PAPER

Deep learning for neural decoding in motor cortex

IOP Publishing J. Neural Eng. 19 (2022) 056021 https://doi.org/10.1088/1741-2552/ac8fb5

Journal of Neural Engineering

PAPER

Deep learning for neural decoding in motor cortex

Fangyu Liu[1], Saber Meamardoost[2], Rudiyanto Gunawan[2], Takaki Komiyama[3], Claudia Mewes[5], Ying Zhang[6], EunJung Hwang[3,4,*] and Linbing Wang[1,*]

[1] Department of Civil and Environmental Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, United States of America
[2] Department of Chemical and Biological Engineering, University at Buffalo, Buffalo, NY 14260, United States of America
[3] Department of Neurobiology, Center for Neural Circuits and Behavior, and Department of Neurosciences, University of California San Diego, La Jolla, CA 92093, United States of America
[4] Cell Biology and Anatomy Discipline, Center for Brain Function and Repair, Chicago Medical School, Rosalind Franklin University of Medicine and Science, North Chicago, IL 60064, United States of America
[5] Department of Physics and Astronomy, University of Alabama, Tuscaloosa, AL 35487, United States of America
[6] Department of Cell and Molecular Biology, University of Rhode Island, Kingston, RI 02881, United States of America
[*] Authors to whom any correspondence should be addressed.

E-mail: EunJung.Hwang@rosalindfranklin.edu **and** wangl@vt.edu

Keywords: deep learning, neural decoding, neural signals, concurrent decoding, time-delay decoding, spatiotemporal decoding

RECEIVED
7 April 2022

REVISED
26 July 2022

ACCEPTED FOR PUBLICATION
6 September 2022

PUBLISHED
23 September 2022

Abstract

Objective. Neural decoding is an important tool in neural engineering and neural data analysis. Of various machine learning algorithms adopted for neural decoding, the recently introduced deep learning is promising to excel. Therefore, we sought to apply deep learning to decode movement trajectories from the activity of motor cortical neurons. *Approach*. In this paper, we assessed the performance of deep learning methods in three different decoding schemes, concurrent, time-delay, and spatiotemporal. In the concurrent decoding scheme where the input to the network is the neural activity coincidental to the movement, deep learning networks including artificial neural network (ANN) and long-short term memory (LSTM) were applied to decode movement and compared with traditional machine learning algorithms. Both ANN and LSTM were further evaluated in the time-delay decoding scheme in which temporal delays are allowed between neural signals and movements. Lastly, in the spatiotemporal decoding scheme, we trained convolutional neural network (CNN) to extract movement information from images representing the spatial arrangement of neurons, their activity, and connectomes (i.e. the relative strengths of connectivity between neurons) and combined CNN and ANN to develop a hybrid spatiotemporal network. To reveal the input features of the CNN in the hybrid network that deep learning discovered for movement decoding, we performed a sensitivity analysis and identified specific regions in the spatial domain. *Main results*. Deep learning networks (ANN and LSTM) outperformed traditional machine learning algorithms in the concurrent decoding scheme. The results of ANN and LSTM in the time-delay decoding scheme showed that including neural data from time points preceding movement enabled decoders to perform more robustly when the temporal relationship between the neural activity and movement dynamically changes over time. In the spatiotemporal decoding scheme, the hybrid spatiotemporal network containing the concurrent ANN decoder outperformed single-network concurrent decoders. *Significance*. Taken together, our study demonstrates that deep learning could become a robust and effective method for the neural decoding of behavior.

1. Introduction

Understanding the relationship between neural activity and perception, cognition, emotion, and movement is one of the most fundamental goals of neuroscience. Accordingly, neural decoding that reads out the sensory stimulus, cognitive processes, emotional states, and motor behavior from neural

signals has become an important analytic tool in neuroscience. One common example of neural decoding is to predict the animal's movement trajectories or analyze the movement parameters based on the neural signals recorded in the motor cortex [1–3]. Another example of neural decoding is to predict the animal's decisions according to the neural activity in associative brain areas such as prefrontal and parietal cortices [4–6]. Predicting the animal's spatial locations based on the neuronal activity in the hippocampus [7, 8] is also a common application.

In many neuroscience applications, traditional decoding methods such as population vectors [9], multiple linear regressions [10], and the Kalman filter [11] have been used due to their relatively simple algorithms and low computational costs [2, 12, 13]. Although less frequent, nonlinear methods that aim to more accurately characterize neural activity have also been developed and applied [13]. These methods include particle filters [14], point process filters [15], hybrid filters [16], mixture of trajectory models [17], and nonlinear dynamic models [18]. These various decoding methods played pivotal roles in elucidating the neural mechanisms underlying various brain functions, but their performance still suffers from the complex nature of neural processes that each method cannot fully capture [13].

A decoder could be regarded as a function approximator [19], mapping neural signals to behaviorally relevant variables such as sensory stimulus, working memory, and motor plans. Neurons encode those variables in a highly complex and nonlinear manner. Deep learning is becoming an appealing solution for neural decoding because of its ability to learn complicated, nonlinear transformations from data [19]. Compared with traditional methods, deep learning could not only significantly improve the decoding accuracy [20] but also help to more accurately identify critical input features [21]. Simple hypothesis-driven models rely on specific assumptions about a biological mechanism that deep learning does not embody, with a risk that it may miss some important, relevant input features [21]. In this case, deep learning could serve as an approximate bound for simple models. In other words, if a human-generated simple hypothesis-driven model is less accurate than a deep learning model in the same task, the hypothesis-driven model likely has failed to capture important principles [21]. Given these promising advantages of deep learning, in this paper, we implemented a variety of deep learning decoders to predict animals' movements from neural activity and examined their performance compared to traditional methods. Furthermore, we analyzed the input features discovered by deep learning decoders to gain insight into the principles underlying the neural code of movements.

2. Methods

2.1. Experimental data

The raw data (i.e. calcium signals of individual neurons and forelimb movement trajectories of a mouse) were obtained from the previous study [3]. Briefly, the mouse was trained to perform a cued-lever press task for two weeks (session 1–14; 1 session/day). In this task, the mouse received a reward when pressing the lever beyond the set thresholds within 10–30 s after auditory cue onset (figure 1(a)). Over the two weeks, the activity of the same set of neurons ($N = 200$) was recorded during the task through two-photon calcium imaging at approximately 28 Hz. The calcium signal and movement trajectory in each trial were aligned to movement onset and 3 s-long segments (84 image frames, spanning 0 s–3 s from movement onset) were used as single-trial data for concurrent decoding, while the time window expanded to include neural signals preceding movement in time-delay decoding.

2.2. Input feature normalization

To use different input features (i.e. neurons) with varying dynamic ranges, the input dataset needs to be normalized. We examined four input feature normalization methods, including MinMax scaling (equation (1)), normalization (equation (2)), standard scaling without centering (equation (3)), and standard scaling with centering (equation (4)), which are provided by Scikit-learn [22]. Only the first three normalization methods would maintain the structure of the sparse matrix [22].

$$x^{(i)}_{j,\,\text{normalized}} = \frac{x^{(i)}_j - x^{(i)}_{\min}}{x^{(i)}_{\max} - x^{(i)}_{\min}} \tag{1}$$

$$x^{(i)}_{j,\,\text{normalized}} = \frac{x^{(i)}_j}{\|x^{(i)}\|_2} = \frac{x^{(i)}_j}{\sqrt{\sum_j \left(x^{(i)}_j\right)^2}} \tag{2}$$

$$x^{(i)}_{j,\,\text{normalized}} = \frac{x^{(i)}_j}{\sigma^{(i)}} \tag{3}$$

$$x^{(i)}_{j,\,\text{normalized}} = \frac{x^{(i)}_j - \mu^{(i)}}{\sigma^{(i)}} \tag{4}$$

where $i = 1, 2, \ldots, m$ (m is the number of features), $j = 1, 2, \ldots, n$ (n is the number of data samples); $x^{(i)}_{\max}$ and $x^{(i)}_{\min}$ are the maximal and minimal values of the ith feature, respectively; $x^{(i)}_j$ is the original jth data sample of the ith feature; and $x^{(i)}_{j,\,\text{normalized}}$ is the normalized jth data sample of the ith feature; $\mu^{(i)}$ is the mean value for the ith feature and $\sigma^{(i)}$ is the standard deviation for the ith feature.

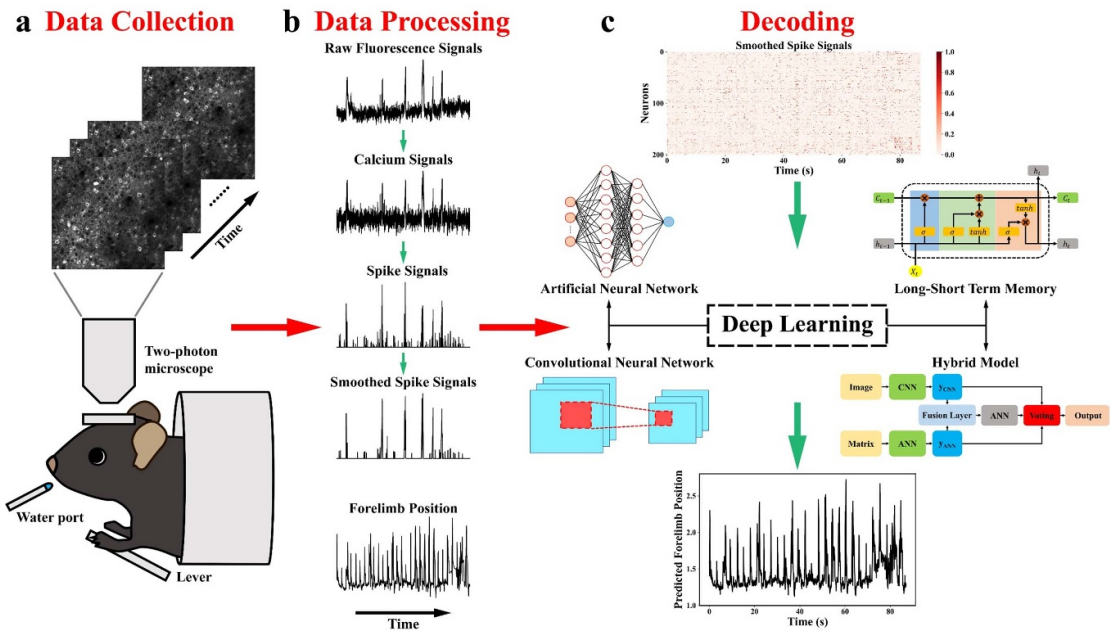

Figure 1. Deep-learning-based neural decoding of mouse forelimb movements. (a) Neural and movement data in this paper were collected while mice learned a forelimb lever-press task [3]. The neural activity of motor cortical neurons was measured via two-photon calcium imaging, and the endpoint of the forelimb was measured using the position sensor of the lever. (b) The raw calcium fluorescence signal was converted to smoothed spike signal to use as input to decoders. The forelimb position was used as the output variable of the decoders. (c) Deep learning is applied to several neural networks that decode forelimb movement trajectories from neural activity.

We present results using MinMax scaling in the paper as we found that MinMax scaling produces the best decoding performance in the concurrent decoding scheme.

2.3. Dataset splitting

To train and evaluate decoding models, the dataset was divided into three subsets: training, validation, and test. To do so, the whole dataset is first shuffled in the dimension of trials, 90% of all trials was designated as training and validation dataset [22, 23]. The remaining 10% was designated as the test dataset. Trials in the training and validation dataset were divided in a 5-fold cross-validation method so that we fit a given model on the training trials and selected the parameters that yield the highest decoding performance score (i.e. R^2) on the validation trials. Then, these fit parameters were used to evaluate the model on the trials in the test dataset.

2.4. Optimization of models

ANN and LSTM with different hyperparameters and parameters were separately optimized using a common loss function, the mean square error (MSE). We used the adaptive moment estimation (Adam) [24] as the optimizer and the R^2 of the test dataset as the criteria when ranking models. All models were trained and evaluated by 5-fold cross-validation in PyTorch [25] and Skorch [26].

$$\mathrm{MSE}(y,\hat{y}) = \frac{1}{n}\sum_{k=1}^{n}(y_k-\hat{y}_k)^2 \tag{5}$$

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{k=1}^{n}(y_k-\hat{y}_k)^2}{\sum_{k=1}^{n}(y_k-\bar{y})^2},\ \bar{y} = \frac{1}{n}\sum_{k=1}^{n}y_k \tag{6}$$

where $\hat{y}$ is the predicted value, y is the true value, $\hat{y}_k$ is the predicted value of the kth sample, and y_k is the true value of the kth sample.

2.5. Statistical test

To infer the statistical significance of differences in decoding performance among different models within a session, the 5 × 2cv paired t-test [27] was used. In the 5 × 2cv paired t-test, the dataset is split into two parts (50% training and 50% test data) 5 times. In each of the five iterations, two models are trained by the training data and evaluated by the test data (their performance: p_A and p_B) and then the training and test data are rotated (the training data becomes the test data and vice versa) to evaluate models again [28]. The two performance differences ($p^{(1)}$, $p^{(2)}$) are measured as equation (7). The mean ($\bar{p}$) and variance (s^2) of the difference are computed as equation (8). The variance of the difference is computed for five iterations and then used to compute the t-statistic as equation (9). Using the t statistic, the p-value can be computed and compared with a predetermined significance level, 0.05 [28].

$$p^{(1)} = p_A^{(1)} - p_B^{(1)},\ p^{(2)} = p_A^{(2)} - p_B^{(2)} \tag{7}$$

$$\bar{p} = \frac{p^{(1)}+p^{(2)}}{2},\ s^2 = (p^{(1)}-\bar{p})^2 + (p^{(2)}-\bar{p})^2 \tag{8}$$

$$t = \frac{p_1^{(1)}}{\sqrt{(1/5)\sum_{i=1}^{5} s_i^2}} \tag{9}$$

where $p_1^{(1)}$ is the p_1 of the first iteration.

When comparing performance between sessions within a given model or between models across multiple sessions, we used the standard Wilcoxon rank-sum test using the five cross-validated performance values per session. The p-value is adjusted for multiple comparisons using the Benjamini–Hochberg false discovery rate (FDR) correction.

2.6. Image reconstruction for convolutional neural network (CNN)

Four types of data were used to reconstruct images, including smoothed spike signals, the neuron types (excitatory or inhibitory), the distance matrix of neurons, and the neuronal connectome. The neuronal connectome was indirectly inferred using the fast and robust connectome inference (FARCI) method [29]. The distance matrix of neurons indicates the relative locations of neurons. The MinMax Scaling normalized smoothed spike signals were used to reconstruct images. In the reconstructed image, the neuron was plotted as the point. To distinguish different types of neurons, the excitatory neurons and inhibitory neurons were plotted as red and blue, separately. The brightness of color represented the activity level ranging from 0 to 1. The lines between points represent the functional connectome and the thickness of each line represents the strength of connectivity between the two corresponding neurons. The reconstructed images were resized as 64 × 64 or 256 × 256 pixels and then normalized with their mean and standard deviation in PyTorch.

2.7. SHapley Additive exPlanations (SHAP)

Although deep learning models have excellent accuracy, mechanistic understanding of these models is often not straightforward [30, 31]. To characterize the input features that significantly contribute to the predictions of our decoding models, we used SHAP value proposed by Lundberg *et al* [30–32]. SHAP value is based on Shapley values from the game theory and additional feature attribution methods [30–33]. Shapley values are the only possible method in the large category of additive feature attribution methods that will meet three crucial properties at the same time: local accuracy (known as 'efficiency' in the game theory), consistency (known as 'monotonicity' in the game theory), and missingness (similar to 'null effects' in the game theory) [30–32].

The Shapley values are computed as the following:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!\,(M - |S| - 1)!}{M!} \left[f_x(S \cup \{i\}) - f_x(S) \right] \tag{10}$$

where F is the set of all features (M features), S is the subset of F, and $S \subseteq F \setminus \{i\}$ means all possible feature subsets. f_x is a conditional expectation function of the model's output.

The larger the absolute SHAP value of a feature, the more important the feature is [33]. The SHAP values of features are computed on each data point. The global feature importance is evaluated by the mean absolute SHAP value per feature across all data points, which is determined as [30, 33]:

$$I_j = \frac{1}{n} \sum_{k=1}^{n} \left| \phi_j^{(k)} \right| \tag{11}$$

where n is the number of data points.

3. Results

In this paper, we sought to decode movement trajectories from neural signals in the motor cortex, using data from a published study [3] in which the activity of hundreds of layer 2/3 neurons in the motor cortex was recorded while mice learned a forelimb lever-press task over two weeks (figure 1(a)). The neural activity was acquired via two-photon calcium imaging, and the forelimb position was measured through a position sensor on the lever as previously described [3]. We used movement and neural data from 14 recording sessions of a single mouse. The original neural data were the calcium fluorescence signals, which were subsequently transformed into fractional changes relative to the baseline ($\Delta F/F$). The calcium signals have a long-decay constant that is a property of the sensor rather than the neural signal. To remove the sensor-originating artifacts, we converted the calcium signals to spike signals using a non-negative deconvolution method [34], followed by smoothing [29] (figure 1(b)). Then, the smoothed spike signals were normalized using MinMax Scaling (Methods) as different dynamic ranges across different neurons are undesirable for training decoders.

Figure 1(c) shows the overview of our neural decoding procedure. We applied deep learning to movement decoding using four different network models: artificial neural network (ANN), long-short term memory (LSTM), CNN, and hybrid spatiotemporal network (STN). ANN is a computing system inspired by biological neural networks. It connects the inputs to a sequence of hidden layers, followed by the output. LSTM is a more complicated recurrent neural network that has a hidden state to form memory. Unlike ANN which processes output at a given time using all inputs that are fed simultaneously, LSTM typically uses sequentially fed inputs to produce outputs. Thus, LSTM can flexibly integrate information over time [35]. CNN is a type of deep neural network that is mainly used for analyzing images. Different from ANN and LSTM, CNN has convolutional layers to extract meaningful local structures of images

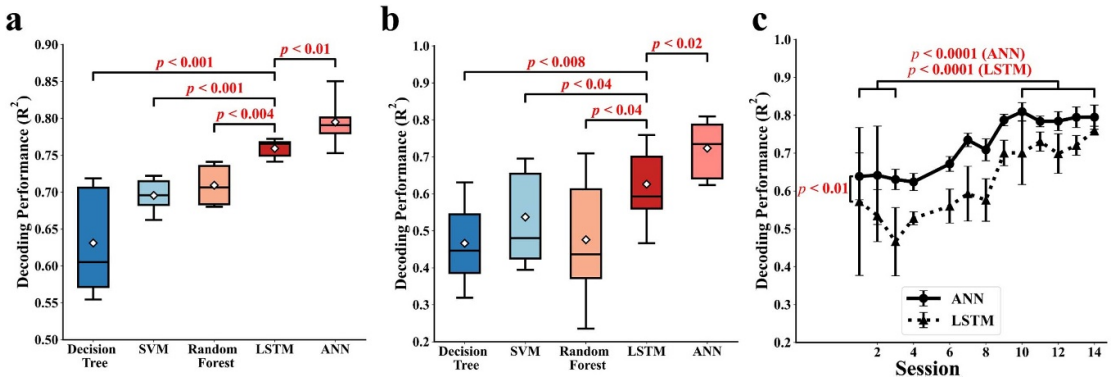

Figure 2. Concurrent decoding. (a) A single session example of decoding performance of five concurrent decoding models. The average and error bar were computed from the 5-fold cross-validation and the statistical test is the 5 × 2cv paired t-test (Methods). Multiple comparisons were corrected using the false discovery rate (FDR) method. (b) The mean decoding performance across all 14 sessions. Wilcoxon rank-sum test using the FDR method to correct for multiple comparisons between models. (c) The decoding performance as a function of the training session for two deep-learning decoders. The error bar was computed from the 5-fold cross-validation. Wilcoxon rank-sum test for comparison between sessions or models.

[35]. As brain imaging technologies (e.g. fMRI, MRI, EEG, and CT) improve, image resolution and volume are increasing, making CNN a crucial and even necessary tool for analyzing the brain state from images [36–39]. Finally, hybrid models including STN combine multiple networks, aiming to process different types of data simultaneously. Several studies utilized hybrid models to infer both the temporal hierarchical features and spatial hierarchical maps of brain networks [40]. Hybrid models have been shown to successfully predict behavioral states (e.g. pre-impact falls for older people) [41].

In the following sections, we describe our deep learning decoders in each of three different decoding schemes: concurrent, time-delay, and spatiotemporal.

3.1. Concurrent decoding

Concurrent decoding is the simplest scheme in this paper, in which neural signals at a given time are used to decode the position of the forelimb movement at that time [20]. In the concurrent decoding scheme, we compared the performance of two deep learning network models, ANN and LSTM, to three traditional machine learning models, Decision Tree, Support Vector Machine (SVM), and Random Forest (RF). The ANN and LSTM were optimized through numerous combinations of hyperparameters and 5-fold cross-validation. Hyperparameters include the number of nodes in the hidden layers of the ANN, dropout rates of the ANN and LSTM, and the number of features in the hidden state of the LSTM.

We found that both the ANN and LSTM perform significantly better than the three traditional machine learning models (figures 2(a) and (b)). The ANN showed a slightly better performance than the LSTM. In addition, the computational time for model optimization was shorter for the ANN than for LSTM. The outperformance of the ANN over traditional machine learning models might be attributed to various factors. Compared to traditional machine learning models, it has been shown that ANN has a more powerful nonlinear-fitting capacity [42] which enables them to adaptively extract more crucial information from input features. Feature extraction would be also important in our neural decoding as the number of input features is much larger than that of output features, such as 200 input features versus one output feature. In addition to a large number of neurons, the noise of neuronal spike signals is a significant factor affecting the accuracy of neural decoding. Although extracting spike signals from calcium signals could reduce sensor-specific artifacts, spike signals still have intrinsic and measurement noise which could deteriorate the performance of neural decoding. Unlike traditional machine learning models, ANN shows stronger robustness and fault tolerance to noise [42]. Considering the high accuracy, low computational cost, and other aforementioned advantages, deep learning ANN is an efficient model to perform concurrent decoding.

As described earlier, the neural and movement data were recorded while the mouse was trained to perform a cued lever-press task daily for two weeks (sessions 1–14) [3]. The previous study found that neural population activity across trials with similar movement trajectories becomes more similar at the expert stage (sessions 10–14) than the naïve stage (sessions 1–3), indicating that a more consistent relationship between movements and neural activity emerges with learning. These observations suggest that neural decoders might perform better at the expert than the naïve stage. To test this idea, we examined the decoding performance of the ANN and LSTM as a function of learning stages (i.e. sessions). As shown in figure 2(c), for both the ANN and LSTM, the overall performance tends to increase, albeit with a few fluctuations over sessions. Accordingly, compared to the naïve stage, both the ANN and LSTM showed significantly higher performance in the expert stage, confirming our prediction. Taken

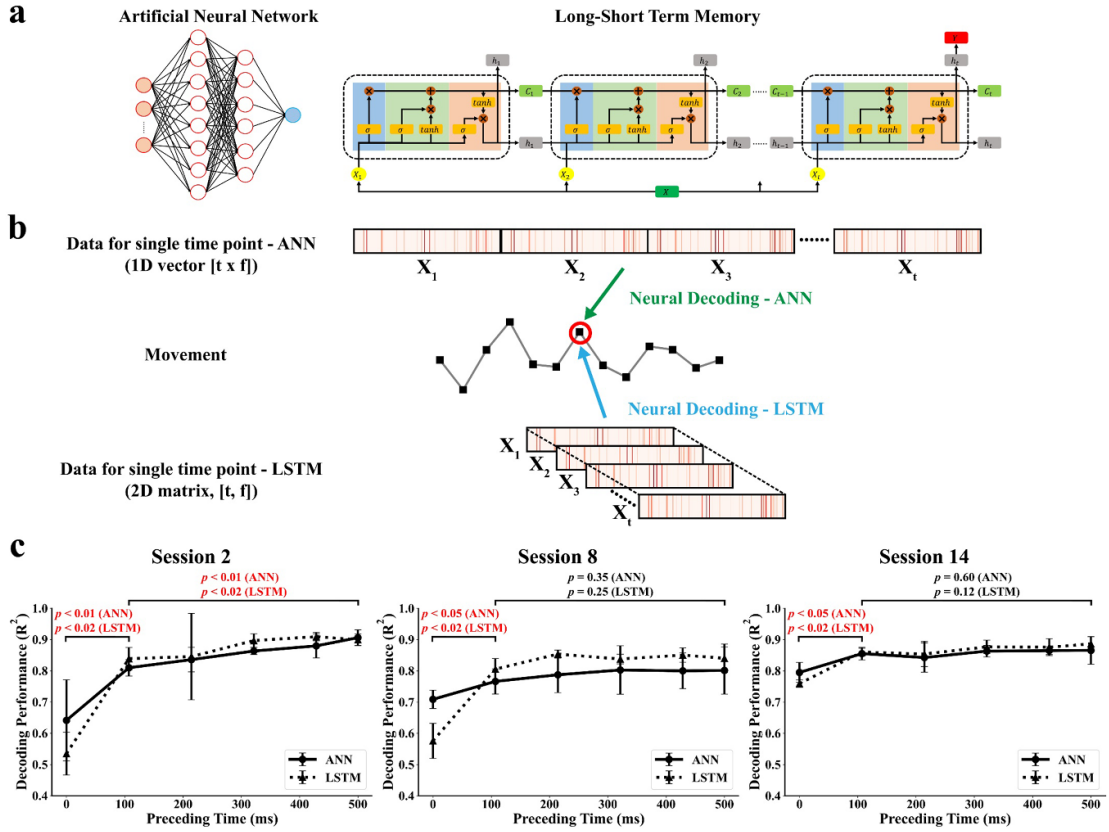

Figure 3. Time-delay decoding. (a) Architectures of ANN (non-recurrent decoder) and LSTM (recurrent decoder) for time-delay decoding. (b) The structure of input and output data for time-delay decoding. To decode the movement position at a given time point, the neural activity in all time points in the preceding time interval (t time points) from all neurons (f neurons) was used. The input features were concatenated into a 1D vector to be fed into ANN all at once, while the input features at different preceding timepoints are fed sequentially into LSTM. (c) The decoding performance of ANN and LSTM as a function of preceding time intervals in three different training sessions. The error bar from the 5-fold cross-validation. Wilcoxon rank-sum test using the FDR method to correct for multiple comparisons between time intervals.

together, the deep learning concurrent decoders outperform the traditional machine learning algorithms, but their performance is still affected by the dynamic change in the temporal relationship between neural activity and movements over time.

3.2. Time-delay decoding

There are conduction and processing delays for neural signals in the motor cortex to reach muscles and thus cause movements. In addition, even if preceding neural signals are not necessarily the direct command to control muscles, they can still carry nontrivial information regarding the upcoming movements [43]. Therefore, the decoding accuracy of movement trajectories may be further enhanced by including neural signals at time points preceding each movement position to decode [20, 35]. To test this idea, we trained deep learning models in a second decoding scheme, time-delay decoding in which neural signals preceding each movement time point are incorporated. The preceding time of neural signals suggested for neural decoding varies across different studies, ranging from 300 ms to more than 1 s [3, 44, 45]. Given this variability, we examined six different time intervals prior to the movement time point, 500, 429, 321, 214, 107, and 0 ms. Note that 0 ms interval corresponds to concurrent decoding. ANN (figure 3(a)) is a non-recurrent network with no persistent internal state (memory) and thus, all input data for decoding the movement position at a given time should be fed to the network at the same time. So we concatenated the time series of spike signals from all time points in the preceding time interval across all neurons as a single vector. In contrast, LSTM is a recurrent network (figure 3(a)) that can exhibit temporally dynamic behavior with the persistent internal state or memory. The hidden memory state can be used as contextual information to improve decoding accuracy [46]. As the LSTM architecture allows 'multi inputs and one output', the spike signals from each time point in the preceding time interval can be sequentially fed to the network (figure 3(b)).

We found that in all sessions, time-delay ANN and LSTM decoders incorporating neural activity in the 107 ms long preceding time interval improved decoding accuracy compared to the concurrent decoders, indicating that this previous time interval carries significant movement-related information. During the first few sessions of motor learning, there was a tendency that decoding accuracy further

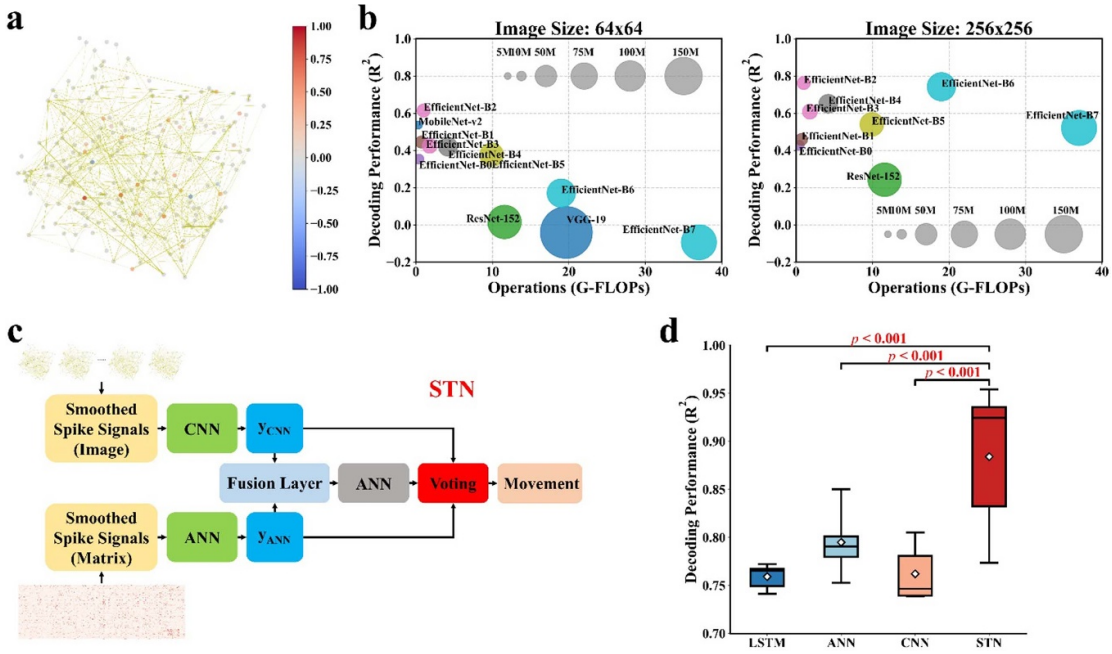

Figure 4. Spatiotemporal decoding. (a) An example reconstructed image of neural activity and connectivity. The color code of each dot represents the activity of neurons at the corresponding location, and the thickness of lines represents the strength of connectivity between the corresponding neuronal pairs. (b) Ball chart reporting the decoding performance (R^2) versus computational complexity (floating-point operations (FLOPs), G-FLOPs $= 1 \times 10^9$ FLOPs) across different CNN models for two sizes of images, respectively. The size of each ball corresponds to the number of parameters for each CNN model. Models with an R^2 less than -0.2 are not shown in the plot. (c) Scheme of the proposed hybrid model, STN. The best ANN (green box) and CNN models in the concurrent decoding scheme are used. In this hybrid model, we only optimized the architecture of the ANN (gray box) receiving the fusion layer output. (d) The decoding performance of four different decoders in session 14. The error bar from the 5-fold cross-validation. The 5×2cv paired t-test for comparisons between models and multiple comparisons were corrected using the FDR method.

increased by including more distant preceding time points (figure 3(c)). The best preceding time interval (corresponding to the highest R^2 of the test dataset) in the early sessions varied across different sessions for both the ANN and LSTM (figure 3(c)). In contrast, in the later sessions, increasing the preceding time interval beyond 107 ms did not significantly affect the decoding accuracy of both the ANN and LSTM (figure 3(c)). This difference between the early and late sessions is consistent with the published observation [3] that movement-related neurons show variable timing of activity on individual trials relative to the movement onset in the naïve stage, whereas movement-related neurons in the expert stage showed more reproducible and stable activity timing, closer to the movement onset.

As the time-delay decoding outperforms the concurrent decoding in the early sessions, it produces consistently high performance across different sessions (R^2 above 0.7) for both the ANN and LSTM. Thus, the time-delay decoding can robustly extract movement-related information from neural signals even if the temporal relationship between movements and neural signals dynamically changes from session to session. It is also notable that the concurrent ANN outperforms the concurrent LSTM, but the time-delay ANN and LSTM perform similarly (figure 3(c)). In other words, performance gain by changing from concurrent to time-delay decoding is larger in the LSTM than in the ANN.

3.3. Spatiotemporal decoding

Both the concurrent and time-delay decoding examined so far extract movement information from the time series of input features (i.e. spike signals), but do not utilize the spatial relationship or connectivity between input features. It is known that neurons interactively communicate to give rise to skillful movement, and the inter-neuron interaction is shaped by the underlying neuronal connectome. A promising avenue for further enhancing the movement decoding performance may be to add a decoder that can mine movement-related information from the spatial relationship between neurons and their functional connectome. The neural data in this paper was acquired from calcium imaging which provides the spatial information of all recorded neurons. Thus, we tested a hybrid spatiotemporal decoding scheme that combines the aforementioned ANN decoders with a decoder that extracts movement trajectory information from reconstructed images of neural activity and connectivity. In reconstructed images, each neuron was represented as a dot positioned at its center of mass and the brightness of the colored dot indicated the activity level (i.e. normalized smoothed spike signal) of the

corresponding neuron (figure 4(a)). The connectivity between a pair of neurons (i.e. connectome) was represented by the thickness of the line connecting the two dots corresponding to the two neurons. There are various connectome inference approaches (Model-based Methods and Model-free Methods [47]). In this paper, we used a recently developed method, the FARCI method [29] which can be effortlessly applied to the data obtained from calcium imaging and infer the strength of association via partial correlations of activity between all pairs of neurons.

As a decoder that extracts the movement trajectory information from reconstructed images, we considered CNN models known to be well-suited for image analysis. We first evaluated the decoding performance of several typical CNN models including VGG-19 [48]; ResNet-34 and -152 [49]; DenseNet-121 and -161 [50]; MobileNetV2 [51]; EfficientNet-B0, B1, B2, B3, B4, B5, B6, and B7 [52]. As input to each CNN model, we compared two input image sizes, 64 × 64 and 256 × 256 (figure 4(b)). We found no systematic relationship between the decoding accuracy and the computational complexity measured as floating-point operations (FLOPs) across different CNN models. EfficientNet-B2 had the highest accuracy in both image sizes (figure 4(b)). The accuracy of EfficientNet-B2 for image size 256 × 256 was up to 0.76 despite its low computational and model complexity. EfficientNet-B5 showed a similar accuracy to EfficientNet-B7 for image size 256 × 256, but it required more than twice as many parameters as EfficientNet-B5 and three times as high FLOPs. EfficientNet-B0 and MobileNetV2 required similar number of parameters and FLOPs, but EfficientNet-B0 was less accurate than MobileNetV2 for image size 64 × 64. There was also no straightforward relationship between the accuracy and image sizes. When the image size increased from 64 × 64 to 256 × 256, the accuracy of several CNN models (e.g. EfficientNets and ResNet-152) improved, while other models such as VGG-19 and MobileNetV2 showed a decline. Due to the lack of a clear relationship, we selected EfficientNet-B2 as our CNN decoder and 256 × 256 as the input image size, based on the empirically observed accuracy (figure 4(b)).

Using the selected CNN model and its input format, we built a hybrid model that combines the aforementioned ANN concurrent decoder and the CNN decoder (figure 4(c)). Hereafter, we refer to this hybrid spatiotemporal model as concurrent STN. In the concurrent STN, the outputs of the two decoders, y_{CNN} and y_{ANN}, were concatenated in the fusion layer and then fed into another ANN. Then, the final predicted movement was the output of the voting layer that computes the mean among the output of the fusion ANN, y_{CNN} and y_{ANN} from the identity shortcuts. As the network depth increases, the accuracy usually becomes saturated and then degrades rapidly due to the shattered gradient problem [49, 53]. The CNN applied in the hybrid models would suffer from the same problem. Identity mapping by short connections is a very powerful tool to fix this problem because the gradients in this architecture are far more resistant to shattering, and decaying sublinearly, while this architecture also adds neither extra parameters nor computational complexity [49, 53]. Based on this idea, we added identity shortcuts (i.e. feeding the ANN and CNN outputs to the voting layer) in the concurrent STN. This concurrent STN was assessed in comparison to three single-network models described earlier, i.e. the concurrent ANN, LSTM, and CNN. As shown in figure 4(d), the STN showed the highest accuracy ($R^2 = 0.88$), followed by the ANN, CNN, and LSTM in that order.

To understand which additional component of the concurrent STN contributes to the enhanced decoding performance, we examined two simpler hybrid models, Model I and Model II, each implementing only part of the concurrent STN architecture. In Model I, we used the average of ANN and CNN outputs as the final output to predict the movement (figure 5(a)). The accuracy of Model I was very close to that of the ANN, indicating little improvement by the added CNN (figure 5(b)). In Model II, we added a fusion layer that combines ANN and CNN outputs and another ANN that takes the fusion layer output (figure 5(a)). Then, the output of this ANN was taken as the final decoding output. The configuration of Model II is closer to the STN but lacks the identity shortcuts and the final voting layer. Despite this small difference, the accuracy of Model II was significantly worse than the STN and only 0.01 higher than that of Model I (figure 5(b)). These results from the two simpler hybrid models suggest that the largest decoding enhancement is attributable to the difference between Model II and the concurrent STN, which is the identity shortcuts. This finding validates our design choice to add the identity shortcuts described earlier.

Next, we examined whether the decoding performance of the STN could be further improved by replacing the concurrent ANN decoder with a time-delay ANN decoder (figure 5(c)). The modified STN will be referred to as time-delay STN. Different from the earlier time-delay decoding analysis of the single-network models, including neural signals in preceding time intervals in the time-delay STN resulted in little improvement in decoding accuracy (figure 5(d)). This result suggests that the concurrent STN that combines the concurrent ANN and CNN reached a saturated level of performance, at least in the expert stage, such that the preceding spike signals in the time-delay decoders to the STN provide little extra information regarding movements.

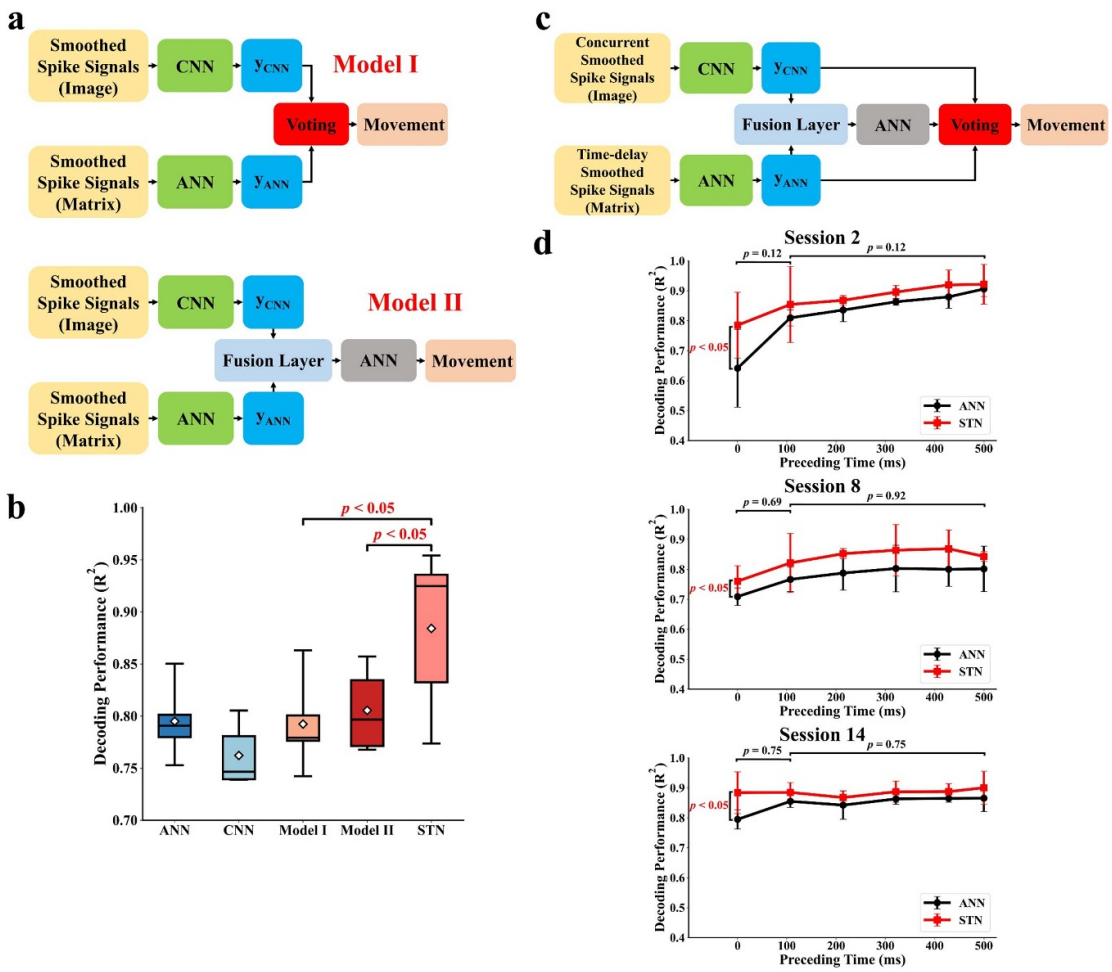

Figure 5. Hybrid model variations. (a) Model I and Model II implement only part of the proposed model, STN. Model I contains no fusion ANN. Model II contains no identity short connections. (b) The decoding performance of ANN, CNN, Model I, Model II, and STN. The error bar from the 5-fold cross-validation. The 5 × 2cv paired t-test for model comparisons and multiple comparisons were corrected through the FDR method. (c) A hybrid model utilizing the time-delay ANN decoder. Note the CNN is the same concurrent decoder as the STN. (d) The decoding performance of time-delay ANN and hybrid models as a function of preceding time intervals in three different training sessions. The error bar from the 5-fold cross-validation. Wilcoxon rank-sum test using the FDR method to correct for multiple comparisons between time intervals.

3.4. Sensitivity analysis

So far we found that deep learning decoders outperform traditional machine learning decoders, and their performance can be further improved by including neural signals from larger preceding temporal windows or adding a network to extract information from reconstructed neural images. The better performance of deep learning decoders likely reflects their superior capability to discover relevant input features. Thus, we identified the input features that played a significant role in our decoders by performing a sensitivity analysis that assesses how important each input feature is to the decoder output. Among several sensitivity analysis methods, we chose SHAP Methods which has been shown to exhibit higher computational performance and better consistency with human intuition [30]. The previous study that generated the dataset in this paper classified movement-related neurons whose activity is significantly higher during movement than non-movement period based on a statistical permutation test [3]. We used these functionally interpretable movement-related neurons as a reference to contrast SHAP-identified important input features in each decoder. To compare important input features across different decoders including the STN which requires a long computational time for SHAP, we applied the sensitivity analysis only to a single expert session.

We first examined important neurons in three concurrent decoders, RF, ANN, and LSTM, and found that the classified important neurons for each decoder were not exactly identical to the movement-related neurons but overlapped to some extent (figure 6(a)). It is noteworthy that even the overlapping neurons show different degrees of contribution across different decoders (figure 6(b)). We also quantified the collective contribution of a subset of neurons to the decoder output using the ratio of the sum absolute SHAP value of the classified important neurons to that of all neurons. In this analysis, we ranked neurons based on their SHAP values and selected top-ranking neurons that match

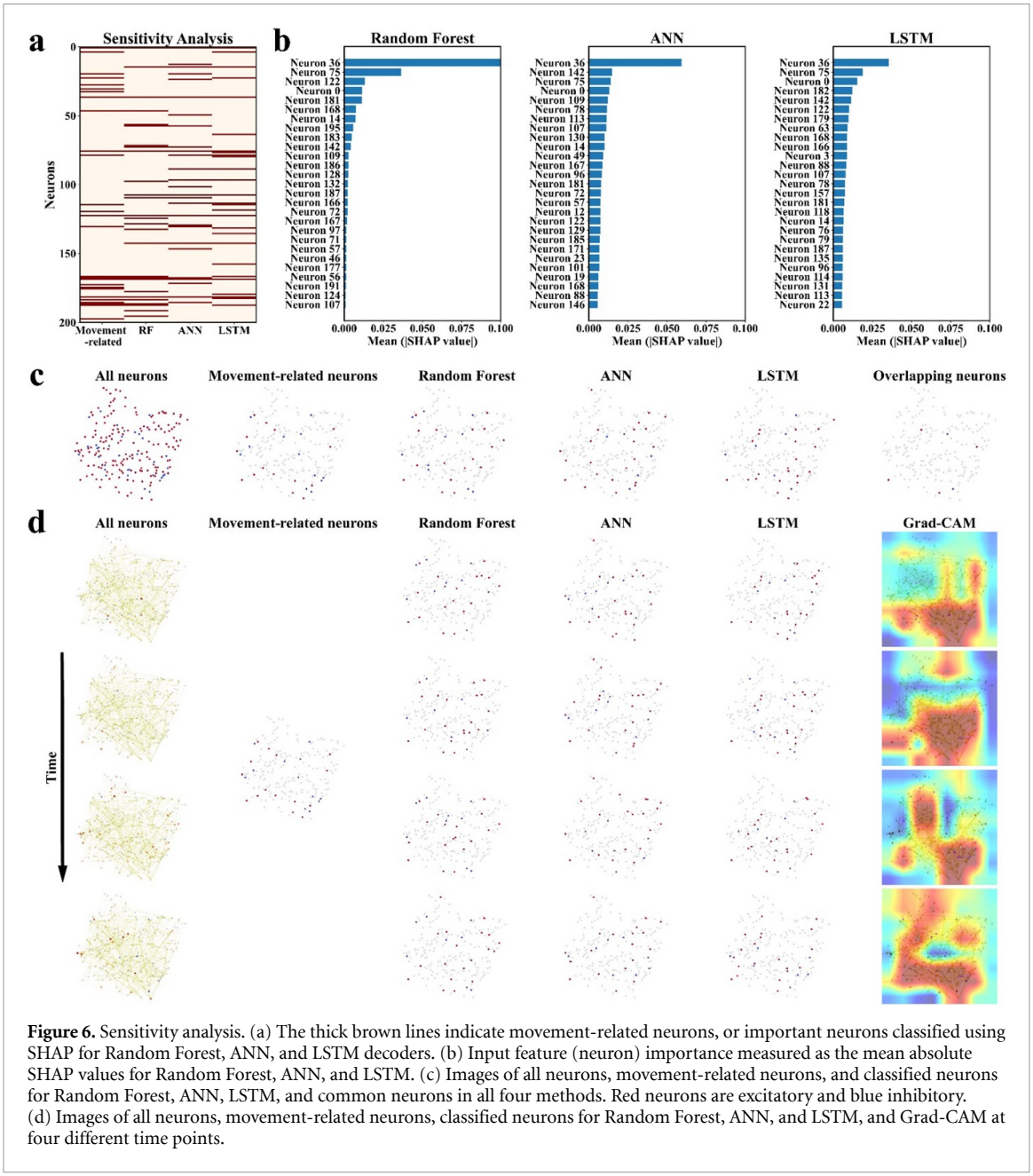

Figure 6. Sensitivity analysis. (a) The thick brown lines indicate movement-related neurons, or important neurons classified using SHAP for Random Forest, ANN, and LSTM decoders. (b) Input feature (neuron) importance measured as the mean absolute SHAP values for Random Forest, ANN, and LSTM. (c) Images of all neurons, movement-related neurons, and classified neurons for Random Forest, ANN, LSTM, and common neurons in all four methods. Red neurons are excitatory and blue inhibitory. (d) Images of all neurons, movement-related neurons, classified neurons for Random Forest, ANN, and LSTM, and Grad-CAM at four different time points.

the number of movement-related neurons. The collective contribution ratio in RF was up to 84%, while the ratios in the ANN and LSTM were 41% and 37%, respectively. These results suggest that deep learning decoders identified more input features than movement-related neurons, which are relevant to movement decoding.

Figures 6(a) and (b) shows the classified important neurons without indicating their locations or cell types (i.e. excitatory or inhibitory neurons). To visualize their spatial distribution and cell types, we created images similar to the reconstructed images used for the CNN decoder (figure 6(c)). The classified important neurons were dispersed across the motor cortex. In the motor cortex, there are more excitatory neurons than inhibitory neurons (e.g. 166 excitatory versus 34 inhibitory neurons in our data sample). However, the proportion of classified important neurons (15 excitatory versus 12 inhibitory neurons) was higher in inhibitory neurons than in the general population, suggesting that inhibitory neurons have a higher tendency to contribute to movement decoding.

Next, to identify the features in the reconstructed image that provide extra information extracted by the CNN in the STN, we computed Grad- class activation mapping (CAM) [54]. Grad-CAM visualizes the class-discriminative features for CNNs at each time point [54]. To facilitate the comparison of the class-discriminative features in Grad-CAM to the important neurons classified in RF, ANN, and LSTM, we used SHAP values at each time point, instead of the global average of SHAP values across all time points, to rank and identify the important neurons

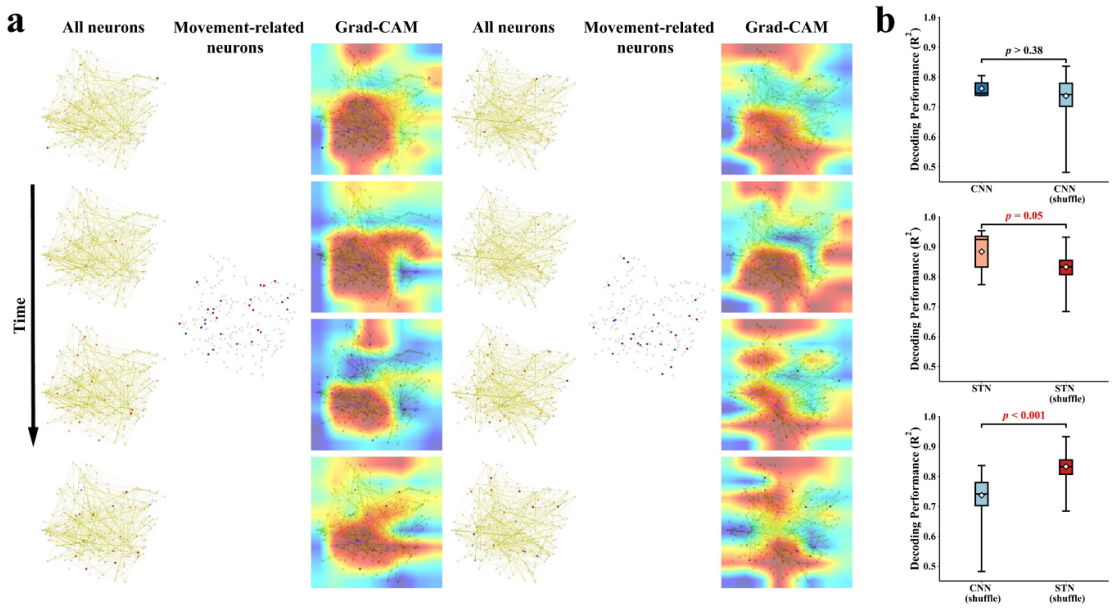

Figure 7. Analysis of the spatial information of neurons. (a) Images of all neurons, movement-related neurons, and Grad-CAM at four different time points (in the same format as figure 6(d)) for two examples of shuffled data. Note, in the shuffled data, the spatial location of each neuron is randomly reassigned in the reconstructed images. (b) The decoding performance of the original and shuffled data. The accuracy of CNN and STN decreased after the spatial information of neurons is disrupted. The error bar from the 5-fold cross-validation. Wilcoxon rank-sum test for model comparisons.

(figure 6(d)). As shown in figure 6(d), only a subset of neurons is consistently classified as the important neurons by RF, ANN, and LSTM. In contrast, Grad-CAM visualizations of smoothed spike signals highlighted a relatively constant area across time. These informative regions identified in the spatial domain that is consistent across time might provide extra information for the STN.

The finding of the spatial regions suggests that some spatial organized patterns might exist in neurons of the motor cortex. Alternatively, movement information decoded by the CNN originates mainly from the activity of individual neurons carrying movement-related information and is not influenced by how those neurons are spatially arranged. To distinguish the two possibilities, we disturbed the natural spatial structure of our neural signals by shuffling the locations of individual neurons while preserving their neural signals and connectomes, and then performed CNN and STN decoding analysis using the shuffled images. If the spatial arrangement is irrelevant, we expect that CNN and STN decoding performance would not be affected by shuffling. We found that for each shuffled data, CNN identified different regions as the class-discriminative features and those regions are relatively consistent across time, similar to the original data (figure 7(a)). However, shuffling did not change decoding accuracy significantly (figure 7(b)). This result suggests that the performance improvement from CNN to STN is unlikely due to the spatial organization of neuronal activity that CNN discovered.

4. Discussion

In this paper, we implemented various deep learning decoders that predict movements from the neural activity in the motor cortex and found that the performance of deep learning decoders surpasses machine learning decoders. Our sensitivity analysis revealed that deep learning discovered a larger number of important input features (i.e. neurons) than traditional decoders or a simple hypothesis test (i.e. movement-related neurons), suggesting that it may have extracted features that are related to movements in complex and nonlinear manners and thus achieved higher decoding accuracy. Taken together, we demonstrated that deep learning could be an effective method for neural decoding of behavior, solely from data with little human intervention.

It is noteworthy that the performance of deep learning decoders could become even better than what we report here, for the following reasons. First, compared to the number of input features (∼200), the sample size of our dataset was relatively small, ranging from 1344 to 8400. Such a small dataset is prone to overfitting for highly complex deep learning models, which deteriorates the performance in test sets. Although some regularization methods such as dropout help to reduce the overfitting issue, the small size of the dataset has forced us to choose deep learning models with relatively low model complexity. Thus, our results may represent lower bounds of performance that deep learning decoders can achieve. In the future, once a large dataset in the source

domain is established, transfer learning may be utilized to efficiently use deep learning models for limited, small datasets [55, 56]. More specifically, deep learning models that are pre-trained using a very large dataset in the source domain could be efficiently fine-tuned for a small dataset in the target domain. This strategy has been successfully applied in neural decoding for the brain-computer interface (BCI) [57, 58]. Secondly, we used the same input features across our deep learning decoding schemes, in order to assess the performance changes solely due to the changes in decoder architectures. In other words, our decoders may be able to produce even more accurate and robust outcomes if feature engineering is performed on each decoder specifically. Thirdly, the STN used the reconstructed image of neural signals rather than the calcium images directly observed from the experiment. Although the reconstruction reduced some background noise, it may have caused some loss of relevant information. It is also noteworthy that for future engineering applications, such as BCI [59, 60], it is time-consuming to use the reconstructed image of neural signals instead of the raw images, increasing the computational load of real-time decoding. Lastly, the architecture of ANN and CNN in our STN model were not optimized in the hybrid setting due to a long computational time. As computational power increases, each network in hybrid models can be better tuned in the hybrid setting, which we expect to further improve decoding accuracy. Despite these caveats, our deep learning decoders still performed better than conventional methods, suggesting that deep learning decoders would excel even further when they are more finely optimized.

Data availability statement

The data generated and/or analyzed during the current study are not publicly available for legal/ethical reasons but are available from the corresponding author on reasonable request.

Acknowledgments

This work was supported by grants from the National Science Foundation (No. 1939987).

ORCID iDs

Fangyu Liu https://orcid.org/0000-0003-0844-563X
Saber Meamardoost https://orcid.org/0000-0002-9491-457X
Takaki Komiyama https://orcid.org/0000-0001-9609-4600
Ying Zhang https://orcid.org/0000-0001-8759-0194
EunJung Hwang https://orcid.org/0000-0003-2786-5963
Linbing Wang https://orcid.org/0000-0003-2670-376X

References

[1] Ethier C, Oby E R, Bauman M J and Miller L E 2012 Restoration of grasp following paralysis through brain-controlled stimulation of muscles *Nature* **485** 368–71
[2] Serruya M D, Hatsopoulos N G, Paninski L, Fellows M R and Donoghue J P 2002 Instant neural control of a movement signal *Nature* **416** 141–2
[3] Peters A J, Chen S X and Komiyama T 2014 Emergence of reproducible spatiotemporal activity during motor learning *Nature* **510** 263–7
[4] Baeg E, Kim Y, Huh K, Mook-Jung I, Kim H and Jung M 2003 Dynamics of population code for working memory in the prefrontal cortex *Neuron* **40** 177–88
[5] Ibos G and Freedman D J 2017 Sequential sensory and decision processing in posterior parietal cortex *Elife* **6** e23743
[6] Hwang E J, Link T D, Hu Y Y, Lu S, Wang E H-J, Lilascharoen V, Aronson S, O'Neil K, Lim B K and Komiyama T 2019 Corticostriatal flow of action selection bias *Neuron* **104** 1126–40. e6
[7] Zhang K, Ginzburg I, McNaughton B L and Sejnowski T J 1998 Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells *J. Neurophysiol.* **79** 1017–44
[8] Davidson T J, Kloosterman F and Wilson M A 2009 Hippocampal replay of extended experience *Neuron* **63** 497–507
[9] Moran D W and Schwartz A B 1999 Motor cortical representation of speed and direction during reaching *J. Neurophysiol.* **82** 2676–92
[10] Paninski L, Fellows M R, Hatsopoulos N G and Donoghue J P 2004 Spatiotemporal tuning of motor cortical neurons for hand position and velocity *J. Neurophysiol.* **91** 515–32
[11] Wu W, Gao Y, Bienenstock E, Donoghue J P and Black M J 2006 Bayesian population decoding of motor cortical activity using a Kalman filter *Neural Comput.* **18** 80–118
[12] Carmena J M, Lebedev M A, Crist R E, O'Doherty J E, Santucci D M, Dimitrov D F, Patil P G, Henriquez C S and Nicolelis M A L 2003 Learning to control a brain–machine interface for reaching and grasping by primates *PLoS Biol.* **1** e42
[13] Wu W, Kulkarni J E, Hatsopoulos N G and Paninski L 2009 Neural decoding of hand motion using a linear state-space model with hidden states *IEEE Trans. Neural Syst. Rehabil. Eng.* **17** 370–8
[14] Brockwell A E, Rojas A L and Kass R E 2004 Recursive Bayesian decoding of motor cortical signals by particle filtering *J. Neurophysiol.* **91** 1899–907
[15] Truccolo W, Eden U T, Fellows M R, Donoghue J P and Brown E N 2005 A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects *J. Neurophysiol.* **93** 1074–89
[16] Srinivasan L, Eden U T, Mitter S K and Brown E N 2007 General-purpose filter design for neural prosthetic devices *J. Neurophysiol.* **98** 2456–75
[17] Yu B M, Kemere C, Santhanam G, Afshar A, Ryu S I, Meng T H, Sahani M and Shenoy K V 2007 Mixture of trajectory models for neural decoding of goal-directed movements *J. Neurophysiol.* **97** 3763–80
[18] Byron M Y, Cunningham J P, Shenoy K V and Sahani M 2007 Neural decoding of movements: from linear to nonlinear trajectory models *Int. Conf. on Neural Information Processing* (Springer) pp 586–95
[19] Livezey J A and Glaser J I 2021 Deep learning approaches for neural decoding across architectures and recording modalities *Brief. Bioinform.* **22** 1577–91

[20] 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
[21] Glaser J I, Benjamin A S, Farhoodi R and Kording K P 2019 The roles of supervised machine learning in systems neuroscience *Prog. Neurobiol.* **175** 126–37
[22] Pedregosa F *et al* 2011 Scikit-learn: machine learning in Python *J. Mach. Learn. Res.* **12** 2825–30
[23] Stone M 1974 Cross-validatory choice and assessment of statistical predictions *J. R. Stat. Soc.* B **36** 111–33
[24] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)
[25] Paszke A *et al* 2019 PyTorch: an imperative style, high-performance deep learning library *Proc. 33rd Int. Conf. on Neural Information Processing Systems* pp 8026–37
[26] Tietz M, Fan T, Nouri D and Bossan B 2017 Skorch: a scikit-learn compatible neural network library that wraps PyTorch (available at: https://skorch.readthedocs.io/en/stable/)
[27] Dietterich T G 1998 Approximate statistical tests for comparing supervised classification learning algorithms *Neural Comput.* **10** 1895–923
[28] Raschka S 2018 MLxtend: providing machine learning and data science utilities and extensions to Python's scientific computing stack *J. Open Source Softw.* **3** 638
[29] Meamardoost S, Bhattacharya M, Hwang E J, Komiyama T, Mewes C, Wang L, Zhang Y and Gunawan R 2021 FARCI: fast and robust connectome inference *Brain Sci.* **11** 1556
[30] Lundberg S M and Lee S-I 2017 A unified approach to interpreting model predictions *Proc. 31st Int. Conf. on Neural Information Processing Systems* pp 4768–77
[31] Lundberg S M *et al* 2018 Explainable machine-learning predictions for the prevention of hypoxaemia during surgery *Nat. Biomed. Eng.* **2** 749–60
[32] Lundberg S M, Erion G, Chen H, DeGrave A, Prutkin J M, Nair B, Katz R, Himmelfarb J, Bansal N and Lee S-I 2020 From local explanations to global understanding with explainable AI for trees *Nat. Mach. Intell.* **2** 56–67
[33] Molnar C 2020 *Interpretable Machine Learning* (Lulu Company)
[34] Pachitariu M, Stringer C and Harris K D 2018 Robustness of spike deconvolution for neuronal calcium imaging *J. Neurosci.* **38** 7976–85
[35] Livezey J A and Glaser J I 2020 Deep learning approaches for neural decoding: from CNNs to LSTMs and spikes to fMRI (arXiv:2005.09687)
[36] Qureshi M N I, Oh J and Lee B 2019 3D-CNN based discrimination of schizophrenia using resting-state fMRI *Artif. Intell. Med.* **98** 10–17
[37] Acharya U R, Oh S L, Hagiwara Y, Tan J H and Adeli H 2018 Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals *Comput. Biol. Med.* **100** 270–8
[38] Ghafoorian M, Karssemeijer N, Heskes T, van Uden I W, Sanchez C I, Litjens G, de Leeuw F-E, van Ginneken B, Marchiori E and Platel B 2017 Location sensitive deep convolutional neural networks for segmentation of white matter hyperintensities *Sci. Rep.* **7** 1–12
[39] Gao X W, Hui R and Tian Z 2017 Classification of CT brain images based on deep learning networks *Comput. Methods Programs Biomed.* **138** 49–56
[40] Zhang W *et al* 2020 Hierarchical organization of functional brain networks revealed by hybrid spatiotemporal deep learning *Brain Connect.* **10** 72–82
[41] Yu X, Qiu H and Xiong S 2020 A novel hybrid deep neural network to predict pre-impact fall for older people based on wearable inertial sensors *Front. Bioeng. Biotechnol.* **8** 63
[42] Abiodun O I, Jantan A, Omolara A E, Dada K V, Mohamed N A and Arshad H 2018 State-of-the-art in artificial neural network applications: a survey *Heliyon* **4** e00938
[43] Shenoy K V, Sahani M and Churchland M M 2013 Cortical control of arm movements: a dynamical systems perspective *Annu. Rev. Neurosci.* **36** 337–59
[44] Golub M D, Yu B M, Schwartz A B and Chase S M 2014 Motor cortical control of movement speed with implications for brain-machine interface control *J. Neurophysiol.* **112** 411–29
[45] Hira R, Ohkubo F, Ozawa K, Isomura Y, Kitamura K, Kano M, Kasai H and Matsuzaki M 2013 Spatiotemporal dynamics of functional clusters of neurons in the mouse motor cortex during a voluntary movement *J. Neurosci.* **33** 1377–90
[46] Tampuu A, Matiisen T, Ólafsdóttir H F, Barry C, Vicente R and Battaglia F P 2019 Efficient neural decoding of self-location with a deep recurrent network *PLoS Comput. Biol.* **15** e1006822
[47] de Abril I M, Yoshimoto J and Doya K 2018 Connectivity inference from neural recording data: challenges, mathematical bases and research directions *Neural Netw.* **102** 120–37
[48] Simonyan K and Zisserman A 2014 Very deep convolutional networks for large-scale image recognition (arXiv:1409.1556)
[49] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 770–78
[50] Huang G, Liu Z, van Der Maaten L and Weinberger K Q 2017 Densely connected convolutional networks *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 2261–69
[51] Sandler M, Howard A, Zhu M, Zhmoginov A and Chen L-C 2018 MobileNetV2: inverted residuals and linear bottlenecks *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 4510–20
[52] Tan M and Le Q V 2019 EfficientNet: rethinking model scaling for convolutional neural networks (arXiv:1905.11946)
[53] Balduzzi D , Frean M , Leary L , Lewis J , Ma K W-D and McWilliams B 2017 The shattered gradients problem: if resnets are the answer, then what is the question? *Int. Conf. on Machine Learning* (PMLR)
[54] Selvaraju R R, Cogswell M, Das A, Vedantam R, Parikh D and Batra D 2017 Grad-CAM: visual explanations from deep networks via gradient-based localization *Proc. IEEE Int. Conf. on Computer Vision* pp 618–26
[55] Pan S J and Yang Q 2009 A survey on transfer learning *IEEE Trans. Knowl. Data Eng.* **22** 1345–59
[56] Tan C, Sun F, Kong T, Zhang W, Yang C and Liu C 2018 A survey on deep transfer learning *Int. Conf. on Artificial Neural Networks* (Springer) pp 270–79
[57] Schwemmer M A, Skomrock N D, Sederberg P B, Ting J E, Sharma G, Bockbrader M A and Friedenberg D A 2018 Meeting brain–computer interface user performance expectations using a deep neural network decoding framework *Nat. Med.* **24** 1669–76
[58] Makin J G, Moses D A and Chang E F 2020 Machine translation of cortical activity to text with an encoder–decoder framework *Nat. Neurosci.* **23** 575–82
[59] Wolpaw J R, Birbaumer N, Heetderks W J, McFarland D J, Peckham P H, Schalk G, Donchin E, Quatrano L A, Robinson C J and Vaughan T M 2000 Brain-computer interface technology: a review of the first international meeting *IEEE Trans. Rehabil. Eng.* **8** 164–73
[60] Sakhavi S, Guan C and Yan S 2018 Learning temporal information for brain-computer interface using convolutional neural networks *IEEE Trans. Neural Netw. Learn. Syst.* **29** 5619–29