Supplementary Material

No Author Given

No Institute Given

1 Supplementary for method

This section included the subsections "Preprocessing with task data" and "Extracting Firing rate"(The missing content in the method section of the paper, due to space limitations) .

1.1 Preprocessing with task data

Due to the presence of high-frequency noise (mainly sensor noise) that inevitably affects the kinematic data during acquisition, the position data was initially sampled at a rate of 250 Hz. Subsequently, a non-causal, 4th-order Butterworth low-pass filter with a cutoff frequency of 10 Hz was applied to the position data to effectively remove sensor noise and restore the true hand movement information as accurately as possible. Velocity and acceleration were computed by taking the first and second derivatives of the position data, respectively. The kinematic data, including position, velocity, and acceleration, were downsampled to match the timescale of the Multi-Unit Activity (MUA) data. Except for the Kalman filter, which utilized all the kinematic state variables (position, velocity, and acceleration), all other decoders only utilized the velocity data. This choice was based on extensive experimental evidence demonstrating that velocity exhibits higher correlation with brain activity in the motor cortex compared to position and acceleration [4, 3].

1.2 Extracting Firing rate

The raw data were band-pass filtered using a causal, 4th-order Butterworth IIR filter with a passband range of 500 to 5000 Hz. Spikes were detected when the absolute value of the filtered signals crossed a threshold (typically set to 3 to 5 times the standard deviation). MUA refers to the collection of all detected spikes on each channel, and the number of spike occurrences within a time window is known as the firing rate. Previous studies have confirmed that a window width of 256ms yields optimal performance for deep learning decoders [2]. For the purpose of data alignment, we have set the window duration to 0.24s with an overlap time of 0.12s.

The method of extracting firing rate using binning is widely used and referred to as the binning method. However, in 2018, Nur proposed a modification to the binning method based on the distribution characteristics of firing rates. This

modification, called Bayesian Adaptive Kernel Smoother (BAKS), incorporates a Gaussian kernel function. The BAKS preprocessing method has been shown to significantly improve the decoding performance of MUA signals. Main equations are as follows:

$$\hat{\lambda}(t) = \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}\hat{h}}(t) \exp\left\{ -\frac{(t-t_i)^2}{2\hat{h}(t)^2} \right\} \tag{1}$$

The expression for $\hat{h}(t)$ is as follows:

$$\hat{h}(t) = \frac{\Gamma(\alpha) \sum_{i=1}^{n} \left[\frac{(t-t_i)^2}{2} + \frac{1}{\beta} \right]^{-\alpha}}{\Gamma\left(\alpha + \frac{1}{2}\right) \sum_{i=1}^{n} \left[\frac{(t-t_i)^2}{2} + \frac{1}{\beta} \right]^{-\alpha - \frac{1}{2}}} \tag{2}$$

It can be observed that BAKS has two parameters, namely the shape parameter (α) and the scale parameter (β). Similar to the binning and FKS methods, the window width of BAKS is adjusted to maximize decoding performance. The shape parameter (α) is set to 4, and the scale parameter (β) is set to $n^(4/5)$, where n represents the number of spikes. The specific details of parameter optimization are described in detail in [1], and will not be further discussed here.

2 Explanation of the formula for calculating kernel size.

This section aims to supplement the part of the original text that lacks a detailed explanation of the kernel size calculation formula in the TQRNN structure due to space constraints.

2.1 Supplementary

The calculation logic of ECA is as follows:

$$\text{Kernel\_size} = \left(\log_2 \text{Channels} + \alpha\right) / \beta \tag{3}$$

α and β are adjustable parameters used to control the kernel size. In this application scenario (Temporal-attention module), 'Channels' refers to the input time steps, denoted as T. Our calculations are shown in the fig. 1. The reason for this design is that the ECA-Net initially only considered scenarios with a large number of channels. They used the log function to keep the convolution kernel size within a reasonable range when the number of channels is large. However, whether in the Channel-Attention module or the Time-Attention module, when the number of channels decline, the importance of individual channels increases, and the amount of information in their latent space also increases. We require a larger value for parameter α to ensure an adequate number of kernels. However, since α has a exponential influence on 'channels', this can lead to a significant increase in the number of kernels when the number of channels is large. Therefore, the following formula is added to make the infulence Smoother:

$$y = \ln\left(1 + \exp x\right) \tag{4}$$

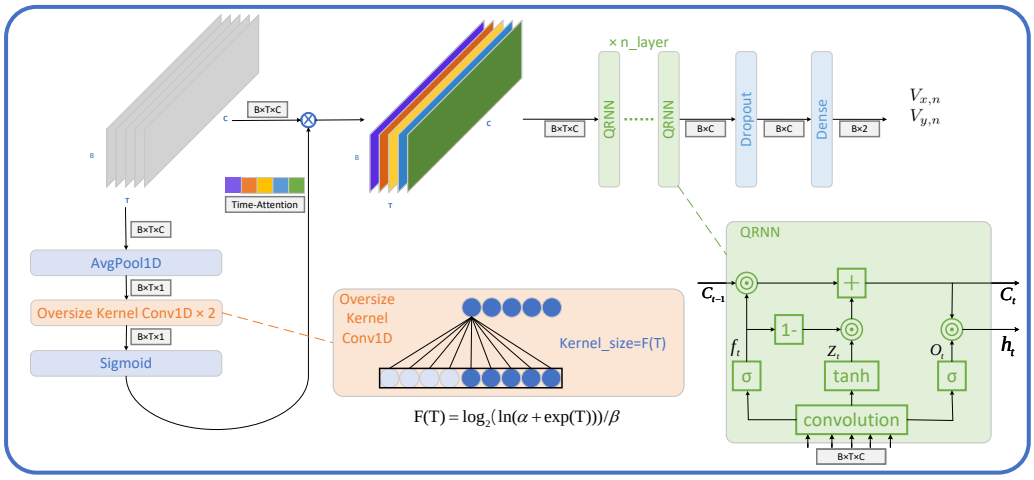

Fig. 1. Illustration of proposed TQRNN. The overall network structure consists of a Temporal attention module, QRNN layers, a Dropout layer, and a Dense layer. The Time Attention module comprises an average pooling layer, two convolutional layers, a sigmoid activation function, and a multiplier.

Combining it with Equation3, we obtain the following kernel calculation formula:

$$\text{Kernel\_size} = \log_2(\ln(\alpha + \exp(\text{Channels})))/\beta \tag{5}$$

Similar to Equation 3, here the 'channels' in the Temporal-Attention module represents T (number of time steps). Parameter α limits the kernel size for lower dimensions, while parameter β and the logarithmic function limit the kernel size for higher dimensions. However, unlike Equation 3, in this formula, The parameter α does not have a exponential influence on 'channels'. Instead, α have a logarithmic influence on 'channels'. This formula differs from the kernel size calculation formula of ECA in that it does not require adjusting parameters α and β when the number of channels changes. Due to space constraints, we have provided a detailed explanation of this in the supplementary materials.

References

1. Ahmadi, N., Constandinou, T.G., Bouganis, C.S.: Estimation of neuronal firing rate using bayesian adaptive kernel smoother (baks). Plos one **13**(11), e0206794 (2018)
2. Ahmadi, N., Constandinou, T.G., Bouganis, C.S.: Robust and accurate decoding of hand kinematics from entire spiking activity using deep learning. Journal of Neural Engineering **18**(2), 026011 (2021)
3. Gilja, V., Nuyujukian, P., Chestek, C.A., Cunningham, J.P., Yu, B.M., Fan, J.M., Churchland, M.M., Kaufman, M.T., Kao, J.C., Ryu, S.I., et al.: A high-performance neural prosthesis enabled by control algorithm design. Nature neuroscience **15**(12), 1752–1757 (2012)

4. Makin, J.G., O'Doherty, J.E., Cardoso, M.M., Sabes, P.N.: Superior arm-movement decoding from cortex with a new, unsupervised-learning algorithm. Journal of neural engineering **15**(2), 026010 (2018)