nature human behaviour

Supplementary information

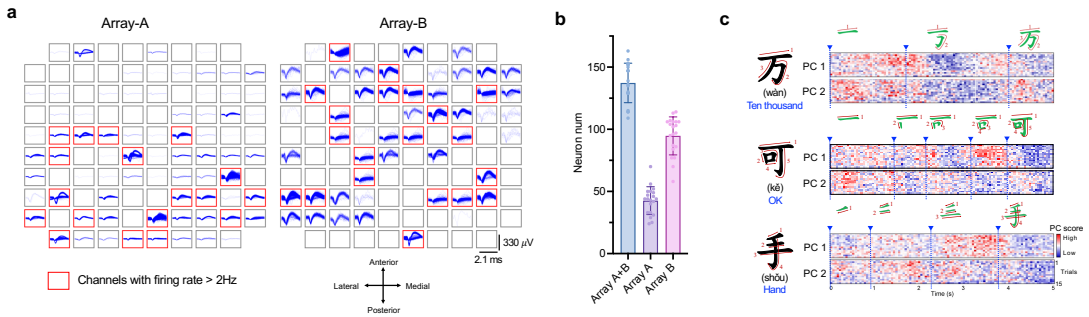
https://doi.org/10.1038/s41562-025-02157-x

Human motor cortex encodes complex handwriting through a sequence of stable neural states

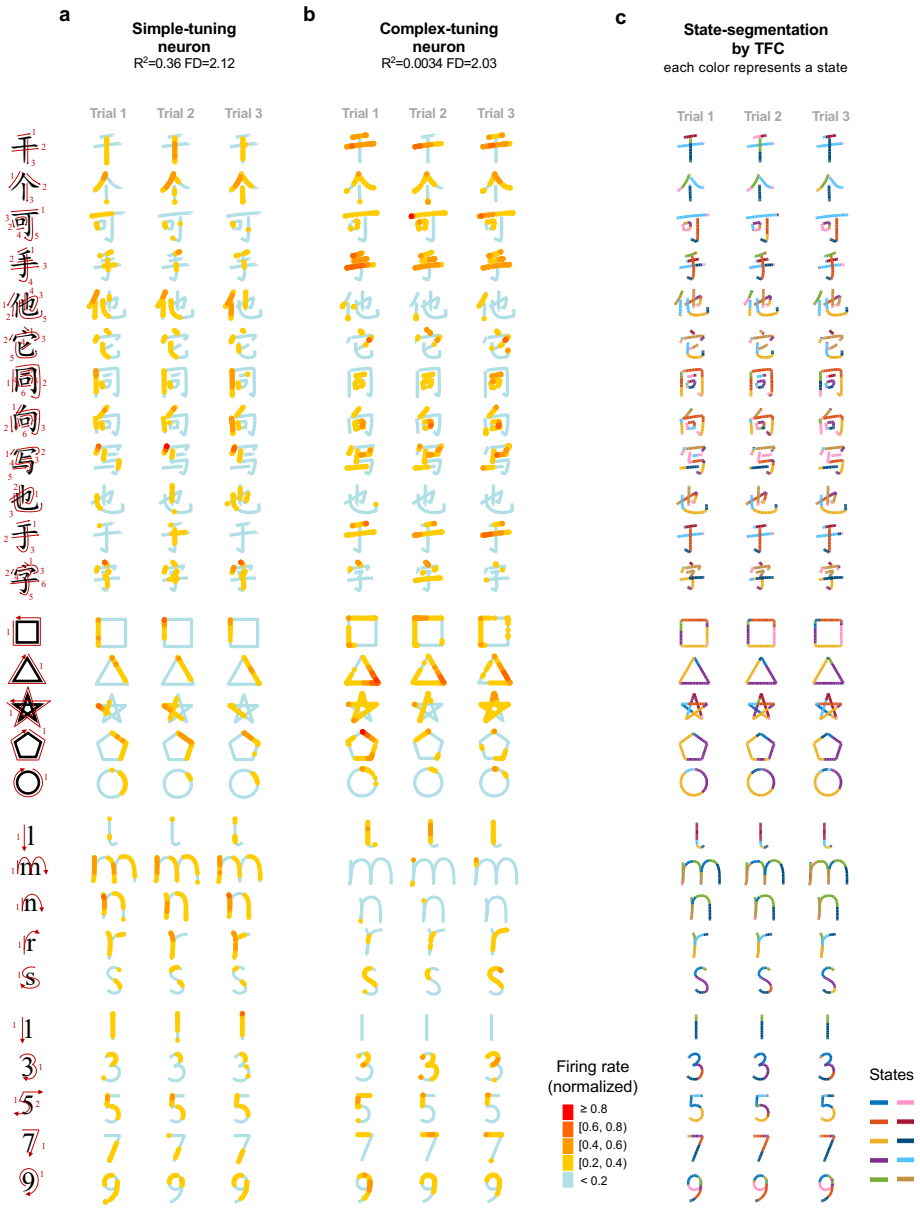
In the format provided by the authors and unedited

Supplementary Information

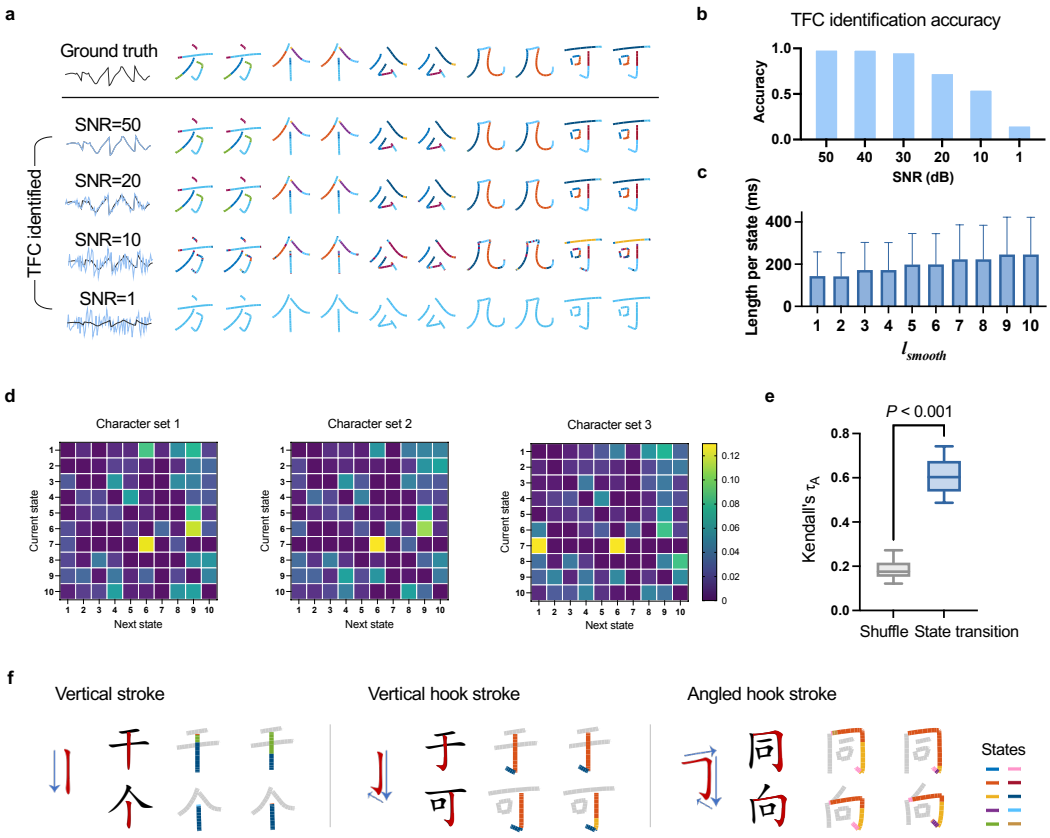
Supplementary Figures



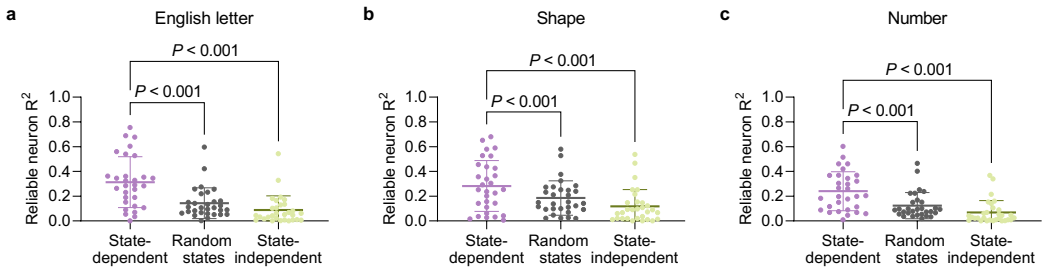
Supplementary Fig. 1 | Example of neural signal recordings and the neural representation of handwriting. a, Illustration of spiking activity recorded from each microelectrode array during a 3-minute window, captured on the 1402nd day after array implantation. **b**, Statistics of single unit numbers for both arrays across 20 sessions (M±SD). There were 137.3±15.8 units after offline spike sorting with Array A and B together (with 42.6±11.2 for Array A and 94.7±15.2 for Array B, n = 20 sessions). **c**, Neural activity in the top 2 principal components is shown for three example characters, 15 repetitions for each character. The color scale is normalized within each panel separately for visualization.



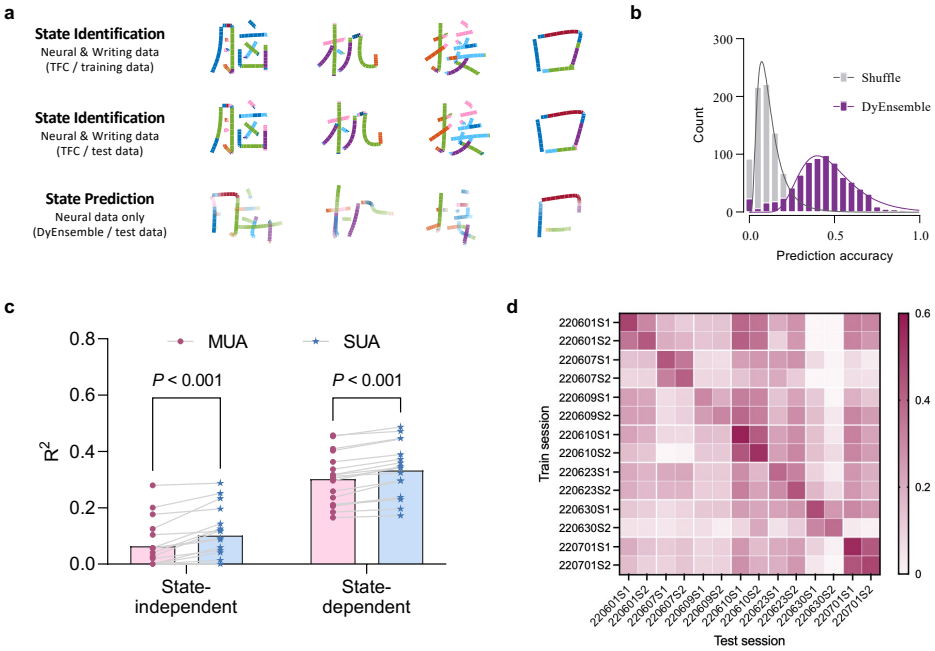
Supplementary Fig. 2 | Examples of neuronal responses and state segmentation during handwriting of different contents. a, Illustrative examples showcasing the neural response of a simple-tuning neuron. The neuron demonstrates a consistent preferred direction of down left, while it only selectively fires at part of down left trajectories. This neural activity can be explained by a state-dependent model, where each state shares a consistent preferred direction with diverse modulation depth. **b**, Similar to **a**, but with a complex-tuning neuron, this neuron exhibits divergent preferred directions and fires in both leftward, rightward, and downward trajectories. This neural activity can be explained by a state-dependent directional tuning model where the preferred direction shifts across states. **c**, State-segmentation results with the TFC method across different characters in three repetitions.



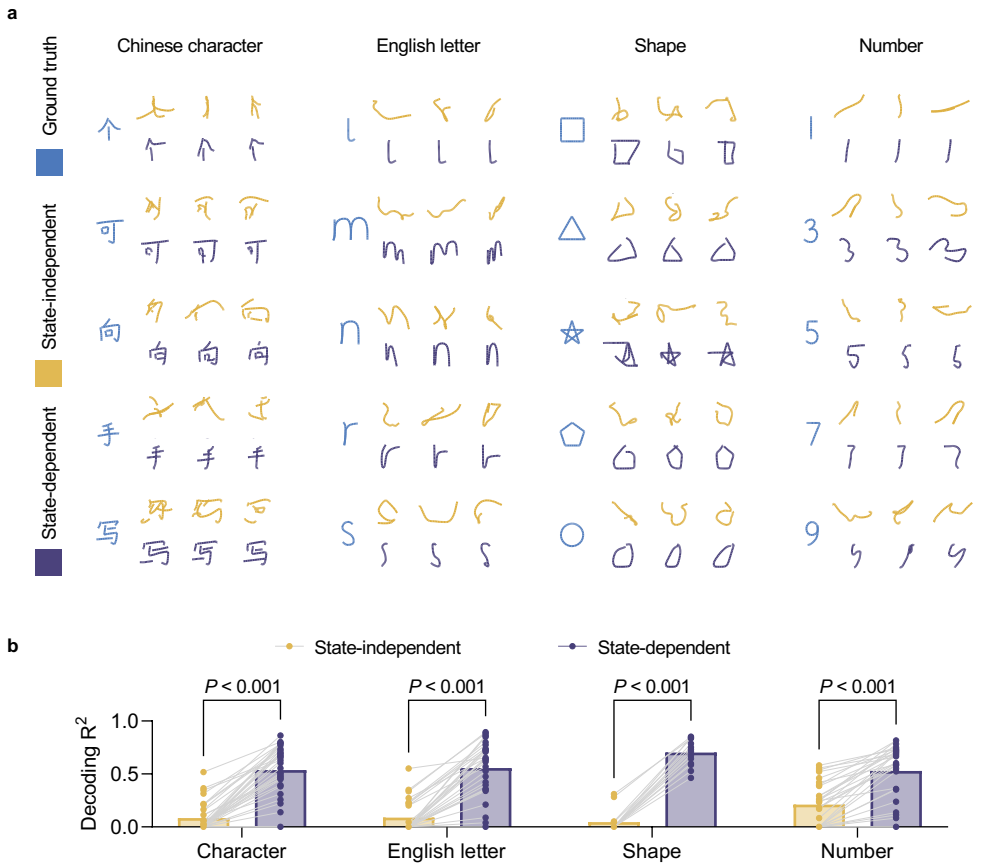
Supplementary Fig. 3 | Evaluation of the TFC algorithm in writing state identification. a, Simulation experiment assessing the TFC algorithm's efficacy in correctly identifying distinct mode (state) and mode (state) switch processes during handwriting (see details in Supplementary Text). **b**, State identification accuracy under varying signal-to-noise ratios (SNR). **c**, Statistics of the length of states with different selections of l_{smooth}. **d**, The state-transition matrices with three non-overlapping character sets (session 220623-S5). Each element (i, j) indicates the state transfer probability from state i to state j. The diagonal values are set to 0. **e**, The similarity of state-transition matrices of each pair of non-overlapping character sets within sessions (n = 18), measured by the Kendall's τ_A value. The Kendall's τ_A values are significantly higher than the shuffle condition (paired two-tailed t-test, $t(17) = 18.21$, $P = 1.373\text{E-}12$, Cohen's d = 6.82), indicating state-transition patterns independent of character sets. **f**, The state sequences during writing three exemplar strokes. We only focused on the highlighted strokes (colored), and the other parts of the characters were left gray. Consistent state sequences were observed when writing the same stroke, regardless of the Chinese character in which the stroke appeared.



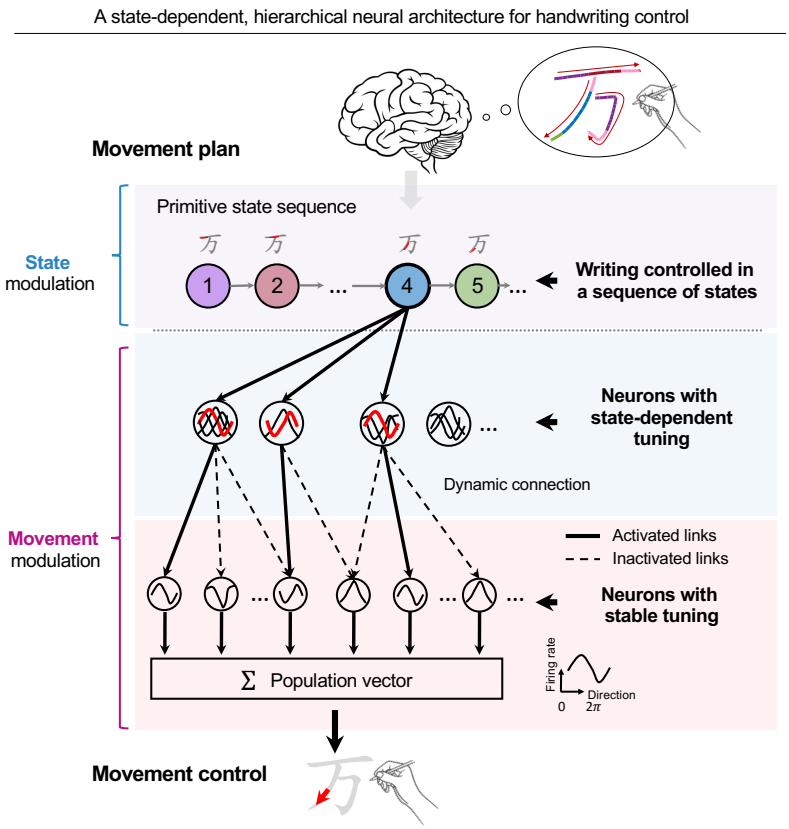
Supplementary Fig. 4 | The state-dependent encoding model better predicts neuronal activities when writing different content. a-c, The state-dependent directional tuning model significantly enhances the R^2 for reliably tuned neurons (n = 30) with the writing of English letters (**a**, State-dependent vs. Random states: paired two-tailed *t*-test, $t(29) = 6.747$, $P = 2.098\text{E-}7$, Cohen's d = 0.997; State-dependent vs. State-independent: paired two-tailed *t*-test, $t(29) = 7.718$, $P = 1.642\text{E-}8$, Cohen's d = 1.348), shapes (**b**, State-dependent vs. Random states: paired two-tailed *t*-test, $t(29) = 6.747$, $P = 2.098\text{E-}7$, Cohen's d = 0.997; State-dependent vs. State-independent: paired two-tailed *t*-test, $t(29) = 7.718$, $P = 1.642\text{E-}8$, Cohen's d = 1.348), and numbers (**c,** State-dependent vs. Random states: paired two-tailed *t*-test, $t(29) = 6.747$, $P = 2.098\text{E-}7$, Cohen's d = 0.997; State-dependent vs. State-independent: paired two-tailed *t*-test, $t(29) = 7.718$, $P = 1.642\text{E-}8$, Cohen's d = 1.348).



Supplementary Fig. 5 | State prediction with the state-dependent decoding model. a, State identification with TFC, with the training set (first line), test set (second line, without model learning, directly applied the learned models during training). The third line illustrates the state predictions in a state-dependent decoding process with the neural data only. **b**, The state prediction accuracy of the state-dependent neural decoder is shown in the histogram (n = 800, 22 sessions with leave-one-repeat-out cross-validation; please see Supplementary Text for details). **c**, Comparison of handwriting decoding performance with single-unit activity (SUA, offline sorted) and multi-unit activity (MUA, unsorted). The mean difference of R^2 was minor (0.33 vs. 0.30 for state-dependent and 0.10 vs. 0.06 for state-independent, n = 18 sessions. State-independent: paired two-tailed *t*-test, $t(17) = 4.453$, $P = 3.49E\text{-}4$, Cohen's d = 0.426; State-dependent: paired two-tailed *t*-test, $t(17) = 9.054$, $P = 6.507E\text{-}8$, Cohen's d = 0.330). **d**, The cross-session/day decoding performance with the state-dependent decoder, with a month time span.



Supplementary Fig. 6 | Comparison of state-dependent and state-independent handwriting decoding with different writing contents. a, Comparison of the decoded handwriting trajectories. **b**, Comparison of the decoding performance with the R^2 (Character: paired two-tailed *t*-test, n = 36, $t(35) = 12.23$, $P = 3.385E\text{-}14$, Cohen's d = 2.232; English letter: paired two-tailed *t*-test, n = 30, $t(29) = 11.38$, $P = 3.229E\text{-}12$, Cohen's d = 2.271; Shape: paired two-tailed *t*-test, n = 15, $t(14) = 23.05$, $P = 1.567E\text{-}12$, Cohen's d = 6.341; Number: paired two-tailed *t*-test, n = 30, $t(29) = 9.012$, $P = 6.614E\text{-}10$, Cohen's d = 1.396).



Supplementary Fig. 7 | Neural processing architecture of handwriting. Sophisticated handwriting is divided into stable states, each corresponding to a stroke fragment. Under each state, movement is controlled through a state-specific neural ensemble, which possibly consists of two types of neurons. The complex tuning neurons encode a fragment of the movement trajectory and are connected to a set of simple tuning neurons, which directly control the movement direction at the current time moment.

Supplementary Text

1 Data collection sessions

We scheduled 2-3 experiment days in a week, and each experimental day contained 2-3 sessions. During a session, the participant sat in a wheelchair in an upright position, with a pillow placed to support his head and neck. His hand rested on a dinner board in front of him. A computer monitor was placed about 1 meter in front of him, indicating which character to write. The experimental sessions used for analysis in this work are listed in Supplementary Table 1. The sessions used in each figure are listed in Supplementary Table 2.

Supplementary Table 1. Data sessions included in this study.

Session	N_C	N_R	Characters	P
220601-S1	30	3	成吃此的定对多法还好和会看来里没那你年如时是我行学永有者这作	
220601-S2	30	3	成吃此的定对多法还好和会看来里没那你年如时是我行学永有者这作	
220607-S1	30	3	把本但当地点动而发分过后话回活间见经开老们身世所同位用知总走	
220607-S2	30	3	把本但当地点动而发分过后话回活间见经开老们身世所同位用知总走	
220609-S1	30	3	爱被常到道得第都高给国果孩家理美面其起前亲使说她西现些样因种	
220609-S2	30	3	爱被常到道得第都高给国果孩家理美面其起前亲使说她西现些样因种	
220610-S1	30	3	不出次从大儿尔方个公几可名女去什生手他天头外为问无也在长中自	
220610-S2	30	3	不出次从大儿尔方个公几可名女去什生手他天头外为问无也在长中自	
220623-S4	30	3	报并场处传风告各更光合何画欢机近决军空拉利论罗妈男品求全任实	CW
220623-S5	30	3	报并场处传风告各更光合何画欢机近决军空拉利论罗妈男品求全任实	
220630-S1	30	3	变表别读房放非该花或结界金觉科苦快连林流命母呢轻师思体条往物	
220630-S2	30	3	变表别读房放非该花或结界金觉科苦快连林流命母呢轻师思体条往物	
220701-S1	30	3	安必边布车达代电东夫关化即加交克吗民目内平却让色声失始受书司	
220701-S2	30	3	安必边布车达代电东夫关化即加交克吗民目内平却让色声失始受书司	
220707-S1	30	3	巴白比产打反父火及记今乐立马片气切认水死四岁太听万王五向写由	
220707-S2	30	3	巴白比产打反父火及记今乐立马片气切认水死四岁太听万王五向写由	
220708-S1	30	3	完先笑信星性许言业医音应友语元员远约月运再早则怎战找至主字坐	
220708-S2	30	3	完先笑信星性许言业医音应友语元员远约月运再早则怎战找至主字坐	
221027-S1	6	15	川万可什公手	CW
230327-S2	8	5	浙江大学脑机接口	SW
230511-S3	32	3	埋坡坏坤环理玻珅权杈杯枓杆种和私科秆汉叹仅扠钗汊呔怯汰钛伏吠沃状	CW
230724-S3	37	3	干个可手他它同向写也于字, 5 shapes, 10 numbers, 10 English letters (k-t)	CW

* N_C: The number of characters
* N_R: The number of repeats

* P: Paradigm

* CW: Single-character writing with visual guidance paradigm

* SW: Sentence writing with visual guidance paradigm

Supplementary Table 2. List of all data collection sessions used for the figures

Figures	Sessions
Fig. 2a, 3f, 5a, 5b, 5c, 5d, 5e, S1b, S3c	220601-S1; 220601-S2; 220607-S1; 220607-S2; 220609-S1; 220609-S2; 220610-S1; 220610-S2; 220623-S4; 220623-S5; 220630-S1; 220630-S2; 220701-S1; 220701-S2; 220707-S1; 220707-S2; 220708-S1; 220708-S2; 230511-S3; 230724-S3
Fig. 2b, 2c, 2d, S2a, S2b, S2c, S3f, S4a, S4b, S4c	230724-S3
Fig. 3b, 3c, 3d, 3e, 3h, 4a, 4b, 4c, 6c, 6d, S3e, S5c, S5d	220601-S1; 220601-S2; 220607-S1; 220607-S2; 220609-S1; 220609-S2; 220610-S1; 220610-S2; 220623-S4; 220623-S5; 220630-S1; 220630-S2; 220701-S1; 220701-S2; 220707-S1; 220707-S2; 220708-S1; 220708-S2
Fig. 3g	220601-S1
Fig. 6b, S5a, S5b, S6a, S6b	230327-S2
Fig. S1c	221027-S1
Fig. S3a, S3b	220610-S1
Fig. S3d	220623-S4

2 Neural representation of handwriting

2.1 The tuning behavior of neurons (Fig. 2a)

To evaluate the tuning behavior of neurons, we assessed it using two metrics: R^2 and FD. A total of 20 experimental sessions were included (please refer to Supplementary Tables 1 and 2), involving 2850 neurons. Note that, since we plotted neurons from different sessions together, it may contain reduplicated neurons appeared at multiple sessions.

The R^2 of one neuron represents the proportion of its neural variance explained by the directional tuning model. The spiking probability P(spike|velocity) is modeled as a linear function of the velocity components along the x and y axes, given by:

$$P(spike|velocity) = b_0 + b_x v_x + b_y v_y \quad (1)$$

where v_x and v_y are the velocity along x- and y-axes, respectively, and b_0, b_x, b_y are the

model parameters. The R^2 was computed by:

$$R^2 = 1 - \frac{\Sigma_k (y_k - \hat{y}_k)^2}{\Sigma_k (y_k - \bar{y})^2} \tag{2}$$

where y_k is the firing rate at time step k of the analyzed neuron, $\hat{y}_k$ is the firing rate estimated by the directional tuning model at time step k, and $\bar{y}$ is the average firing rate of this neuron.

The FD (Fisher's discriminant value) is a metric indicating one neuron's discriminative ability (or reliability) of the tuning patterns by comparing the inter-character distance to the intra-character distance of neural representations, which was defined as:

$$FD_c = \frac{d_{inter}^c}{d_{intra}^c} = \frac{\Sigma_{C_l=c, C_m \neq c} (y_l - y_m)^2 / N_{(l,m)}}{\Sigma_{C_l=c, C_n=c} (y_l - y_n)^2 / N_{(l,n)}} \tag{3}$$

$$FD = \frac{1}{N_C} \Sigma_C FD_C \tag{4}$$

where c denotes a certain Chinese character and N_C denotes the total number of the target characters. The term d_{inter}^c represents the average neural distance between different characters , while d_{intra}^c demotes the average neural distance between different repetitions of the same character. l, m, n are trial indexes and $\Sigma_{C_l=c, C_m \neq c} (y_l - y_m)^2$ refers to the sum over all (l, m) pairs, which satisfy trial l's target C_l is character c while trial m's target C_m is not character c. $N_{(l,m)}$ represents the total number of (l, m) pairs that satisfy this condition. A high FD value indicates a high discriminative ability of neuronal response, namely consistent in repetitions while discriminative with different characters.

2.2 The rasters overlaid on characters (Fig. 2b, 2c, 2d, S2a, S2b)

To plot a raster overlaid on a character, spike counts of the example neurons were first binned into 50 ms bins. Then, max-min normalization was applied to the neuronal firing rate in bins. Next, the sequence of neural responses was plotted on the corresponding trajectory of the character. Here, the handwriting trajectory is the same as the visual instruction in the video. Only strong neural responses above a certain threshold (0.2) were displayed for clearer visualization.

3 Identification of stable states

3.1 Simulation for TFC evaluation (Fig. S3a, S3b)

To validate the reliability of the TFC algorithm, we first conducted experiments on simulated data, as shown in Fig. S3a and S3b. The simulated experiments utilized the character set from 220610-S1 consisting of 30 Chinese characters. Initially, each character was randomly divided into multiple segments, with lengths varying between 5 and 20 bins. Subsequently, we randomly generated $N_{\text{state}} = 10$ encoding models, supposing there are 10 states in the writing process. Each segment was then randomly assigned to one encoding model, and the corresponding neural signals were generated using this model and the kinematic segment. Finally, the neural signals for each character were constructed by concatenating the neural signals generated by different encoding models, with

the addition of Gaussian noise at different levels. Note that we generated three sets of neural signals for each character, with the same model assignments but random noise to mimic the real experimental settings.

Specifically, for each encoding model, we first generated cosine tuning curves for $d_y = 100$ simulated neurons by drawing their preferred directions (PDs) randomly from a uniform distribution over $[0, 2\pi]$. Baseline firing rates were drawn from a uniform distribution over [1, 2.5], and modulation depths were drawn from a uniform distribution over [0.005, 0.025], where these parameter ranges were determined based on observations of real data. Then the encoding model can be constructed as a linear mapping matrix $B \in \mathbb{R}^{N\times(1+d_x)}$, where $d_x = 2$ is the dimension of kinematics (writing velocities along x- and y-axes), 1 denotes the dimension of bias. Here, the first column of the mapping matrix B is the baseline firing rate of each neuron, while the second and third columns of B are the sine and cosine values of neuron PDs, multiplied by the corresponding modulation depth. The above process was repeated 10 times to obtain 10 encoding models. These encoding models allow us to predict the firing rates based on the kinematic inputs.

Different levels of Gaussian white noise were added to the final constructed neural signals, according to the signal-to-noise ratio (SNR):

$$\mathrm{SNR_{dB}} = 10\log_{10}\left(\frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}}\right) = 10\log_{10}\frac{\Sigma_k y_k^2}{\Sigma_k(\tilde{y}_k - y_k)^2} \tag{5}$$

where y_k is the clean neural signals at time step k, and $\tilde{y}_k$ is the neural signals with added noise, which can be obtained directly from the function awgn(.) in MATLAB. In our simulation, we set up 5 noise levels of $\mathrm{SNR}_{dB} = 10, 20, 30, 40, and\ 50$, respectively.

After generating the simulation data, we then used the TFC algorithm to identify these 10 states using $M = 10$ models. The error smooth window size l_{smooth} was 1. In Fig. S3b, the accuracy of model assignment was reported according to:

$$acc_{\mathcal{H}_m} = \frac{\max\left(T_{\mathcal{H}_m,1}, T_{\mathcal{H}_m,2}, \dots, T_{\mathcal{H}_m,j}, \dots, T_{\mathcal{H}_m,N_{\mathrm{state}}}\right)}{T_{\mathcal{H}_m}} \tag{6}$$

$$acc = \frac{\sum_{m=1}^{M} acc_{\mathcal{H}_m}}{M} \tag{7}$$

where $T_{\mathcal{H}_m,j}$ represents the number of overlapping indices between the data selected by model $\mathcal{H}_m$ and the data of state j. $T_{\mathcal{H}_m}$ denotes the length of data selected by model $\mathcal{H}_m$.

3.2 Identifying stable states with TFC (Fig. S2c, S3f)

We used the TFC algorithm to analyze real data and segmented the writing process into stable states. Fig. S2c, S3f shows the examples from 230724-S3. Before applying the TFC algorithm, the spike counts were divided into 200 ms bins using a sliding window with a step of 50 ms. A moving average of 10 bins was then applied for smoothing. The neural signals corresponding to the writing phase were extracted for analysis. The TFC algorithm was configured with $M = 10$ models. The error smooth window size l_{smooth} was set to 5 bins.

3.3 The state-transition patterns (Fig. S3d, S3e)

To assess whether state transitions exhibit temporal dependencies, i.e., whether one state comes after another randomly or some states are more likely to come after a certain state, we divided the data from each session into three non-overlapped subsets and calculated the state transition probabilities for each subset. For each state transition matrix, we removed the diagonal elements that reflect self-transition within a state since they usually contain outlying large values. In S3d, we visualized the state transition matrices for the three subsets (with different character sets each) from an example session.

Subsequently, we computed the pairwise Kendall's τ_A values between the three subsets for all 18 sessions, where τ_A ranges from [-1, 1], with higher values indicating greater similarity between the transition matrices. To obtain the shuffled control, Kendall's τ_A was recalculated after randomly shuffling the transition matrices.

4 State-dependent neural encoding of handwriting

4.1 Neural encoding loss with different model numbers in TFC (Fig. 3b, 3c, 3d, 3e)

To investigate the relationship between the neural encoding capability of the TFC algorithm and the number of specified models, we varied the number of models from 1 to 100 and plotted the resulting loss curve for neural encoding. The experiment included 18 sessions of data, as described in Supplementary Tables 1 and 2. Data were first divided into a training set and a test set, with three-fold cross-validation with no overlapping characters. That is, each session contains 30 characters with 3 repeats, and we divided them into 3 folds, and each fold contains 20 characters for training and the other 10 characters for test. The figure plots the average encoding loss for both the training set and test set with each model number (n = 18 sessions), with the error bar denoting the standard deviation value.

Besides, we also plotted the loss decrease for both the training set and test set, which is calculated from the difference of the loss value. To choose an appropriate number of models, we computed the BIC (Bayesian Information Criterion) value using the following equation:

$$\mathrm{BIC} = -2\ln(L) + k \cdot \ln(n) \tag{8}$$

where L is the likelihood of the model, and k is the number of parameters in the model, n is the number of data points. Since we already have the value of the encoding loss, i.e., the mean squared error (MSE), the only thing we need to do is to calculate the $ln(L)$. Assuming the residuals follow a Gaussian distribution with zero mean, then the variance can be estimated using the mean squared error, MSE. Thus, the log-likelihood can be calculated as follows:

$$\ln(L) = \ln \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-\hat{y}_i)^2}{2\sigma^2}} \tag{9}$$

$$= \sum_{i=1}^{n} \ln \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-\hat{y}_i)^2}{2\sigma^2}}$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} -\ln\sqrt{2\pi\sigma^2} - \frac{(y_i - \hat{y}_i)^2}{2\sigma^2} \\
&= -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\
&= -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{n}{2} \\
&= -\frac{n}{2}\ln(2\pi \cdot \mathrm{MSE}) - \frac{n}{2}.
\end{aligned}
$$

Here, y_i is the ground truth of neural signals, and $\hat{y}_i$ is the neural signals estimated by our models. So the final BIC can be calculated directly from the encoding loss:

$$\mathrm{BIC} = n \cdot \ln(2\pi \cdot \mathrm{MSE}) + n + k \cdot \ln(n). \tag{10}$$

4.2 The pair-wise encoding loss across states (Fig. 3f)

In Fig. 3f, we calculated the encoding loss between each pair of states, namely the models in TFC. Our analysis was based on 20 experimental sessions, where each session involved the collection of at least 30 Chinese characters with 3 repetitions (please refer to Supplementary Tables 1 and 2). We employed the leave-one-repeat-out method, resulting in a total of 60 runs of the TFC algorithm. And the reported loss was the average across these runs.

Here, we calculate each pair of encoding loss $\mathcal{L}(i,j)$ between state i and state j. The computation of $\mathcal{L}(i,j)$ involved encoding the kinematic data selected by model $\mathcal{H}_i$ using model $\mathcal{H}_j$. Subsequently, we calculated the mean squared error between the encoded neural signals and the neural signals selected by model $\mathcal{H}_i$. Then the final loss was averaged over all time bins and neurons. This process can be mathematically expressed as:

$$\mathcal{L}(i,j) = \sum_{n=1}^{d_y}\sum_{k=1}^{T_{\mathcal{H}_i}} \sqrt{\left(Y_{\mathcal{H}_i} - \mathcal{H}_j\left(X_{\mathcal{H}_i}\right)\right)^2}. \tag{11}$$

Here, $Y_{\mathcal{H}_i} \in \mathbb{R}^{d_y \times T_{\mathcal{H}_i}}$ represents the neural signals selected by model $\mathcal{H}_i$, $\mathcal{H}_j(\cdot)$ denotes the encoding function of model $\mathcal{H}_j$, and $X_{\mathcal{H}_i}$ corresponds to the kinematic data selected by model $\mathcal{H}_i$.

4.3 Directional tuning curves in different states (Fig. 2b, 2c, 2d, 5b, 5c)

To closely examine the responses from reliably tuning neurons ($R^2 \geq 0.1$ and FD ≥ 1.4, n = 179) out of 20 experimental sessions, we plotted the tuning curves of these neurons with and without considering the states. Fig. 5b and 5c display four example neurons from session 230724-S3, including two simple-tuning neurons and two complex-tuning neurons.

Regarding the specifics of drawing the tuning curve, the movement kinematics during writing were meticulously partitioned into eight directional intervals (0°~45°, 45°~90°, … , 315°~360°). For each interval, we computed the average firing rate of the neuron at every corresponding time point. The resulting graph illustrates the firing rates for each angle and its subsequent $45°$ range. Furthermore, we applied a cosine function to fit the mean firing rates and visually represented it in the graph.

For each tuning curve, we used the thickness of the curve to reflect the data size; the thicker the curve, the more data points were included for that state. Additionally, we used the color of the curve to represent the performance of the neurons; the darker the color, the higher the R^2 of the neuron within that state.

Fig. 2b, 2c, 2d plot the directional tuning curves without the fitting step of four example neurons from 220324-S2, 220724-S3, and 220724-S3, respectively.

5 State-dependent decoding of handwriting trajectories

5.1 Particle-based estimation for the state-dependent decoder

The DyEnsemble model can be solved using a particle filtering algorithm, which estimates a posterior distribution with a set of particles and their associated weights. Specifically, to estimate $p(x_k|y_{0:k})$, two conditional probabilities of $p(x_k|h_k(\cdot) = \mathcal{H}_m(\cdot), y_{0:k})$ and $p(h_k(\cdot) = \mathcal{H}_m(\cdot)|y_{0:k})$ should be estimated.

Let's consider time step k, where we already have prior information $p(h_k(\cdot) = \mathcal{H}_m(\cdot)|y_{0:k-1}), m = 1, \dots, M$ and a set of particles with weights $\{\omega_{k-1}^i, \chi_{k-1}^i\}_{i=1}^{N_{\text{par}}}$, where N_{par} represents the size of the particle set. Here, χ_{k-1}^i denotes the i^{th} particle with weight ω_{k-1}^i at time $k-1$. The particles can be randomly initialized, or given by the kinematic data from the training set. We used the kinematic data of the training set in this study. The term of $p(x_k|h_k(\cdot) = \mathcal{H}_m(\cdot), y_{0:k})$ can be approximated by:

$$p(x_k|h_k(\cdot) = \mathcal{H}_m(\cdot), y_{0:k}) \approx \Sigma_{i=1}^{N_{\text{par}}} \omega_{m,k}^i \delta(x_k - \chi_k^i). \quad (12)$$

Here, $\delta(\cdot)$ represents the Dirac function, and the term $\omega_{m,k}^i$ is the normalized weight of particle χ_k^i when using the encoding model $\mathcal{H}_m(\cdot)$, and $\Sigma_{i=1}^{N_{\text{par}}} \omega_{m,k}^i = 1$.

The term of $p(h_k(\cdot) = \mathcal{H}_m(\cdot)|y_{0:k})$ can be recursively computed given $p(h_k(\cdot) = \mathcal{H}_m(\cdot)|y_{0:k-1})$ and $p_m(y_k|y_{0:k-1})$, and the particle-based approximation for $p_m(y_k|y_{0:k-1})$ is given by:

$$p_m(y_k|y_{0:k-1}) \approx \Sigma_{i=1}^{N_{\text{par}}} \omega_{m,k-1}^i p_m(y_k|\chi_k^i). \quad (13)$$

Here, $p_m(y_k|\chi_k^i)$ represents the Gaussian likelihood of particle χ_k^i when using the encoding model $\mathcal{H}_m(\cdot)$.

5.2 State prediction accuracy (Fig. S5a, S5b)

To verify the accuracy of the DyEnsemble state predictions, we conducted experiments on 22 sessions. For each session, we first applied three-fold cross-validation to divide it into training and test sets. After the division, we followed three steps:

1. State Identification on the Training Set Using TFC: We used the TFC method to identify states in the training set by inputting neural and writing signals, resulting in state segmentation

(corresponding to the first row in Fig. S5a).

2. State Identification on the Test Set Using TFC: Similarly, the TFC method was applied to the test set using the same inputs (neural and writing signals). The results from this step served as the ground truth for subsequent state predictions (corresponding to the second row in Fig. S5a).
3. State Prediction on the Test Set Using DyEnsemble: The DyEnsemble method was employed to predict states in the test set, this time using only neural signals without writing signals. DyEnsemble calculated the probability of each state based on the current neural signal input, converting these probabilities into state weights. In the figure, colors represent the dominant model at each time point, and the transparency indicates the weight of the dominant model (corresponding to the third row in Fig. S5a).

Fig. S5a is an example of one repeat from session 230327-S2. Fig. S5b shows the statistical results of all repeats across 22 sessions and $M = 10$ states (n = 800). $\text{Hard weight accuracy} = N_{\text{consist}}^{(m)} / N_{\text{GT}}^{(m)}$ is a metric used to measure the accuracy of state predictions, where $N_{\text{GT}}^{(m)}$ is the number of time bins where the m^{th} model is the ground truth, and $N_{\text{consist}}^{(m)}$ is the number of time bins where the m^{th} model is both the predicted dominant model and the ground truth.

Fig. S5b displays the distribution of the accuracy of DyEnsemble's state predictions (in purple) compared to the distribution of accuracy after shuffling the predicted state weights (in grey). The accuracy after shuffling is approximately at the basic level of a 10-class classification, around 1/10, which is significantly different from the accuracy of DyEnsemble's state predictions.

5.3 Visualization of decoded writing trajectories (Fig. 6b, S6a)

We selected the four Chinese characters “脑机接口” (meaning: Brain-computer interface) to illustrate the decoding performance in Fig. 6b. Two decoders were compared, with one state-dependent decoder (DyEnsemble, see Methods for details) and a decoder without considering states (Kalman filter). Initially, the spike counts were grouped into 200 ms bins using a sliding window with a step of 50 ms. To smooth the signals, a moving average with a window size of 10 bins was applied. For decoding, we employed a cross-validation approach known as leave-one-repeat-out. This means that the preprocessed data was divided into training sets and test sets according to the number of times each character was repeated. In each iteration, we used one repeat for test and the remaining repeats for training.

For the DyEnsemble decoder, we first utilized the TFC algorithm to identify $M = 10$ states and their associated models in the training data. The parameter l_{smooth} was set to 10, and the early-stopping threshold β was set to 0.001. Subsequently, we used the dynamic ensemble algorithm to calculate the weights for each model and predict the kinematic states based on the neural signals at each time step during testing. The particles used in the algorithm were unique kinematic states from the training set, and the sticking factor α was set to 0.1.

For the Kalman filter (KF), we employed a velocity Kalman filter with a linear-Gaussian state-space model:

$$x_k = Ax_{k-1} + b + \varsigma_{k-1} \tag{14}$$

$$y_k = Hx_k + \varepsilon_k \tag{15}$$

where k denotes the time step index, $x_k \in \mathbb{R}^{d_x}$ is the kinematic state and $d_x = 2$ is the dimensions of x_k, including the writing velocities in X-axis and Y-axis. $y_k \in \mathbb{R}^{d_y}$ is the neural measurement and d_y is the dimensions of y_k, determined by the number of neurons recorded in this session. $\varsigma_k \sim N(0, \sigma_\varsigma^2)$, $\varepsilon_k \sim N(0, \sigma_\varepsilon^2)$ are state transition noise and measurement noise.

The stroke trajectory was obtained by integrating the decoded velocities to visualize the characters. The start point of each stroke was predefined with the knowledge of the character, i.e., we applied the standard stroke start position with the Kai font of the character.

In Fig. S6a, we used data from the example session 230724-S3, which included four different types of writing content: Chinese characters, English letters, shapes, and numbers. Apart from the differences in writing content, all other decoding settings were consistent with those used in Fig. 6b.

5.4 Details of state-dependent decoding of handwriting (Fig. 6c, 6d, S6b)

To generate the statistical results for Fig. 6c, 6d, the decoding performance of the Kalman filter and the DyEnsemble decoder was evaluated using 18 experimental sessions, consisting of 1620 trials with 270 targets. For Fig. S6b, the decoding performance was assessed using one example session (230724S3), which included different types of writing content: Chinese characters, English letters, shapes, and numbers. Each writing type was decoded separately, using the leave-one-repeat-out test.

The results of Fig. 6c, 6d were offline decoding performance. The decoding was performed within each session using the three-fold cross-validation with no overlapping characters. With the training set, we first used the TFC to learn 10 encoding models, and these models served as the model pool for the DyEnsemble decoder. With the training data and the model pool, the DyEnsemble estimated the parameters for $f(\cdot)$ and $\varsigma_k \sim N(0, \sigma_\varsigma^2)$, and parameters of ε_{v_k}, using the estimation residuals of each encoding model in the pool. With these parameters, DyEnsemble can estimate the handwriting velocity recursively online, given the incoming neural signals.

We evaluated the decoding performance of the strokes in characters. Two evaluation metrics were used for the decoded velocity: root mean squared error (RMSE), and coefficient of determination (R^2). The formulas for these metrics are as follows:

$$\text{RMSE} = \sqrt{\frac{1}{T}\Sigma_{k=1}^{T}(x_k - \hat{x}_k)^2} \tag{16}$$

$$R^2 = 1 - \frac{\Sigma_k(x_k - \hat{x}_k)^2}{\Sigma_k(x_k - \bar{x})^2}. \tag{17}$$

In these formulas, x_k represents the velocity at time step $k \in [1, T]$, $\hat{x}_k$ represents the velocity the decoder estimates at time step k, and $\bar{x}$ is the average velocity of all time steps.

To calculate the final performance of each session, the metrics were averaged over all replicates. In the graph, each data point represents the decoding result of a session (n = 18). To assess the

significance of the decoding performance between the two algorithms, a paired *t*-test was performed, with $P < 10^{-4}$ in terms of RMSE and R^2.

5.5 Decoding performance with SUA and MUA (Fig. S5c)

To assess whether spike sorting step affects the results, we compared the decoding performance between SUA- and MUA-based decoders. In Fig. S5c, the decoding performance (R^2) of both state-independent (Kalman filter) and state-dependent (DyEnsemble) decoders is shown for SUA and MUA data. The dataset and testing protocol were identical to those used for Fig. 6c and 6d.

5.6 Cross session/day decoding performance (Fig. S5d)

To investigate the stability of neural activity over time, we evaluated the decoding performance across sessions spanning one month. Due to fluctuations in the number of recorded neurons across different days, we utilized MUA signals for decoding. In Fig. S5d, the matrix shows the cross-session decoding performance (R^2) using the DyEnsemble decoder. Each element in the performance matrix at position (*i, j*) represents the decoding accuracy when training the model using neural and kinematic signals from session *i*, and testing it on neural signals from session *j*.

5.7 Online decoding settings and delay analysis

We carried out online experiments that decoded the neural signals into handwriting trajectory with the state-dependent decoder in real-time and output the decoded handwriting trajectory to a robotic arm to evaluate the possibility of online handwriting BCIs.

In the online decoding process, neural signals were unsorted, and we used the thresholding approach to detect spike events for each channel. The time bin was 10 ms without overlap for quick response. The online experiment contains a training session and a test session. The training session used the 'sentence writing with visual guidance paradigm' (SW) without output to collect several sentences to train the state-dependent decoder, and then the decoder was deployed for the online decoding process in the second session (SW paradigm, with the robotic arm output). For the demonstration in Supplementary Video 2, the training session was 230327-S1 and the test session was 230327-S2. The detailed training and test contents are listed in Supplementary Table 3. The online decoding R^2 was 0.51 for session 230327-S2.

Supplementary Table 3. Training and test sessions for online demonstration

Session	N_C	N_R	Characters	P
230327-S1	30	3	今天很开心我在写字浙江大学脑机接口你好他乡还吗明见外面气再	SW
230327-S2	8	5	浙江大学脑机接口	SW

The robotic arm was controlled by position signals (x, y, and z) at each 10 ms, to hold a pen and

write on a whiteboard. Chinese characters contain multiple strokes such that there were pen-lift and pen movements between adjacent strokes. As the verification of the state-dependent encoding model, we only decoded the movements of the stroke writing process and output the trajectories of strokes. Thus, extra information was required to composite strokes into characters: 1) the start and end timestamp of each stroke, and 2) the start position of each stroke on the board. In the proof-of-concept demonstration, to know when each stroke started and ended, for each character, we adopted the timestamps of the start and end for each stroke in the generated trajectory (synchronous to the instruction video). The start position of each stroke was predefined according to the Kai font of the character. During the online decoding process, at the end point of each stroke, the robotic arm automatically lifted the pen and moved the pen tip to the start point of the next stroke.

The computation of neural decoding could be achieved in each 10 ms bins. The computational cost for the state-dependent decoder was 11.8±3E-3 ms per bin with CPU only, and 4.15±2E-4 ms per bin with GPU acceleration. We used GPU acceleration for online experiments. The workstation used for computing was equipped with an AMD CPU (AMD Ryzen 5 5600G with Radeon Graphics, 3.90 GHz) and an NVIDIA GPU (NVIDIA GTX 1080), with 64G of memory.

During the online process, we observed a delay in robotic arm control. The delay was mostly due to the movement of robotic arm between the writing of adjacent strokes. The first delay was caused by the pen-tip lifting process, where after writing each stroke, the robotic arm should pull the pen tip 2 mm away from the whiteboard, move the pen to the start point of the next stroke, and then push the pen tip to the whiteboard again. The pull and push process was additional for the robotic arm, such that it caused a delay (about 300 ms per character). The second delay was the movement of robotic arm from the end of the preceding strokes to the start of the succeeding strokes, which can be uncertain, since the end points of the preceding strokes relied highly on the online decoding results.