy did not decrease after 50 consecutive training epochs (early stopping). An example of loss dynamic over training epochs during a single network training (i.e. a single cross-validation fold) is shown in figure S1 of supplementary materials with reference to the Bayesian-optimized CNN, in each decoding problem and each monkey. The loss is displayed up to the last performed epoch, as resulted from early stopping.

Once the hyper-parameter search was concluded (i.e. all 60 configurations of optimal hyper-parameters were derived), the following analysis was performed for each network family, separately. We identified a single hyper-parameter configuration for each network family, where each hyper-parameter was set to the value occurring more frequently during hyper-parameter search (as previously done in [44, 53] while decoding neural time series), across the different motor decoding problems. This unique hyper-parameter configuration was used to design the network. Then, for each decoding problem, each monkey and each cross-validation fold, the so designed network was trained and tested under four different training conditions. Three of them were conceived to analyze decoding performance of this unique hyper-parameter structure not only on the entire dataset but also on reduced datasets obtained by dropping out cells or training trials. Thus, we explored whether the reach and reach-to-grasp decoding abilities from V6A still persist when the dataset was artificially reduced, simulating scenarios where less cells or training trials are available. The fourth training condition served to test the effect of transferring the knowledge from one task (source) to another (target) within the same monkey (task-to-task transfer learning), by analyzing whether fine-tuning on the target task a network pre-trained on the source task was beneficial to improve performance compared to training from scratch, as a function of the number of training examples of the target task.

i. No dropping (whole dataset). Neural decoders were trained using the entire dataset available, i.e. using all recorded cells and training trials. Thus, only one training was performed for each decoding problem, monkey, and cross-validation fold. Here, neural networks were randomly initialized before training (networks were trained from scratch).
ii. Cell dropping. Neural decoders were 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, 10·7 trainings (10 trainings for each of the seven values of N') were performed for each decoding problem, monkey, and cross-validation fold. Here, neural networks were randomly initialized before training.

iii. Training trial dropping. Neural decoders were trained using a subset of training trials corresponding to the 12.5, 25, 37.5, 50, 62.5, 75, 87.5%, randomly sampled (10 times) from the entire training set. Thus, 10·7 trainings (10 trainings for each of the seven percentages) were performed for each decoding problem, monkey, and cross-validation fold. Here, neural networks were randomly initialized before training. It is worth noticing that, for each percentage considered, the number of examples (i.e. chunks of neural activity) used for training in this training condition scaled down with respect to the overall number of examples reported in table 2.

iv. Task-to-task transfer learning. At first, neural networks were trained in one reach-to-grasping condition, light or dark (source decoding task), using the entire training set as in point (i), for each monkey and cross-validation fold. Then, the networks were trained with training trials belonging to the other reach-to-grasping task, dark or light (target decoding task), but initializing the network with the trained parameters obtained in the source task within the same monkey, instead of using random values. The same was repeated by inverting the task used as source task and as target task. To evaluate transfer learning as a function of the number of training examples used in the target task, the training set for the target task was randomly sampled (10 times) using a percentage from 0% to 100% of the entire dataset, with a step of 12.5% (overall, eight percentages). The 0% condition corresponds to using the pre-trained network with no training on the target task. Therefore, here 10·8 trainings (10 trainings for each of percentage of training trials) were performed for each target task (reach-to-grasping light and dark), monkey, and cross-validation fold. Furthermore, to evaluate the beneficial effect of using transfer learning, for each percentage of training trials, we contrasted the performance obtained with the pre-trained network against the performance obtained using the network trained from scratch, by exploiting the results obtained in the previous points (iii) (for intermediate percentages) and point (i) (for 100% of training trials).

2.3.4. Neural decoders based on traditional machine learning

In addition to neural networks, a NB classifier was considered as neural decoder, as it was previously adopted to decode the same datasets used in this study [41, 42]. Therefore, this classifier is referred in the following as 'reference' machine learning algorithm, to distinguish it from the other machine learning approaches. This Bayesian decoder assumes conditional independency across input features (V6A activity from different neurons) and uses Bayes' rule to create a decoding model. The effects of each input neuron are combined linearly. Furthermore, we considered also SVM and XGBoost classifiers, as these algorithms resulted the best-performing traditional machine learning classifiers in a previous benchmark analysis performed on neurons' activity by Glaser *et al* [45]. To this aim, we adopted the same SVM and XGBoost classifiers as in Glaser *et al* [45] (accessible at https://github.com/KordingLab/Neural_Decoding). The traditional machine learning decoders were trained only according to training condition 2.3.3-i (i.e. considering the whole dataset), for comparison with DNNs under the same training condition (see section 3.2).

2.3.5. Processing of the evaluation metric

The accuracy was used as evaluation metric. The following processing was applied for each training performed with the previous training strategies. Each trained learning system (FCNN, CNN, RNN, NB, SVM, XGBoost) was tested, computing the accuracy chunk by chunk; note that in this way, we obtained a temporal pattern of decoding accuracy over the trial course thanks to the adopted sliding window decoding approach, that enabled to highlight the dynamics of reach encoding and reach-to-grasp encoding in V6A with a high time resolution (5 ms). Furthermore, in condition 2.3.3-ii and 2.3.3-iii, the accuracy was averaged, chunk by chunk, across the 10 random extractions for each dropping value. Therefore, one temporal pattern of accuracy per decoding problem, monkey and fold was obtained in the condition 2.3.3-i, while seven averaged temporal patterns of accuracy were obtained per decoding problem, monkey and fold in conditions 2.3.3-ii and 2.3.3-iii, each pattern corresponding to a different dropping value. Similarly, in the condition 2.3.3-iv, the accuracy was averaged, chunk by chunk, across the 10 random extractions for each percentage of training examples of the target task; overall, in this condition, nine averaged temporal patterns of accuracy were obtained per decoding problem, monkey and fold, each pattern corresponding to a different percentage of training examples.

Finally, as sliding window decoding highlighted the motor encoding of V6A in the temporal domain,

we considered the accuracies over time obtained with the top-performing learning system on the entire dataset (no dropping condition) to analyze the temporal motor encoding in V6A for each of the three motor paradigms.

2.4. Statistical analyses

The following tests were conducted on learning systems trained on the whole dataset (no dropping condition):

i. Comparison between the accuracy temporal dynamics scored with each neural network and each traditional machine learning algorithms (NB, SVM, and XGBoost), separately within each decoding problem. Permutation cluster tests (1000 iterations) with threshold-free cluster enhancement [73] were performed.
ii. Comparison of accuracies averaged within each epoch across decoders (NB, SVM, XGBoost, FCNN, CNN, RNN), separately for each decoding problem. Wilcoxon signed-rank pairwise tests were performed between learning systems (6 decoders), testing all possible comparisons within each epoch (15 comparisons per epoch). To correct for multiple tests (overall 15 tests/epoch·6 epochs = 90 tests), the Benjamini–Hochberg correction [74] was used.

The following statistical tests are relative to neural networks only, to further evaluate their performance in conditions simulating reduced datasets (point (iii)), under transfer learning condition (point (iv)), and when inferring information about motor encoding in area V6A (point (v)).

iii. Comparison between accuracies scored by different neural networks on reduced datasets, both while dropping out cells and training trials. Permutation cluster tests (1000 iterations) with threshold-free cluster enhancement [73] were performed. All combinations were tested (3 in total).
iv. Comparison between accuracies scored by CNNs while performing task-to-task transfer learning vs. training networks from scratch. Here, we tested the feasibility of task-to-task transfer learning for CNNs, as these networks resulted the most accurate decoder across the different percentages of training examples when dropping training trials (see section 3.2). Specifically, the temporal dynamics of the accuracy scored by the CNNs trained with transfer learning (i.e. pre-trained network) and without transfer learning (i.e. randomly initialized network) were compared, separately for each percentage of training trials considered. Permutation cluster tests (1000 iterations) with threshold-free cluster enhancement [73] were performed (9 total tests for each target task, i.e. reach-to-grasping light and reach-to-grasping dark).
v. Comparison of encoding measures—quantified by the accuracies of the top-performing network (i.e. CNNs, see section 3.2)—between reach vs. reach-to-grasp with the same illumination condition (light condition), and between reach-to-grasp tasks in the two different illumination conditions (light vs. dark). Permutation cluster tests (1000 iterations) with threshold-free cluster enhancement [73] were performed.

3. Results

3.1. Neural network structures resulting from Bayesian optimization

The optimal networks resulting from hyperparameter search (see figures S2–S4 of supplementary materials for the distributions of optimal hyperparameters for the three DNN families), consisted in: a FCNN with two hidden fully-connected layers; a shallow CNN with a single hidden convolutional layer performing convolution both in the space and time domains of the input multivariate neural time series (i.e. corresponding to a mixed spatio-temporal CNN); a deep RNN with three hidden GRU layers. Table 4 reports the computational complexity and computational time for the Bayesian-optimized neural networks. Computational complexity was expressed in terms of number of trainable parameters, and computational time indices were the training time per training epoch, number of training epochs required to converge, and training time required to converge (i.e. time per training epoch multiplied by the number of training epochs).

Remarkably, CNNs resulted the lightest DNNs, as they included approximately 73k trainable parameters (on average, across monkeys and decoding problems) compared to about 191k and 287k trainable parameters introduced by FCNNs and RNNs, respectively. Furthermore, CNNs were approximately as fast as FCNNs to be trained in each training epoch (0.74 s/training epoch vs. 0.66 s/training epoch, on average across monkeys and decoding problems), while RNNs resulted the slowest (3.06 s/training epoch). However, overall, FCNNs showed the lowest time to converge (8.4 s, 19.3 s, 77.0 s, respectively for FCNNs, CNNs, and RNNs), also due to their lowest number of training epochs required to converge (13, 26, 25 training epochs, respectively for FCNNs, CNNs, and RNNs).

3.2. Decoding analyses

The decoding accuracy when using the entire dataset (i.e. no dropping strategy) is reported in figure 4, that shows the accuracy temporal dynamics of the decoding of each DNN together with the accuracy of the three traditional machine learning approaches (NB, SVM, XGBoost).

Table 4. Computational complexity (as measured by the number of trainable parameters) and computational time indices (as measured by the training time per training epoch, number of training epochs required to converge, and training time required to converge) of the Bayesian-optimized neural networks while addressing different motor decoding problems. The number of trainable parameters, strictly tied to each monkey-specific decoder, are reported separately for each monkey, i.e. m1/m2 or m3/m4 for reaching and reach-to-grasping (both light and dark illumination conditions), respectively. Computational time indices are reported averaged across monkeys and cross-validation folds.

		Reaching (m1/m2)	Reach-to-grasping: light (m3/m4)	Reach-to-grasping: dark (m3/m4)
No. of trainable parameters	FCNNs	266 345/231 785	179 813/145 253	179 813/145 253
	CNNs	101 417/89 321	67 333/55 237	67 333/55 237
	RNNs	302 217/295 305	284 421/277 509	284 421/277 509
Time per training epoch (s/training epoch)	FCNNs	0.83	0.57	0.57
	CNNs	1.03	0.60	0.60
	RNNs	4.20	2.5	2.5
No. of training epochs to converge	FCNNs	9	12	19
	CNNs	26	27	25
	RNNs	24	21	31
Time to converge (s)	FCNNs	7.5	6.8	10.8
	CNNs	26.8	16.2	15.0
	RNNs	100.8	52.5	77.5

Among the traditional machine learning approaches, the highest accuracies were achieved by SVM in case of reaching decoding, while in case of reach-to-grasp decoding, XGBoost resulted the top-performing traditional machine learning decoder (especially during the movement epoch) for both light and dark condition. All DNNs proved to significantly outperform the reference machine learning algorithm previously adopted with these datasets (NB) widely in time, in particular from the early delay epoch (epoch 1) to hold epoch (epoch 5) while decoding reaching movements, and especially from the late delay epoch (epoch 2) to movement epoch (epoch 4) while decoding reach-to-grasping. Furthermore, only CNNs among DNNs also outperformed significantly the other two traditional approaches (SVM and XGBoost), widely across epochs. This result was consistent across decoding problems, except for reach-to-grasping in dark conditions, in which XGBoost was significantly outperformed by CNN only in epochs 3 (and not in other epochs).

Then, the neural decoders were compared in figure 5, where accuracies were averaged within each epoch to provide a more compact visualization of this comparative analysis. For brevity and to improve readability of figure 5, for each decoding task, only 6 out of 15 statistical results per epoch are displayed. These results are relative to the six pairwise comparisons per epoch among the four decoders including the three neural networks and the traditional machine learning algorithm that performed best in the specific decoding task among the three traditional algorithms (resulting SVM in case of reach decoding, and XGBoost in case of reach-to-grasp decoding, see figure 4). However, it is worth remarking that the displayed statistical results were obtained and corrected for multiple tests considering all 15 pairwise comparisons per epoch (resulting from considering all six decoders, see point (ii) in section 2.4), and not only 6 comparisons per epoch.

First, it is worth highlighting that the accuracies scored by decoders within the free epoch (epoch 0) were approximately at chance level, that is 11% and 20% for reach and reach-to-grasp decoding, respectively; this was expected, as the animal was not engaged in the movement task yet during this interval. That is, the free epoch represents a control interval in which classifiers should behave as random classifiers while decoding motor-related properties. Second, this analysis further emphasizes the significant improvement in performance of the DNNs compared to NB (reference machine learning algorithm), widely across task-related epochs (i.e. epochs following epoch 0), as already emerged from figure 4. Third, CNNs not only outperformed significantly both RNNs and FCNNs in all decoding problems in late delay, reaction time, and movement epochs (from epochs 2 to 4) but outperformed significantly also the top-performing machine learning algorithm (SVM for reaching and XGBoost for the two reach-to-grasping tasks) in all previous intervals and also in the hold epoch (overall, from epochs 2 to 5).

Moreover, in figure 6 neural networks were compared not only in terms of accuracy (as in previous figures), but also in terms of computational complexity (the number of trainable parameters, table 4) and computational time (here quantified by the training time to converge, table 4). In this figure, the decoding accuracy was extracted from the epoch with the highest performance (movement interval, i.e. epoch 4).

Across all decoding problems, the CNNs resulted the networks with best decoding accuracies and lowest computational complexity, while FCNNs were the ones with lowest computational time. From figure 6, CNNs represented a good compromise

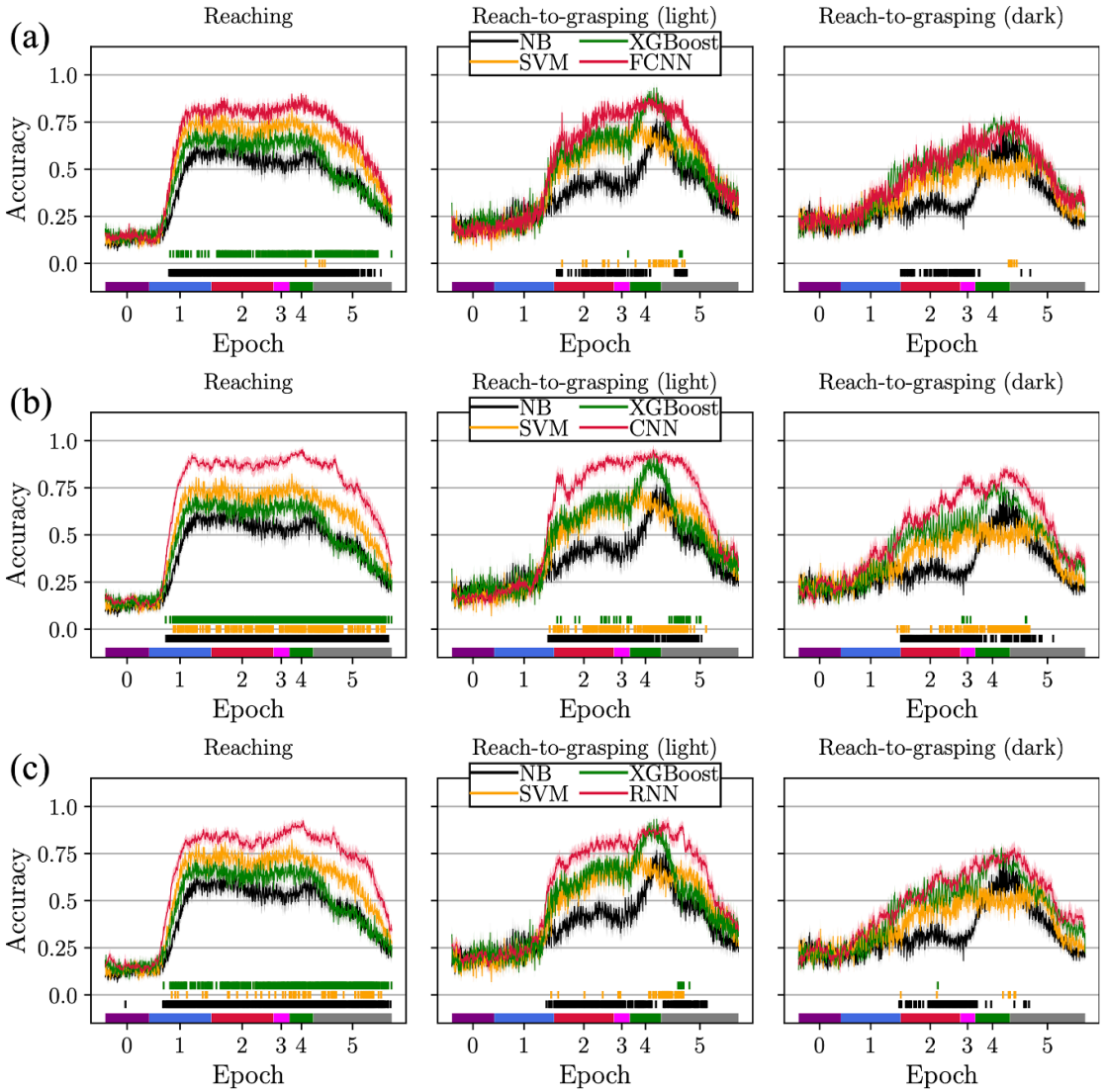

Figure 4. Accuracies over time of each DNN (red, in each panel) and the traditional machine learning algorithms, i.e. Naïve Bayes (NB, black), support vector machines (SVM, orange), and XGBoost (green). Each panel (from (a) to (c)) reports the accuracies over time scored with a DNN (FCNN, CNN, RNN, respectively) and the other machine learning decoders when decoding reach and reach-to-grasp (under the 2 different illumination conditions). Decoding accuracies over time are displayed by their mean values (thick lines) and standard error of the mean (overlayed area) across monkeys and cross-validation folds. In each plot, the vertical stripes on the bottom display the results of the performed permutation cluster test (see section 2.4), denoting the time intervals in which the DNN differed significantly ($p < 0.05$) compared to each machine learning approach. The stripes are color-coded as: black: NB vs. DNN, orange: SVM vs. DNN, green: XGBoost vs. DNN. Only significant comparisons are marked. 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: movement; grey: hold.

between decoding performance, computational complexity, and computational time, overall.

The performance analysis of DNNs was further extended by analyzing how their decoding capabilities change with variable-sized datasets, to further validate decoders with comparative analyses on smaller datasets. Figures 7 and 8 display the decoding accuracies obtained when the number of input cells and number of training trials were artificially reduced, respectively.

Even though neural networks were trained with less cells and training trials, they achieved accuracies well above the chance levels in the addressed decoding problems widely across the analyses with simulated reduced datasets. For example, accuracies were >0.6 within the movement epoch (the interval with maximum discharge in V6A neurons), on average, when using ⩾30 cells (40 cells in FCNNs) and ⩾25% of training trials. However, when reducing both the number of cells to decode and the number of training trials, CNNs resulted the top performing DNN. Indeed, CNNs significantly outperformed FCNNs with accuracy differences (dropping out cells/dropping out training trials) up to 15.6%/18.8%, 17.5%/22.1%, 19.1%/22.3%, and outperformed RNNs with accuracy differences up to 8.4%/12.6%, 11.4%/14.8%, 9.5%/13.5%, on average for reaching, reach-to-grasping (light) and reach-to-grasping

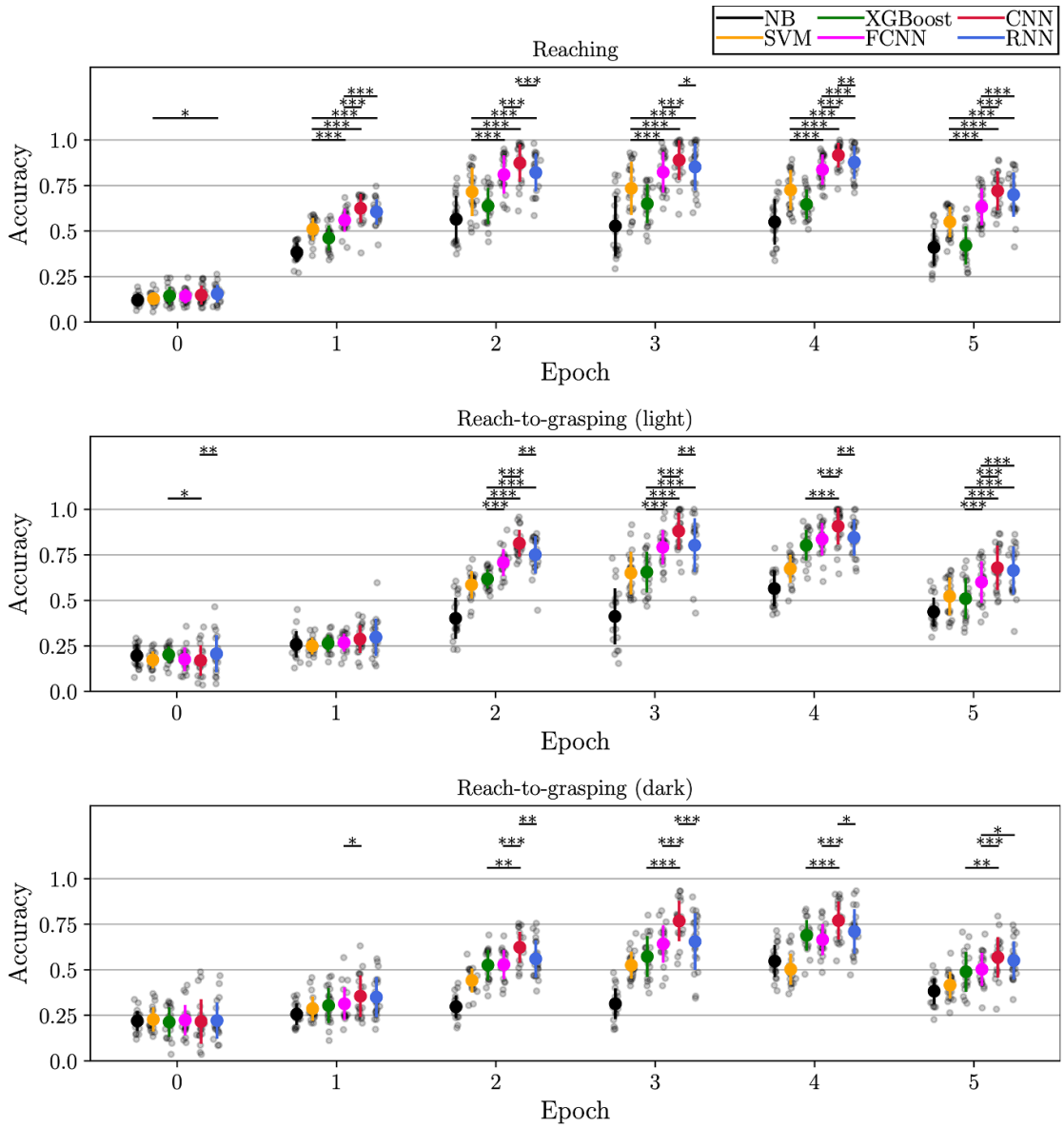

Figure 5. Accuracies averaged within epochs. The representations report the average accuracies within each epoch of the trial (see section 2.1) for all the neural decoders in case of reach decoding and reach-to-grasp decoding (in light and dark illuminations). Bigger dots represent the mean value, while the error bars represent the standard deviation across monkeys and cross-validation folds. Smaller dots represent the accuracy scored for each monkey and each cross-validation fold (i.e. 20 data points). The results from the performed Wilcoxon signed-rank tests are reported too (see section 2.4); p-values are corrected for multiple comparisons via the Benjamini–Hochberg procedure and only the significant comparisons are marked ($^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$). In addition, note that, for brevity and to provide a more compact visualization, only the results relative to the three neural networks and the machine learning algorithm performing best among the three ones (NB, SVM, XGBoost) in each decoding task (SVM for reaching, XGBoost for both reach-to-grasping tasks) are reported.

(dark), respectively. These differences were obtained mainly within epochs 1–5 when decoding reaching and within epochs 2–5 when decoding reach-to-grasping with both illumination conditions (i.e. both light and dark), and were scored mainly when using >10 cells (see figure 7) and widely across the percentages of training trials evaluated (see figure 8). Generally, CNNs achieved significantly higher accuracies compared to RNNs largely in epochs 2, 3 and 5, while this improvement did not always hold within the movement epoch (epoch 4) across the paradigms and performed analyses. In particular, in the cell dropping analysis with the lowest number of cells (10 cells), RNNs significantly outperformed CNNs when decoding reach-to-grasp (but not when decoding reach endpoints). However, this was the only condition where RNNs overcame CNNs. Lastly, FCNNs resulted the worst performing DNNs, as they obtained significantly lower accuracies compared to CNNs and RNNs (see figures 7 and 8), especially in the movement and hold epochs (epochs 4–5).

Overall, the performance improvement scored by CNNs when using the entire dataset (i.e. no dropping strategy, see figures 4 and 5), was maintained also

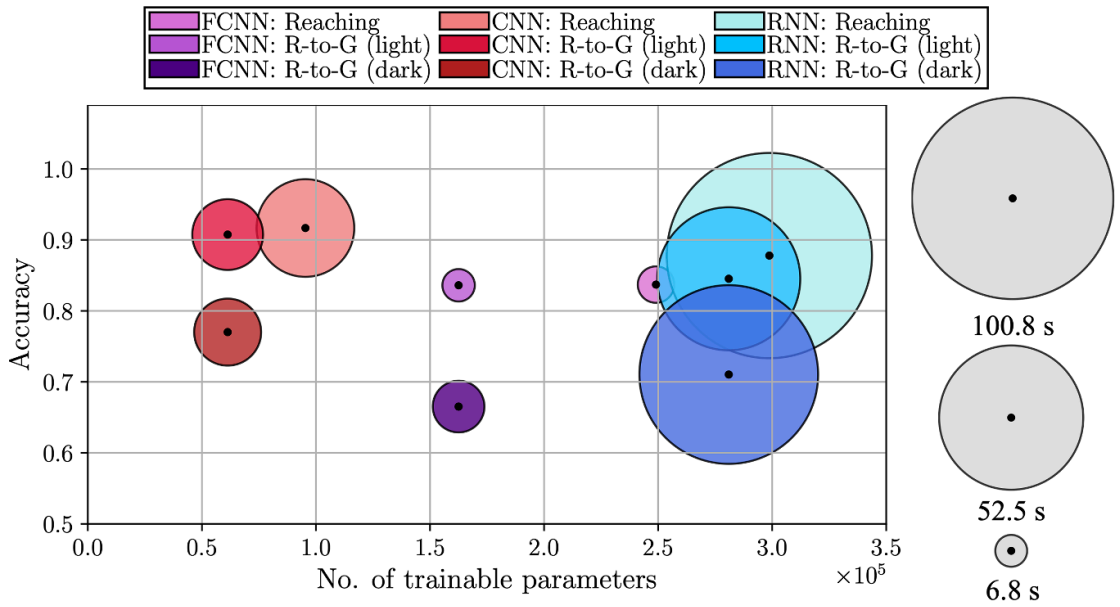

Figure 6. Comparison of neural networks in the plane performance (i.e. decoding accuracy in the movement epoch)—computational complexity (i.e. number of trainable parameters). Each network trained for each decoding task is denoted by a circle, and the circle size is modulated by the training time required to converge. All measures were averaged across monkeys and cross-validation folds. Note that reach-to-grasping was shortened to 'R-to-G' to increase readability.

while simulating smaller datasets in terms of recorded cells and recorded trials used to train decoders.

CNNs were additionally tested by evaluating the effect of task-to-task transfer learning in the reach-to-grasping tasks. Figures 9 and 10 display the decoding accuracies over time with (green) or without (black) transfer learning from reach-to-grasping in light condition (source task) to reach-to-grasp in dark condition (target task) and vice versa, respectively, as a function of the amount of training trials used in the target task.

From figures 9 and 10, task-to-task transfer learning resulted beneficial especially when using low percentages of training trials, with significant improvements in decoding performance at 0% (no training at all, pre-trained networks only tested on the target task) in both tested cases (from light to dark and from dark to light, figures 9 and 10) and at 12.5% in case of knowledge transfer from light to dark conditions (figure 9); at the same percentage, slight (but not significant) improvements were observed in the reversed condition (figure 10). It is worth noticing that slight (but not significant) improvements were observed also at higher percentages, e.g. when transferring knowledge from light to dark at 25%, 37.5% and 50% conditions, especially in the late delay epoch and reaction time epoch (epochs 2 and 3).

According to the previous results, the decoding of the two tasks appeared to largely rely on shared features, as the pre-trained network, even in absence of any training on the target task, perform well above chance. To inspect commonalities between the features learned in the two different reach-to-grasping tasks, we performed a correlation analysis between the convolutional filters learned by the CNN for decoding reach-to-grasping in light condition and in dark condition when using the whole training set (as in case of the source task in transfer learning). Since each CNN learned 32 filters, these were averaged together before computing correlation. The Pearson's correlation coefficients between the average filters learned in the two tasks were computed separately for each cross-validation fold and each monkey and are reported in figure 11(a). All correlations resulted statistically significant ($p < 0.001$). For most of the trained models (across monkeys and cross-validation folds), strong correlations were observed (between 0.7 and 1.0), and only for 2 out of 20 models the correlations were only moderate (between 0.4 and 0.7). These results confirm that, even though the models were trained independently on two different tasks, similar features were learned on the same monkey. Thus, transfer learning did benefit from these commonalities between features and improved the performance of decoders, especially when using a low amount of training examples (0% and 12.5% of training examples). Lastly, we also provide a visualization of the convolutional filters learned in reach-to-grasping in the two illumination conditions for monkey m4 (as an example), in a representative cross-validation fold, to qualitatively inspect similarities between filters. Filters were averaged together in their absolute value and normalized between 0 and 1, and the result is reported in figure 11(b), separately for light and dark conditions. By reporting the absolute value, we are highlighting the discriminatory power in space and time of these filters, as commonly performed in literature (e.g. see [53, 75] in case of

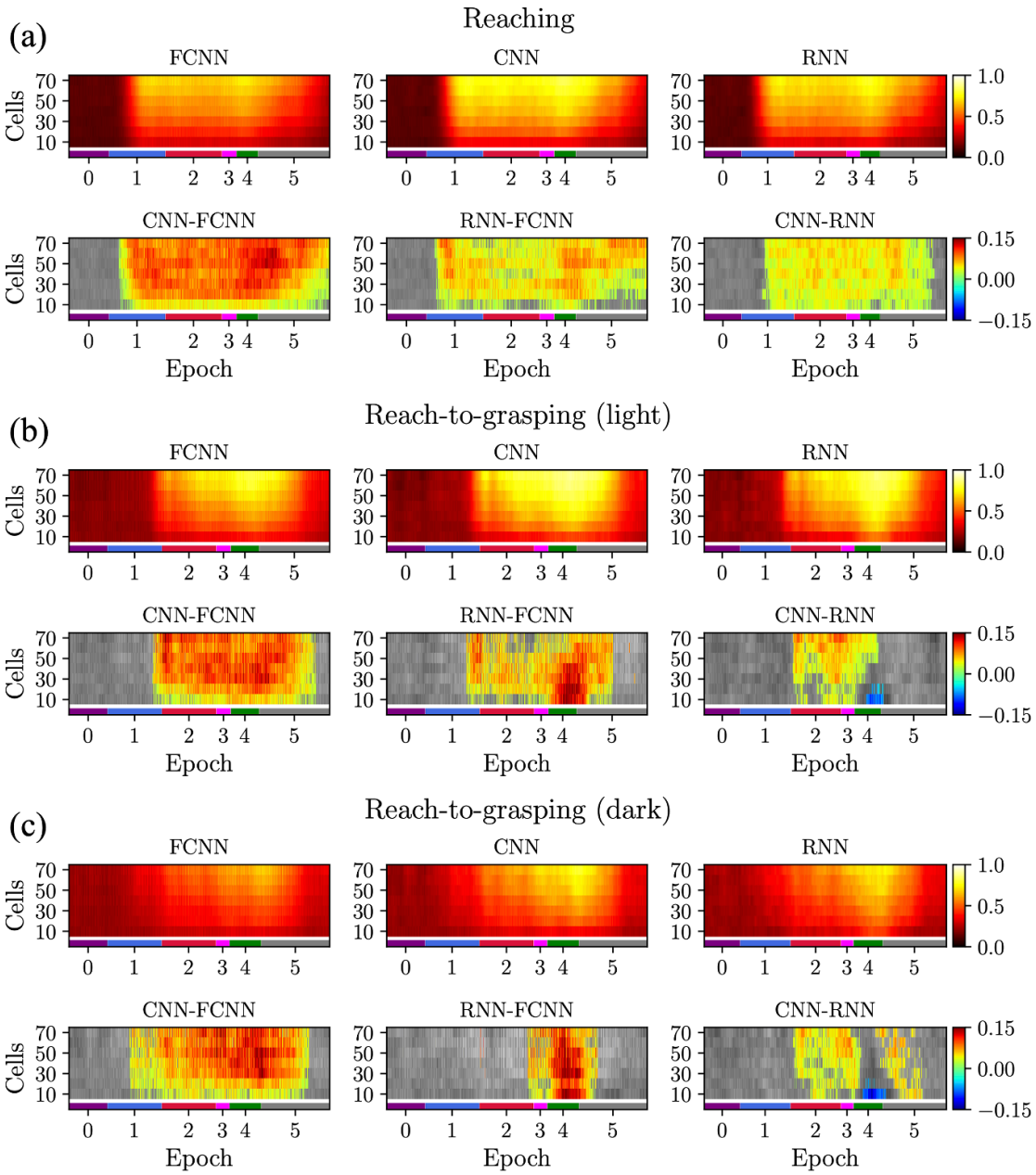

Figure 7. Accuracies over time while dropping out cells from the input multivariate neural activity. Results when decoding reaching, reach-to-grasping (light) and reach-to-grasping (dark) are reported in panels (a)–(c), respectively. The maps on top of each panel report the average accuracies of each type of DNNs as a function of the number of cells included in the input (from 10 to 70 with a step of 10 cells) and as a function of time. The maps on the bottom of each panel report the average performance differences between each pair of DNN types considering all possible combinations. In the difference maps, the results from the statistical analysis are embedded in the visualization, by coloring the significant ($p < 0.05$) differences resulting from the performed permutation cluster tests (see section 2.4) and leaving in grey the unsignificant ones. In all panels, the decoding accuracies and accuracy differences are displayed as heatmaps reporting the number of cells by row and the time steps by column and were obtained by averaging 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: movement; grey: hold.

EEG decoding). From these representations, it results that most cells had similar spatio-temporal dynamics of grip discriminatory power di between the two tasks, presumably representing neurons responding most to motor information (i.e. motor neurons of V6A), while few cells presented differences between tasks that could arise from the presence in V6A also of visuomotor neurons, modulating their activity depending on the light condition [76].

3.3. Encoding of reaching and reach-to-grasping in V6A

Based on the previous analyses (section 3.2), CNNs resulted the top-performing learning system for decoding the three motor tasks. Thus, we used the CNNs decoding accuracy over time to analyze how much movement information was represented by area V6A, i.e. to analyze neural encoding of V6A motor information. Figure 12 reports, for

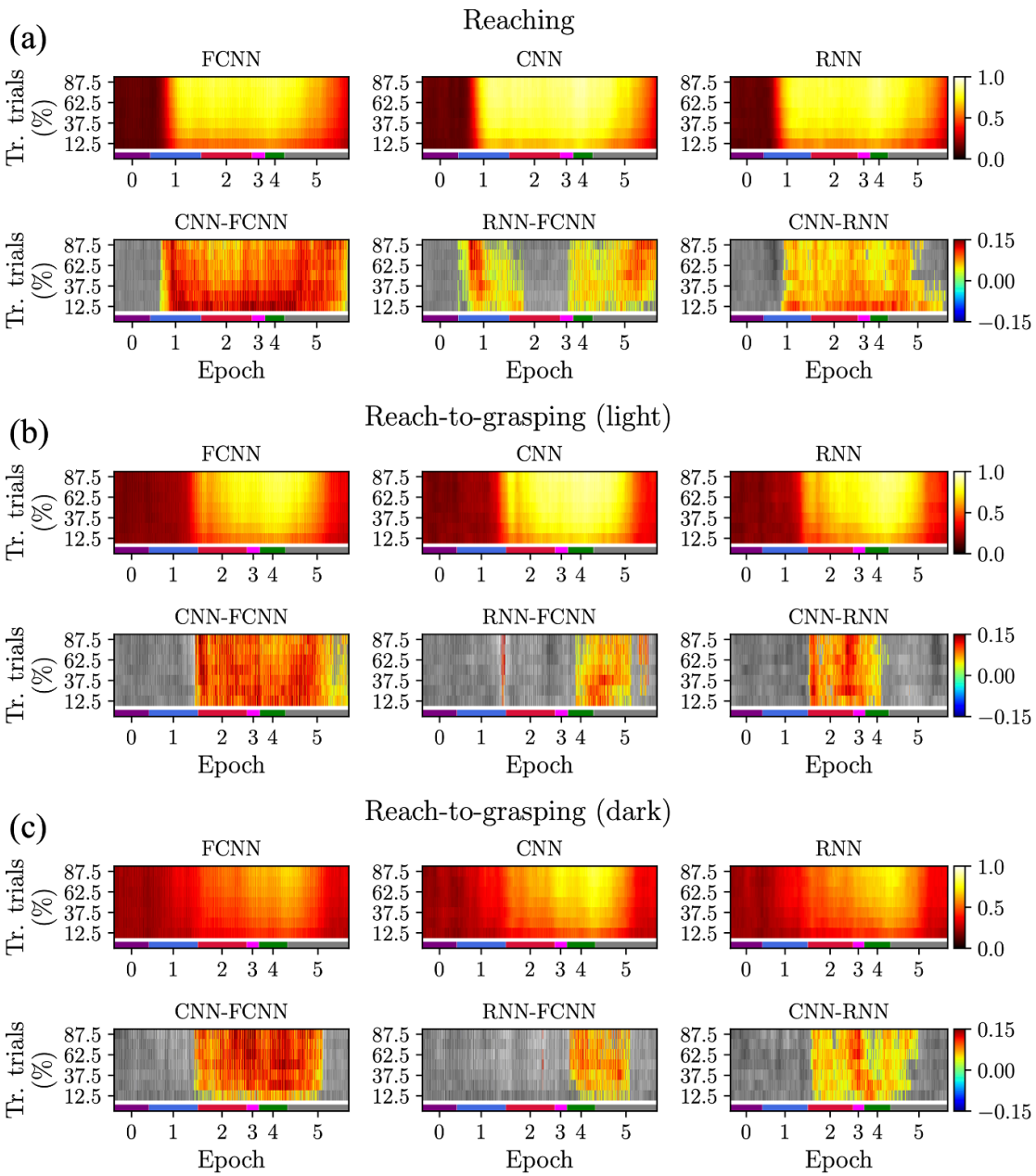

Figure 8. Accuracies over time while dropping out training trials while optimizing DNN trainable parameters. Results when decoding reaching, reach-to-grasping (light) and reach-to-grasping (dark) are reported in panels (a)–(c) respectively. The maps on top of each panel report the average accuracies of each type of DNNs as a function of the percentage of training trials used (from 12.5% to 87.5% with a step of 12.5%) and as a function of time. The maps on the bottom of each panel report the average performance differences between each pair of DNNs types, considering all possible combinations. In the difference maps, the results from the statistical analysis are embedded in the visualization, by coloring the significant ($p < 0.05$) differences resulting from the performed permutation cluster tests (see section 2.4) and leaving in grey the unsignificant ones. In all panels, the decoding accuracies and accuracy differences are displayed as heatmaps reporting by row the percentage of training trials and by column the time steps and were obtained by averaging 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: movement; grey: hold.

comparison, the decoding accuracy scored by the CNN in each of the three motor tasks, when the entire dataset was used (i.e. these waveforms replicate the red ones in figure 4(b)).

From figure 12, it is evident that the neural encodings (as measured by the decoding accuracy) shared a common trend across the three motor tasks: the encoding measure remains stable at the chance level during the free epoch (epoch 0), then starts increasing during the delay epochs (epochs 1 and 2), reaching a plateau within movement epoch (epoch 4) and decreases within the hold epoch (epoch 5). Despite these similarities in the overall temporal dynamics, significant differences can be observed within the early delay epoch between reaching and reach-to-grasping encodings with the same

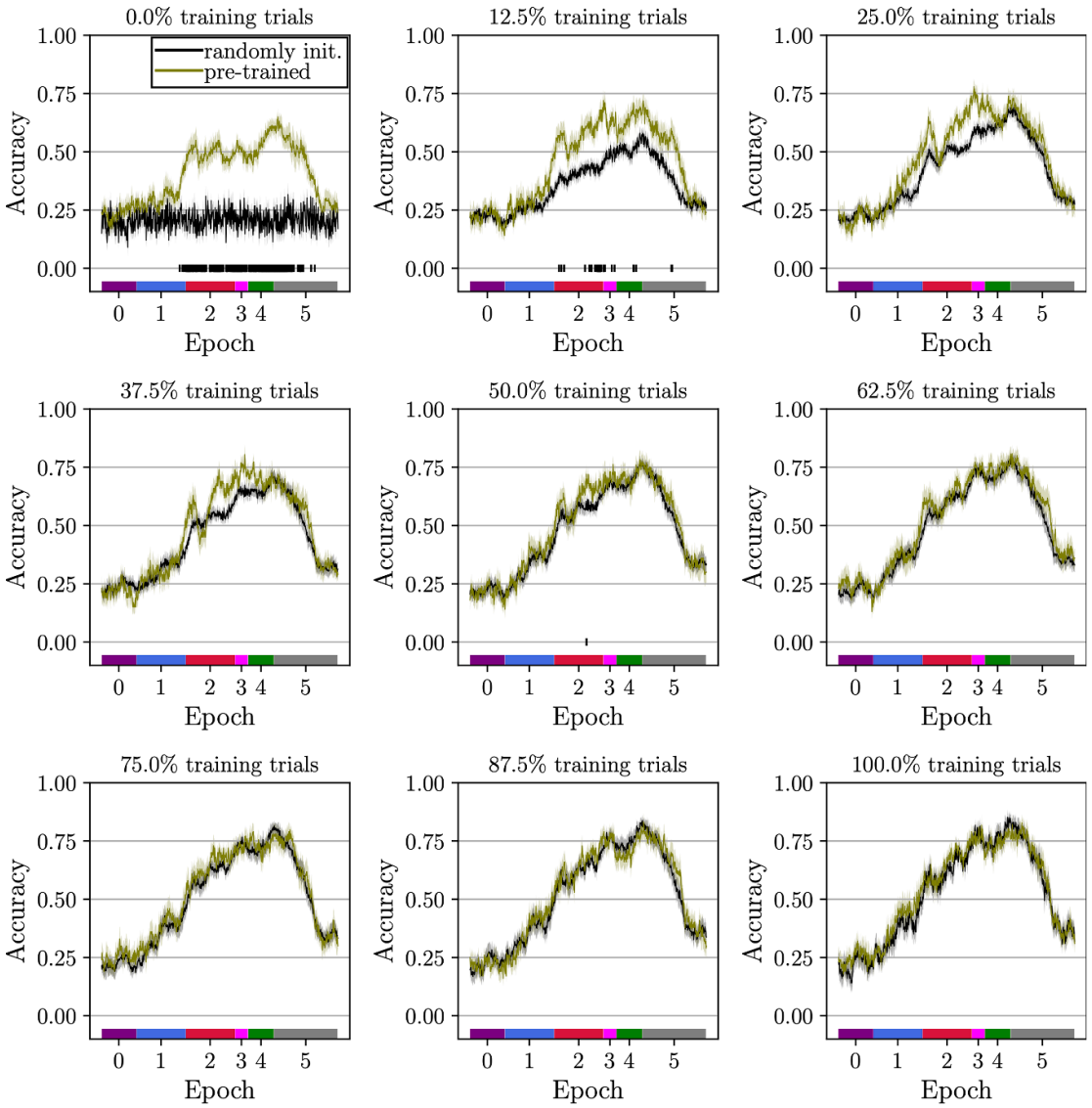

Figure 9. Accuracies over time while performing task-to-task transfer learning from reach-to-grasping in the light condition to the dark condition. Each panel reports the accuracies over time scored by the CNN while decoding reach-to-grasping in the dark condition (target task) in case of a specific percentage of training examples, both when the CNN was randomly initialized (black) and when it was pre-trained on reach-to-grasping in the light condition (source task, green). Decoding accuracies over time are displayed by their mean values (thick lines) and standard error of the mean (overlayed area) across monkeys and cross-validation folds. Vertical black stripes on the bottom display the results of the performed permutation cluster test (see section 2.4), denoting the time intervals where the performance scored by the CNN differed significantly ($p < 0.05$) between training from scratch (random initialization) and transfer learning (pre-trained network). Only significant comparisons are marked. 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: movement; grey: hold.

illumination condition (i.e. light), and within the late delay epoch between reach-to-grasping encodings in the two different illumination conditions (i.e. light vs. dark).

4. Discussion

Overall, this study presents a unique benchmark analysis of neural decoders on multiple motor decoding problems (3 different recording paradigms), exploiting signals collected in four macaque monkeys from area V6A, a pivotal parietal area of the dorsomedial visual stream of macaque brain. Specifically, neural networks (FCNNs, CNNs, RNNs) with Bayesian-optimized architectures were developed and applied to decode reaching and reach-to-grasping from the activity of V6A neurons and were compared to traditional machine learning approaches. Moreover, decoding capabilities of the neural networks were analyzed under different conditions, i.e. while using reduced datasets, both in terms of number of cells and of training trials, and while applying transfer learning. To the best knowledge of the authors, here, for the first time, transfer learning was applied to

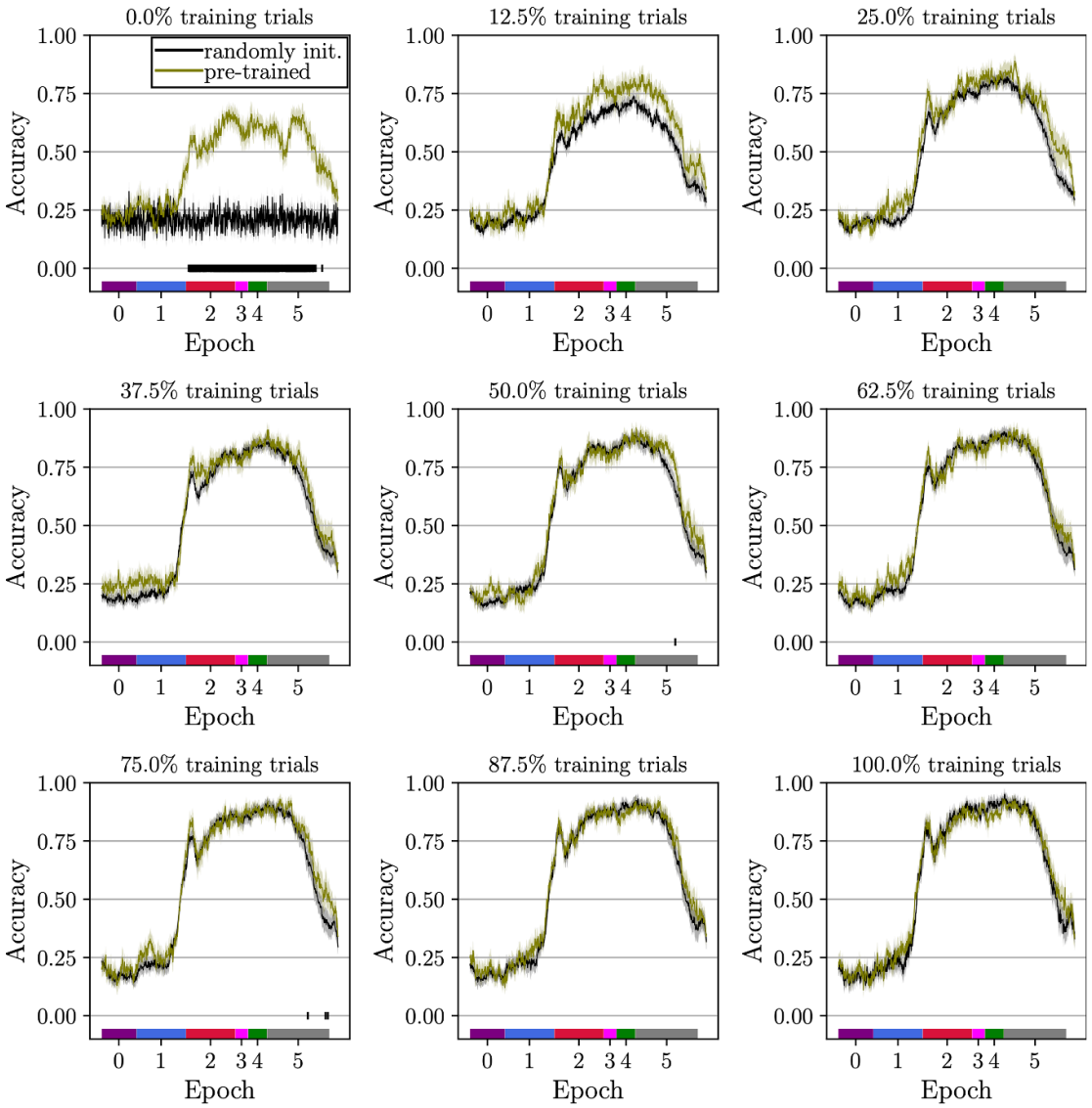

Figure 10. Accuracies over time while performing task-to-task transfer learning from reach-to-grasping in the dark condition to the light condition. Each panel reports the accuracies over time scored by the CNN while decoding reach-to-grasping in the light condition (target task) with a specific percentage of training examples, both when the CNN was randomly initialized (black) and when it was pre-trained on reach-to-grasping in the dark condition (source task, green). All conventions are as in figure 9.

single-neuron decoding, for transferring the knowledge from a task to another, with prospective implications for BCI usages. Lastly, by exploiting the sliding window approach, decoders enabled the analysis of V6A motor encoding in the temporal domain.

4.1. Decoding analyses

All neural networks significantly outperformed NB, representing in this study the reference machine learning approach since previously applied on the same datasets [41, 42]. Thus, neural networks were able to capture more relevant features to discriminate between different motor properties (either reach- or grip-related) compared to NB, widely over the entire task course (figure 4). It is worth and fair remarking that in the previous study by Filippini *et al* [42], while decoding the same reach-to-grasping dataset with a NB classifier and a sliding window decoding approach, accuracies up to and above 80% over time were obtained, hence, NB achieved higher performance than the ones reported in our study. While the study of Filippini *et al* [42] was of high importance to evidence the possibility of decoding grasping information from V6A neurons, their adopted procedure involves profound differences compared to the one adopted here. First, 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. Second, NB was separately trained and tested for each step of the sliding window approach (i.e. $\forall i$, see equations (1) and (2)), thus generating different decoders over time, i.e. a different decoder for each analyzed 300 ms chunk. Third, the dataset was binned at 20 ms. Here, all the

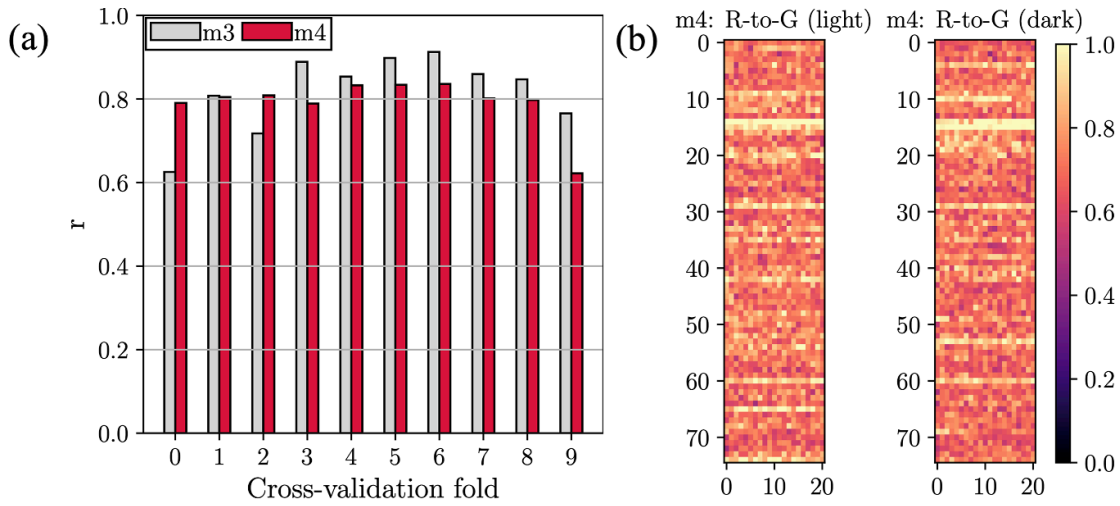

Figure 11. Commonalities between convolutional filters learned by the CNN in reach-to-grasping in light condition and dark condition. Panel (a)—Correlation analysis between light and dark conditions. Here, the Pearson's correlation coefficients (r) are reported (bar height) for each model, i.e. for each cross-validation fold and monkey (m3 in grey and m4 in red). For each decoding task (light and dark), each cross-validation fold, and each monkey, the 32 filters learned by the CNN were averaged together and the correlation was computed between the average filter obtained in light condition and dark condition, within each fold and each monkey. All correlations were statistically significant ($p < 0.001$). Panel (b)—Visualization of the average filters learned in the light (left) and dark (right) conditions for monkey m4 for a representative cross-validation fold. The average pattern (with shape (75, 21)) is visualized as an heatmap, with neurons along rows (75 cells for m4) and filter time steps along columns (21 time steps, see figure S3 of supplementary materials). In the figure, reach-to-grasping is shortened in 'R-to-G', to increase readability.

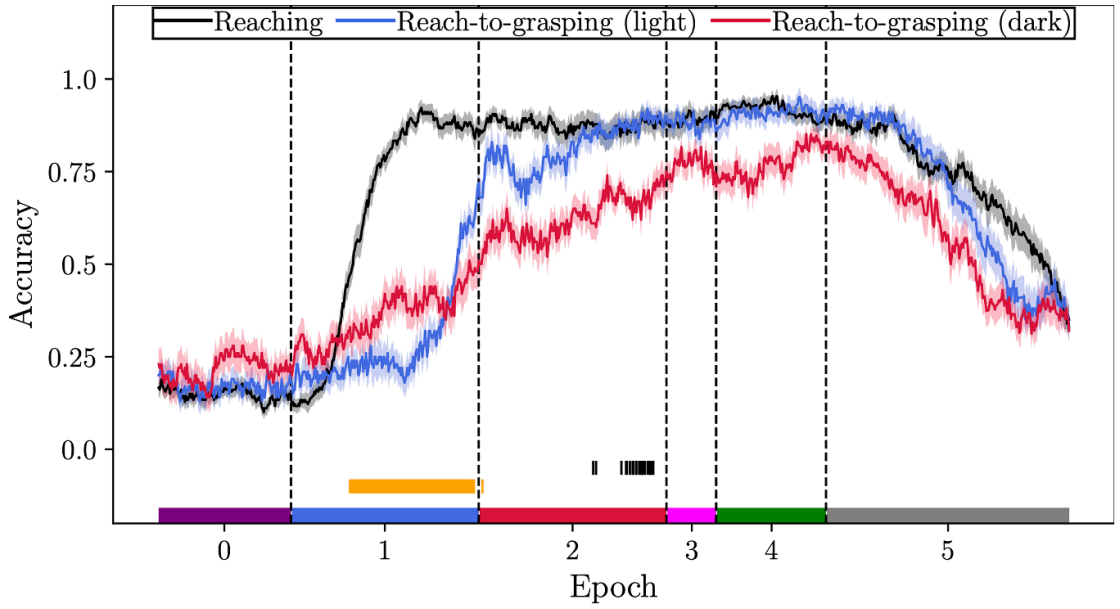

Figure 12. Encodings of reaching and reach-to-grasping in V6A as measured by the decoding accuracy scored by the top-performing learning system (i.e. the CNN). Note that these temporal patterns correspond to the ones reported in figure 4(b). The decoding accuracy over time is reported, displaying the mean value (thick lines) and standard error of the mean (overlayed area) across monkeys and cross-validation folds. Results from the performed permutation cluster tests (see section 2.4) are reported too. Orange vertical stripes: time intervals in which the V6A encoding of reaching was significantly ($p < 0.05$) different compared to reach-to-grasping with the same illumination condition (i.e. light). Black vertical stripes: time intervals in which the V6A encoding of reach-to-grasping differed significantly ($p < 0.05$) between the two illumination conditions (i.e. light vs. dark). Only significant comparisons are marked. 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: movement; grey: hold. The vertical dashed lines denote the separation between the epochs.

cells of the dataset were used, without any *a priori* selection, thus, leaving the learning systems the ability to explore all the available information for decoding and to learn to discard task-unrelated information. Furthermore, in this study, the decoders were 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

for each cross-validation fold, see section 2.1), but more challenging, as being time-aspecific. Remarkably, the sliding window training adopted here represents a more appropriate way to train and validate offline decoders for real-life BCI applications, as training a new decoder for each time step during the trial course is not feasible in practical applications. In addition, here, signals were binned within a shorter window (5 ms vs. 20 ms); therefore, the learning system could exploit also higher frequency components in gamma bands, which were suggested playing a key role during movements [60, 61]. All these points represent more challenging conditions for the neural decoder and the neural networks proved to outperform NB with these settings. Performance differences between neural networks and NB may be associated to the ability of neural networks to better capture temporal dynamics (e.g. via mixed spatio-temporal convolutions) and non-linearities encoded in neural activity, better extracting relevant features from the input neural activity, and discarding task-unrelated information.

These results confirm and extend the ones we obtained in a recent paper [44], where under the same settings of training and testing as used here, we found that the CNNs significantly outperformed the NB decoder in decoding reaching endpoints. Here, we significantly expanded our previous analysis by considering different families of neural networks and of traditional machine learning approaches (SVM and XGBoost), and different motor decoding problems (not only decoding reaching endpoints). From our results, the significant performance improvement in motor decoding previously observed in [44] using CNNs vs. NB classifier did hold across different motor decoding tasks and also across other machine learning algorithms (SVM and XGBoost). Conversely, considering the other networks (FCNNs, and RNNs) the significant improvement did hold only vs. NB but not vs. SVM and XGBoost, as these families of networks achieved only comparable accuracies with these last two traditional decoders. Therefore, CNNs resulted not only the top-performing neural network but also the top-performing algorithm, overall, across all the tested motor decoding problems.

It is worth noticing that in our previous study [44], CNNs were realized using separable convolutions, as we focused on a parsimonious structure in terms of trainable parameters, while here we used traditional convolutional layers for a more general evaluation. Interestingly, in both cases, by adopting Bayesian optimization, the most frequently selected CNN structure was shallow (with a single convolutional layer) and wide (i.e. with a large number of filters, here 32), suggesting that this CNNs structure was the most appropriate to address these motor decoding problems, independently on how convolutions are specifically implemented. To gain clearer evidence of this result, we also performed ablation tests (consisting in training and testing different variants of a network architecture by changing 1 hyperparameter at a time). Ablation tests were applied to the depth of the CNN to investigate the effect of the number of convolutional layers on the decoding accuracy. Starting from the CNN structure selected by Bayesian optimization (baseline version) we trained and tested other two CNN variants, differing from the baseline only by the number of total convolutional layers, which was set to 3 (i.e. 1 convolutional block composed by 3 convolutional layers) or 6 (i.e. 2 convolutional blocks, each composed by 3 convolutional layers). Results of ablation tests are reported in figure S5 of supplementary materials. The baseline CNN with only one convolutional layer was significantly more accurate ($p < 0.01$) than the other two variants in all the three decoding problems, further suggesting that shallower CNN architectures generalize better than deeper ones for motor decoding from neurons' activity.

Among the different types of neural networks tested here, CNNs were the best performing, especially from late delay up to movement epochs (epochs 2, 3, 4, see figures 4 and 5). Crucially, CNNs resulted the most accurate classifiers during the intervals in which the discharge of V6A neurons is most prominent, i.e. during the action execution and once the action is executed [39, 41, 42]. Even though decoding was performed independently for each 300 ms chunk of neural activity (see section 2.2), CNN predictions not only resulted accurate but also consistent through the trial course (see figure S6 of supplementary materials), the decoded class being stable in time during motor tasks.

In addition to the previous results on the whole datasets, CNNs overall outperformed the other neural networks from late delay up to movement epochs (epochs 2, 3, 4) also when dropping out training examples and cells (>10 cells) from the dataset (see figures 7 and 8). Only when a very small number of cells (10 cells) were retained in the input time series, RNNs outperformed CNNs while decoding reach-to-grasping (but not while decoding reaching endpoints). Indeed, in this interval RNNs resulted slightly more robust to a reduction in the number of input cells compared to CNNs (and also compared to FCNNs), as highlighted in figure S7 of supplementary materials showing the difference in performance between no dropping and dropping out cells, for each architecture (for completeness, figure S8 of supplementary materials shows the difference in performance between no dropping and dropping out trials, for each architecture). The advantage of RNNs in case of very small numbers of cells might be due to the different effect of the number of input cells on the network capacity (i.e. the ability of approximate functions) across the different networks (see figure S9 of supplementary materials). Specifically, by reducing the number of input cells, the model size

and, thus, the capacity of CNNs and FCNNs, resulted substantially reduced compared to RNNs. As a consequence, as model capacity reduced to a less extent in RNNs, these networks might be advantaged, especially in the interval in which the discharge of V6A neurons is maximum [39, 41, 42].

CNNs (overall behaving as top-performing decoders) were used also to transfer the knowledge from one reach-to-grasping task to another. Transfer learning proved to improve the performance of the neural decoder especially in the low data regime (e.g. 0% and 12.5%) and particularly when the target task is more challenging, i.e. reach-to-grasping decoding in darkness. This suggest that in prospective, BCI applications based on invasively recorded signals may also benefit from CNNs trained with task-to-task transfer learning to reduce the number of calibration examples and thus the calibration time to achieve high performance. Overall, the reduction of BCI calibration times might improve the practicality and usability of the interface, possibly increasing the engagement and learning during feedback.

The presented experiments aimed at providing a comprehensive evaluation of the principal DNNs (FCNNs, CNNs, RNNs) for motor decoding from single-neuron activity (here from V6A area). However, neural networks have been already widely proposed and tested for neural time series decoding, mainly using non-invasive modalities, especially EEG [47, 77, 78] due to its high portability, low cost and BCI applications. In the field of EEG decoding, CNNs are the most adopted networks [47], and share some common elements in their structure across studies [15, 16, 48, 49, 53, 79]. Typically, they first learn features in the temporal domain and then in the spatial domain, via separate convolutions in time and space of the multivariate EEG input. Then, more abstract features in the temporal domain are learned in subsequent deeper layers. Among these architectures, EEGNet (consisting of three convolutional layers and one fully-connected layer) [79] and DeepConvNet (five convolutional layers and one fully-connected layer) [48], represent the first successful and general-purpose designs (also for motor decoding), and are used as reference designs. However, the following main pitfalls should be addressed, when trying to transpose DNNs proposed for a non-invasive decoding application (e.g. from EEG) to an invasive decoding application (e.g. from single-neuron activity). First, when applied to EEG decoding, CNNs include more than one convolutional layer in their structure; conversely, from our results, CNNs for decoding motor states from single neurons seem to benefit from extremely shallow architectures with only one convolutional layer. Second, CNNs applied to EEG have proved to result more accurate when temporal convolutions are performed first and then spatial convolutions. However, when addressing single-neuron decoding, the optimal sequence for domain-specific convolutions should be deeply investigated ex novo, due to the different recording modality (invasive vs. non-invasive) and different nature of the neural time series (e.g. neuron spiking rate vs. scalp electric potentials). Therefore, neural networks specific for EEG could be transposed in the future to decode single-neuron activity by: (i) considering a reduction in the number of convolutional layers; (ii) assessing the optimal sequence for convolutions operating in separate domains (e.g. temporal followed by spatial convolutions vs. spatial followed by temporal convolutions).

4.2. Encoding of reaching and reach-to-grasping in V6A

The sliding window approach adopted here permits the neural decoders to provide a straightforward visualization of V6A motor encoding in the temporal domain. Here, the accuracy temporal dynamic of CNNs, resulting the top-performing decoders, was considered for analyzing V6A motor encoding. Crucially, the temporal dynamics reported in figure 12 exhibited a common ramp-up (during the delay interval), plateau and ramp-down (from the end of movement execution) trend across paradigms, highlighting that also the delay interval (both in its early and especially in the late parts) strongly encodes both reaching and reach-to-grasping properties, in addition to the reaction time and movement intervals. That is, also the interval associated to movement planning proved to be a good candidate for decoding spatial reaching positions and grip types from V6A neurons. This interval is close to motor execution, but it is not already influenced by possible afferent feedback signals (known to be present in V6A [80–82]), thus, it could reflect action planning and predictive motor control [28]. In particular, an intermediate visuomotor transformation stage occurs, in which the visual information is converted into motor commands [83]. Overall, the temporal dynamics of the encoding measure reflected the neural population discharges [39, 41, 42]. Indeed, V6A was found discriminating among the different reached endpoints/grip shapes as soon as the motor plan can be formulated because of the illumination of the target LED to reach (fixation epoch)/illumination of the object to grasp (object visualization epoch). Then, the discrimination power of V6A population only slightly increases/remains constant while the monkey is preparing the reaching/reach-to-grasping action and peaks when the action is performed [39, 41, 42].

Regarding reach encoding in V6A, the position of the reached endpoints appeared to be encoded already from about the middle (or even before) of the early delay epoch (i.e. 300–500 ms after cueing the animal). From the beginning of the fixation epoch, the animal was already fixating at the position to be reached and waiting for the visual cue. V6A neurons are known to be modulated not

only by the spatial positions of reaching [82], but also by gaze position [59] and spatial attention [84]. Therefore, these factors may have also contributed at providing high discriminative power of the neural decoder early during the delay epoch. Conversely, in the reach-to-grasp encoding in V6A, considering the same illumination condition as in the reaching task (light condition), the encoding of grip properties became pronounced at the very end of the early delay epoch, closer to action execution. Thus, the intermediate visuomotor transformation appeared delayed in reach-to-grasping tasks. This could be due to a more complex motor planning required to translate the specific object shape into the corresponding grip code, or to the lower spatial processing requirements compared to reaching. Furthermore, grip-related properties appeared less encoded in the dark condition, overall resulting in (see the red curve compared to the blue curve in figure 12): (i) a slower ramp of encoding measure during the delay epoch, especially in epoch 1, where the steep increase observed in the light condition was absent; (ii) a lowered encoding especially during the late delay and movement intervals. That is, without accessing the visual attributes in the delay epoch, the visuomotor transformation not only resulted weaker (mainly due to the visuomotor nature of V6A area) but also slower. This further substantiates the importance of visual information for movement preparation in V6A. Indeed, V6A neurons are on average less active (lower firing rates) if movement preparation and execution are performed in the dark, and this was attributed to the attentional effort in keeping the object to grasp in memory which produces an inhibition in V6A neurons [85].

Similar temporal dynamics in accuracies were obtained in Filippini *et al* [41, 42, 44] on the same data using sliding window decoding. Remarkably, at a variance with the method applied in the previous studies [41, 42, 44], here neural encodings were investigated with a finer temporal resolution of 5 ms instead of 20 ms.

Overall, the results on offline motor decoding obtained in this study suggest that CNNs represent potential candidates to realize neural decoders for BCI applications in humans from recordings in medial PPC. In fact, these networks improved the recognition rate of motor properties already from motor planning (late delay epoch) compared to the other examined learning systems. Indeed, as CNNs achieved higher accuracies particularly during motor preparation across all recording paradigms (even under challenging conditions of less input neurons and less training trials), these decoders could be adopted in BCI systems to accurately decode motor-related properties. Moreover, due to the anticipated nature of decoding, CNNs can also cope with the delays that occur in BCI systems and that negatively affect the contingency between the recording of the neural activity and the feedback provided to the BCI user [3] (e.g. delays in processing neural activity into output commands and in driving the external device). Crucially, this performance improvement was also accompanied with a lower computational time and complexity (see table 4). Thus, CNNs not only provided an accurate prediction but also resulted in a model being more parsimonious (among all DNNs) and faster to train (especially when compared to RNNs). Therefore, CNNs may also help to keep limited the BCI calibration time, required to adapt the BCI system to the specific user.

Besides the previous prospective impacts of the present study in BCI practical applications, the obtained results also indicate that a data-driven approach based on DNNs applied to the analysis of V6A motor encoding provides promising insights and may contribute to deepen our knowledge of the functional role of this area. In future, this approach can be applied to single-neuron recordings obtain in other paradigms and brain areas, extending behind motor tasks, to sensory or perceptual tasks, with further potential implications both for neurophysiological knowledge and for practical advancement.

Of course, the present study is also affected by some limitations, that will be addressed in future studies. First, the data-driven analysis of V6A motor encoding was performed only in the temporal domain and should be extended also to the other domains characterizing multivariate neural signals, such as spatial and frequency domains. In this regard, future developments could benefit from the adoption of post-hoc explanation techniques [86] (e.g. saliency maps or layer-wise relevance propagation), devoted to highlight useful features in a domain under investigation, as recently obtained in CNNs for EEG processing [15, 16, 49, 53, 62, 87–89]. Second, all the performed analyses were conducted offline; thus, the insights provided from the performed offline investigations should be validated online in future studies.

5. Conclusion

In conclusion, in this study we investigated both design and application of neural networks for motor neural decoding from V6A activity, using a sliding window decoding approach. Signals were recorded from area V6A of four macaque monkeys, overall, during three different recording paradigms involving reaching and reach-to-grasping, and a comparative analysis was performed on three different neural network families.

Neural networks were optimized in their architecture by using Bayesian optimization across the addressed motor decoding problems, separately for FCNNs, CNNs, RNNs. The resulting optimal network structures may provide useful indications to researchers when designing neural networks for motor decoding. For both reach and reach-to-grasp

decoding (the latter with two different illumination conditions), neural networks significantly outperformed the reference machine learning algorithm (NB); furthermore, CNNs significantly outperformed also other traditional machine learning approaches (XGBoost, SVM), and the other neural networks (FCNNs and RNNs) in all motor decoding problems. Notably, the performance gain obtained in CNNs, especially during motor preparation, was maintained also while simulating practical scenarios in which less neurons/trials are recorded. Moreover, task-to-task transfer learning using CNNs proved to increase the decoding performance especially in the low data regime, suggesting that neural networks (specifically CNNs) can be also used, in prospective, in practical BCI scenarios providing not only the most accurate translation of the neural activity, but also a reduction of BCI calibration times.

Time pattern of motor encoding in V6A can be analyzed from the decoding accuracy thanks to the adopted sliding window approach. By analyzing motor encoding from CNN decoding accuracies, a different latency in the intermediate visuomotor transformation was observed between the encoding of reached spatial positions and grip types; furthermore, the visuomotor transformation involving grip-related motor plan resulted slower and weaker while the monkey was waiting for the go-signal in complete darkness.

Overall, the high decoding capabilities of CNNs on a variety of training scenarios (notably, during motor planning), together with their light and fast-to-train nature, suggest that future studies might take advantage from the design and application of CNNs not only for offline evaluations but in prospective also in online evaluations, while decoding human medial PPC to control external devices for patients (e.g. with tetraplegia or with neurodegenerative disorders).

Data availability statement

The data cannot be made publicly available upon publication because they are not available in a format that is sufficiently accessible or reusable by other researchers. The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the TITAN V used for this research. The provider was not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

This study was supported by 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.

This work is also supported by #NEXTGENERATIONEU (NGEU) and funded by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), Project MNESYS (PE0000006)—A Multiscale integrated approach to the study of the nervous system in health and disease (DN. 1553 11.10.2022).

Author contributions

In the following are listed the authorship contributions:

- **Davide Borra**: Conceptualization, Methodology, Software, Formal Analysis, Visualization, Writing—Original Draft.
- **Matteo Filippini**: Data Curation, Investigation, Visualization, Writing—Review & Editing.
- **Mauro Ursino**: Formal Analysis, Resources, Validation, Writing—Review & Editing.
- **Patrizia Fattori**: Data Curation, Funding acquisition, Resources, Supervision, Writing—Review & Editing.
- **Elisa Magosso**: Conceptualization, Formal Analysis, Resources, Supervision, Writing—Review & Editing.

Conflict of interest

The authors declare no conflict of interest.

Ethical statement

The study was performed in accordance with the guidelines of EU Directives (86/609/EEC; 2010/63/EU) and Italian national law (D.L. 116-92, D.L. 26-2014) for the care and use of animals for scientific purposes. Experimental protocols have been approved by the Ethical Committee of the University of Bologna and by the Animal Welfare Body of the University of Bologna.

ORCID iDs

Davide Borra https://orcid.org/0000-0003-3791-8555
Matteo Filippini https://orcid.org/0000-0002-0730-4088
Mauro Ursino https://orcid.org/0000-0002-0911-0308
Patrizia Fattori https://orcid.org/0000-0002-0079-3755
Elisa Magosso https://orcid.org/0000-0002-4673-2974

References

[1] Wolpaw J R, Birbaumer N, McFarland D J, Pfurtscheller G and Vaughan T M 2002 Brain–computer interfaces for communication and control *Clin. Neurophysiol.* **113** 767–91

[2] Millán J D R 2010 Combining brain-computer interfaces and assistive technologies: state-of-the-art and challenges *Front. Neurosci.* **4** 161

[3] Wilson J A, Mellinger J, Schalk G and Williams J 2010 A procedure for measuring latencies in brain–computer interfaces *IEEE Trans. Biomed. Eng.* **57** 1785–97

[4] Schalk G and Leuthardt E C 2011 Brain-computer interfaces using electrocorticographic signals *IEEE Rev. Biomed. Eng.* **4** 140–54

[5] Schalk G, Kubánek J, Miller K J, Anderson N R, Leuthardt E C, Ojemann J G, Limbrick D, Moran D, Gerhardt L A and Wolpaw J R 2007 Decoding two-dimensional movement trajectories using electrocorticographic signals in humans *J. Neural Eng.* **4** 264–75

[6] Pistohl T, Ball T, Schulze-Bonhage A, Aertsen A and Mehring C 2008 Prediction of arm movement trajectories from ECoG-recordings in humans *J. Neurosci. Methods* **167** 105–14

[7] Brandman D M, Cash S S and Hochberg L R 2017 Review: human intracortical recording and neural decoding for brain–computer interfaces *IEEE Trans. Neural Syst. Rehabil. Eng.* **25** 1687–96

[8] Collinger J L, Wodlinger B, Downey J E, Wang W, Tyler-Kabara E C, Weber D J, McMorland A J, Velliste M, Boninger M L and Schwartz A B 2013 High-performance neuroprosthetic control by an individual with tetraplegia *Lancet* **381** 557–64

[9] Hochberg L R *et al* 2012 Reach and grasp by people with tetraplegia using a neurally controlled robotic arm *Nature* **485** 372–5

[10] Aflalo T *et al* 2015 Decoding motor imagery from the posterior parietal cortex of a tetraplegic human *Science* **348** 906–10

[11] Waldert S, Preissl H, Demandt E, Braun C, Birbaumer N, Aertsen A and Mehring C 2008 Hand movement direction decoded from MEG and EEG *J. Neurosci.* **28** 1000–8

[12] Yeom H G, Kim J S and Chung C K 2013 Estimation of the velocity and trajectory of three-dimensional reaching movements from non-invasive magnetoencephalography signals *J. Neural Eng.* **10** 026006

[13] Georgopoulos A P, Langheim F J P, Leuthold A C and Merkle A N 2005 Magnetoencephalographic signals predict movement trajectory in space *Exp. Brain Res.* **167** 132–5

[14] Bradberry T J, Rong F and Contreras-Vidal J L 2009 Decoding center-out hand velocity from MEG signals during visuomotor adaptation *NeuroImage* **47** 1691–700

[15] Borra D, Fantozzi S and Magosso E 2020 EEG motor execution decoding via interpretable sinc-convolutional neural networks *15th Mediterranean Conf. on Medical and Biological Engineering and Computing MEDICON 2019* ed J Henriques, N Neves and P de Carvalho (Cham: Springer International Publishing) pp 1113–22

[16] Borra D, Fantozzi S and Magosso E 2020 Interpretable and lightweight convolutional neural network for EEG decoding: application to movement execution and imagination *Neural Netw.* **129** 55–74

[17] Tangermann M *et al* 2012 Review of the BCI competition IV *Front. Neurosci.* **6** 55

[18] Bradberry T J, Gentili R J and Contreras-Vidal J L 2010 Reconstructing three-dimensional hand movements from noninvasive electroencephalographic signals *J. Neurosci.* **30** 3432–7

[19] Kobler R J, Sburlea A I and Müller-Putz G R 2018 Tuning characteristics of low-frequency EEG to positions and velocities in visuomotor and oculomotor tracking tasks *Sci. Rep.* **8** 17713

[20] Lv J, Li Y and Gu Z 2010 Decoding hand movement velocity from electroencephalogram signals during a drawing task *Biomed. Eng. Online* **9** 64

[21] Korik A, Sosnik R, Siddique N and Coyle D 2018 Decoding imagined 3D hand movement trajectories from EEG: evidence to support the use of Mu, beta, and low gamma oscillations *Front. Neurosci.* **12** 130

[22] Úbeda A, Hortal E, Iáñez E, Perez-Vidal C and Azorín J M 2015 Assessing movement factors in upper limb kinematics decoding from EEG signals *PLoS One* **10** e0128456

[23] Úbeda A, Azorín J M, Chavarriaga R and R Millán J D 2017 Classification of upper limb center-out reaching tasks by means of EEG-based continuous decoding techniques *J. NeuroEng. Rehabil.* **14** 9

[24] Kobler R J, Almeida I, Sburlea A I and Müller-Putz G R 2020 Using machine learning to reveal the population vector from EEG signals *J. Neural Eng.* **17** 026002

[25] Tam W, Wu T, Zhao Q, Keefer E and Yang Z 2019 Human motor decoding from neural signals: a review *BMC Biomed. Eng.* **1** 22

[26] Waldert S 2016 Invasive vs. non-invasive neuronal signals for brain-machine interfaces: will one prevail? *Front. Neurosci.* **10** 295

[27] Andersen R A, Kellis S, Klaes C and Aflalo T 2014 Toward more versatile and intuitive cortical brain–machine interfaces *Curr. Biol.* **24** R885–97

[28] Cui H 2016 Forward prediction in the posterior parietal cortex and dynamic brain-machine interface *Front. Integr. Neurosci.* **10** 35

[29] Gardner E P 2017 Neural pathways for cognitive command and control of hand movements *Proc. Natl Acad. Sci. USA* **114** 4048–50

[30] Santandrea E, Breveglieri R, Bosco A, Galletti C and Fattori P 2018 Preparatory activity for purposeful arm movements in the dorsomedial parietal area V6A: beyond the online guidance of movement *Sci. Rep.* **8** 6926

[31] Musallam S, Corneil B D, Greger B, Scherberger H and Andersen R A 2004 Cognitive control signals for neural prosthetics *Science* **305** 258–62

[32] Mulliken G H, Musallam S and Andersen R A 2008 Decoding trajectories from posterior parietal cortex ensembles *J. Neurosci.* **28** 12913–26

[33] Schaffelhofer S, Agudelo-Toro A and Scherberger H 2015 Decoding a wide range of hand configurations from macaque motor, premotor, and parietal cortices *J. Neurosci.* **35** 1068–81

[34] Hauschild M, Mulliken G H, Fineman I, Loeb G E and Andersen R A 2012 Cognitive signals for brain-machine interfaces in posterior parietal cortex include continuous 3D trajectory commands *Proc. Natl Acad. Sci.* **109** 17075–80

[35] Klaes C *et al* 2015 Hand shape representations in the human posterior parietal cortex *J. Neurosci.* **35** 15466–76

[36] Bosco A, Breveglieri R, Chinellato E, Galletti C and Fattori P 2010 Reaching activity in the medial posterior parietal cortex of monkeys is modulated by visual feedback *J. Neurosci.* **30** 14773–85

[37] Bosco A, Breveglieri R, Hadjidimitrakis K, Galletti C and Fattori P 2016 Reference frames for reaching when decoupling eye and target position in depth and direction *Sci. Rep.* **6** 21646

[38] Breveglieri R, Galletti C, Dal Bò G, Hadjidimitrakis K and Fattori P 2014 Multiple aspects of neural activity during reaching preparation in the medial posterior parietal area V6A *J. Cogn. Neurosci.* **26** 878–95

[39] Hadjidimitrakis K, Bertozzi F, Breveglieri R, Bosco A, Galletti C and Fattori P 2014 Common neural substrate for processing depth and direction signals for reaching in the monkey medial posterior parietal cortex *Cereb. Cortex* **24** 1645–57

[40] Fattori P, Raos V, Breveglieri R, Bosco A, Marzocchi N and Galletti C 2010 The dorsomedial pathway is not just for reaching: grasping neurons in the medial parieto-occipital cortex of the macaque monkey *J. Neurosci.* **30** 342–9

[41] Filippini M, Breveglieri R, Hadjidimitrakis K, Bosco A and Fattori P 2018 Prediction of reach goals in depth and direction from the parietal cortex *Cell Rep.* **23** 725–32
[42] Filippini M, Breveglieri R, Akhras M A, Bosco A, Chinellato E and Fattori P 2017 Decoding information for grasping from the macaque dorsomedial visual stream *J. Neurosci.* **37** 4311–22
[43] Filippini M, Morris A P, Breveglieri R, Hadjidimitrakis K and Fattori P 2020 Decoding of standard and non-standard visuomotor associations from parietal cortex *J. Neural Eng.* **17** 046027
[44] Filippini M, Borra D, Ursino M, Magosso E and Fattori P 2022 Decoding sensorimotor information from superior parietal lobule of macaque via convolutional neural networks *Neural Netw.* **151** 276–94
[45] Glaser J I, Benjamin A S, Chowdhury R H, Perich M G, Miller L E and Kording K P 2020 Machine learning for neural decoding *eNeuro* **7** ENEURO.0506–19.2020
[46] Livezey J A and Glaser J I 2021 Deep learning approaches for neural decoding across architectures and recording modalities *Brief. Bioinform.* **22** 1577–91
[47] Roy Y, Banville H, Albuquerque I, Gramfort A, Falk T H and Faubert J 2019 Deep learning-based electroencephalography analysis: a systematic review *J. Neural Eng.* **16** 051001
[48] Schirrmeister R T, Springenberg J T, Fiederer L D J, Glasstetter M, Eggensperger K, Tangermann M, Hutter F, Burgard W and Ball T 2017 Deep learning with convolutional neural networks for EEG decoding and visualization *Hum. Brain Mapp.* **38** 5391–420
[49] Borra D, Fantozzi S and Magosso E 2021 A lightweight multi-scale convolutional neural network for P300 decoding: analysis of training strategies and uncovering of network decision *Front. Hum. Neurosci.* **15** 655840
[50] Borra D, Fantozzi S and Magosso E 2020 Convolutional Neural Network for a P300 Brain-Computer Interface to Improve Social Attention in Autistic Spectrum Disorder *15th Mediterranean Conf. on Medical and Biological Engineering and Computing—MEDICON 2019* ed J Henriques, N Neves and P de Carvalho (Cham: Springer International Publishing) pp 1837–43
[51] Simões M *et al* 2020 BCIAUT-P300: a multi-session and multi-subject benchmark dataset on autism for P300-Based brain-computer-interfaces *Front. Neurosci.* **14** 568104
[52] Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H and He Q 2020 A comprehensive survey on transfer learning (arXiv:1911.02685)
[53] Borra D, Magosso E, Castelo-Branco M and Simoes M 2022 A Bayesian-optimized design for an interpretable convolutional neural network to decode and analyze the P300 response in autism *J. Neural Eng.* **19** 046010
[54] Farahat A, Reichert C, Sweeney-Reed C and Hinrichs H 2019 Convolutional neural networks for decoding of covert attention focus and saliency maps for EEG feature visualization *J. Neural Eng.* **16** 066010
[55] Bergstra J, Bardenet R, Bengio Y and Kégl B 2011 Algorithms for hyper-parameter optimization *24th Int. Conf. on Neural Information Processing Systems—NIPS 2011*
[56] Shi R, Zhao Y, Cao Z, Liu C, Kang Y and Zhang J 2022 Categorizing objects from MEG signals using EEGNet *Cogn. Neurodyn.* **16** 365–77
[57] Peterson S M, Steine-Hanson Z, Davis N, Rao R P N and Brunton B W 2021 Generalized neural decoders for transfer learning across participants and recording modalities *J. Neural Eng.* **18** 026014
[58] Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L and Lerer A 2017 Automatic differentiation in PyTorch *31st Int. Conf. on Neural Information Processing System—NIPS 2017*
[59] Galletti C, Battaglini P P and Fattori P 1995 Eye position influence on the parieto-occipital area PO (V6) of the macaque monkey *Eur. J. Neurosci.* **7** 2486–501
[60] Ball T, Demandt E, Mutschler I, Neitzel E, Mehring C, Vogt K, Aertsen A and Schulze-Bonhage A 2008 Movement related activity in the high gamma range of the human EEG *NeuroImage* **41** 302–10
[61] Nowak M, Zich C and Stagg C J 2018 Motor cortical gamma oscillations: what have we learnt and where are we headed? *Curr. Behav. Neurosci. Rep.* **5** 136–42
[62] Solon A J, Lawhern V J, Touryan J, McDaniel J R, Ries A J and Gordon S M 2019 Decoding P300 variability using convolutional neural networks *Front. Hum. Neurosci.* **13** 201
[63] Iwana B K and Uchida S 2021 An empirical survey of data augmentation for time series classification with neural networks *PLoS One* **16** e0254841
[64] Lashgari E, Liang D and Maoz U 2020 Data augmentation for deep-learning-based electroencephalography *J. Neurosci. Methods* **346** 108885
[65] Ioffe S and Szegedy C 2015 Batch normalization: accelerating deep network training by reducing internal covariate shift *32nd Int. Conf. on Machine Learning* vol 37, ed F Bach and D Blei (Lille: PMLR) pp 448–56
[66] Clevert D-A, Unterthiner T and Hochreiter S 2015 Fast and accurate deep network learning by exponential linear units (elus) (arXiv:1511.07289)
[67] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 Dropout: a simple way to prevent neural networks from overfitting *J. Mach. Learn. Res.* **15** 1929–58
[68] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press)
[69] Chung J, Gulcehre C, Cho K and Bengio Y 2014 Empirical evaluation of gated recurrent neural networks on sequence modeling (arXiv:1412.3555)
[70] Cho K, van Merrienboer B, Bahdanau D and Bengio Y 2014 On the properties of neural machine translation: encoder-decoder approaches (arXiv:1409.1259)
[71] Yu T and Zhu H 2020 Hyper-parameter optimization: a review of algorithms and applications (arXiv:2003.05689)
[72] Kingma D P and Ba J 2017 Adam: a method for stochastic optimization (arXiv:1412.6980)
[73] Smith S and Nichols T 2009 Threshold-free cluster enhancement: addressing problems of smoothing, threshold dependence and localisation in cluster inference *NeuroImage* **44** 83–98
[74] Benjamini Y and Hochberg Y 1995 Controlling the false discovery rate: a practical and powerful approach to multiple testing *J. R. Stat. Soc.* B **57** 289–300
[75] Cecotti H and Graser A 2011 Convolutional neural networks for P300 detection with application to brain-computer interfaces *IEEE Trans. Pattern Anal. Mach. Intell.* **33** 433–45
[76] Gamberini M, Galletti C, Bosco A, Breveglieri R and Fattori P 2011 Is the medial posterior parietal area V6A a single functional area? *J. Neurosci.* **31** 5145–57
[77] Zhang X, Yao L, Wang X, Monaghan J, McAlpine D and Zhang Y 2021 A survey on deep learning-based non-invasive brain signals: recent advances and new frontiers *J. Neural Eng.* **18** 031002
[78] Craik A, He Y and Contreras-Vidal J L 2019 Deep learning for electroencephalogram (EEG) classification tasks: a review *J. Neural Eng.* **16** 031001
[79] Lawhern V J, Solon A J, Waytowich N R, Gordon S M, Hung C P and Lance B J 2018 EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces *J. Neural Eng.* **15** 056013
[80] Breveglieri R, Kutz D F, Fattori P, Gamberini M and Galletti C 2002 Somatosensory cells in the parieto-occipital area V6A of the macaque *NeuroReport* **13** 2113–6
[81] Fattori P, Breveglieri R, Bosco A, Gamberini M and Galletti C 2017 Vision for prehension in the medial parietal cortex *Cereb. Cortex* **27** 1149–63

[82] Fattori P, Kutz D F, Breveglieri R, Marzocchi N and Galletti C 2005 Spatial tuning of reaching activity in the medial parieto-occipital cortex (area V6A) of macaque monkey *Eur. J. Neurosci.* **22** 956–72
[83] Fattori P, Breveglieri R, Raos V, Bosco A and Galletti C 2012 Vision for action in the macaque medial posterior parietal cortex *J. Neurosci.* **32** 3221–34
[84] Galletti C, Breveglieri R, Lappe M, Bosco A, Ciavarro M and Fattori P 2010 Covert shift of attention modulates the ongoing neural activity in a reaching area of the macaque dorsomedial visual stream *PLoS One* **5** e15078
[85] Breveglieri R, De Vitis M, Bosco A, Galletti C and Fattori P 2018 Interplay between grip and vision in the monkey medial parietal Lobe *Cereb. Cortex* **28** 2028–42
[86] Montavon G, Samek W and Müller K-R 2018 Methods for interpreting and understanding deep neural networks *Digit. Signal Process.* **73** 1–15
[87] Borra D and Magosso E 2021 Deep learning-based EEG analysis: investigating P3 ERP components *J. Integr. Neurosci.* **20** 791–811
[88] Zhao D, Tang F, Si B and Feng X 2019 Learning joint space–time–frequency features for EEG decoding on small labeled data *Neural Netw.* **114** 67–77
[89] Vahid A, Mückschel M, Stober S, Stock A-K and Beste C 2020 Applying deep learning to single-trial EEG data provides evidence for complementary theories on action control *Commun. Biol.* **3** 112