Decoding upper limb kinematic parameters from motor cortical activity using a deep learning algorithm

Jisung Park[1] and Sung-Phil Kim[1]
[1]Department of Human Factors Engineering, Ulsan National Institute of Science and Technology (UNIST), Ulsan, Republic of Korea
E-mail: teran0729@unist.ac.kr of Jisung Park
E-mail: spkim@unist.ac.kr of Sung-Phil Kim

***Abstract*— The current neural interface technology opens a new window for collecting multi-site streams of the firing activities of hundreds of neurons simultaneously. It provokes a need to equip with the means to harness as much information as possible from such huge amounts of neural data. Deep learning algorithms capable of representing latent components in large data may offer a means to extract useful information related to specific behavioral functions from these neural data. In this study, we aimed to decode movement-related information from motor cortical neuronal ensembles in a primate while the animal moved the arm and hand to perform an eight-target center-out task. The previous studies addressed the problem by decoding the velocity parameter to reconstruct arm-movement trajectories. However, as velocity can be decomposed into speed and direction, it may be advantageous to decode each parameter independently. Thus, we decoded speed and direction of the hand separately with the long short-term memory network (LSTM) to from the ensemble of one hundred fifty-eight primary motor cortical neurons. A comparison of the suggested LSTM decoder with traditional decoders directly predicting the velocity parameter using the linear Kalman filter or LSTM demonstrated a significant increase in the performance of reconstructing 2D hand trajectory. Our results may add accumulating evidence to the employment of deep learning algorithms for intracortical brain-machine interfaces and suggest that speed and direction can be decoded independently.**

I. INTRODUCTION

Brain machine interfaces (BMIs) convert neural signals directly recorded from the brain into intentions such as arm movement or grasping. One of the key concerns in the development of a BMI is to decode neural signals to extract kinematic information related with the movement such as velocity, position and speed of the arm. To achieve this goal, several algorithms such as population vector algorithm, Kalman Filter or optimal linear estimation have been used. Recently, new machine learning algorithms such as long short-term memory (LSTM), which is a nonlinear prediction algorithm, have also been examined for neural decoding [1].

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No. 2017-0-00432, Development of non-invasive integrated BCI SW platform to control home appliances and external devices by user's thought via AR/VR interface)

When decoding the 2D arm movements from motor cortical activity, many decoders have focused on the decoding of velocity [2,3,4]. However, 2D velocity can be decomposed into speed and direction for point-to-point arm movements. Moreover, it was shown that the speed and directional information of arm movements were independently encoded in the motor cortical neurons by linear regression analysis and mutual information measurements [5].

Studies have demonstrated that the arm kinematic parameters might be encoded nonlinearly in the motor cortical ensemble activity. For instance, it was shown that the directional information was encoded nonlinearly in the motor cortical activity based on the von Mises distribution [6]. Also, Li *et al*. showed that a nonlinear quadratic model could capture the neural activity pattern according to the velocity and position of arm movements using the nonlinear unscented Kalman filter (UKF), better than the linear Kalman filter for BMIs [3]. Intuitively, the speed of a point-to-point arm movement follows a bell-shaped profile, which possesses intrinsically nonlinear characteristics and thus would need to a nonlinear model to describe it.

In these regards, we aimed to decode the speed and direction of 2D arm movements separately with nonlinear prediction algorithms based on LSTM. The proposed decoder was applied

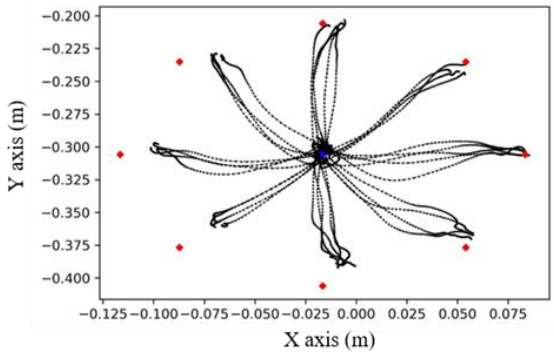

Fig. 1 Center out task profile. The red dot indicated the center position of eight targets. The blue dot at the center indicated the home position. The black dashed line represents the reaching trajectory of the monkey C. The x and y axis in units of meter.

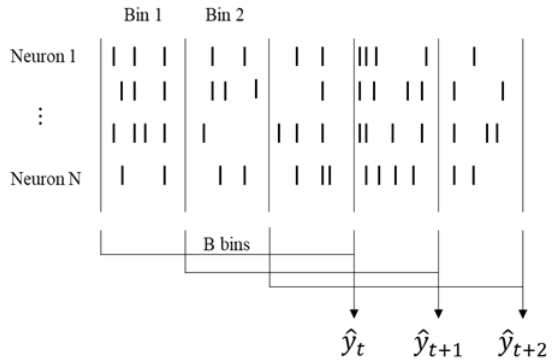

Fig. 2 Input schematic for neural decoder. The $\hat{\boldsymbol{y}}_{\boldsymbol{t}}$ represents the predicted kinematic variable at time t

to the publicly available neural and kinematic data set in which a primate performed the 2D center-out task. Its performance was compared with other decoders, including the linear Kalman filter decoding velocity as well as a LSTM decoding velocity. Hereafter, we will call the proposed decoder as a speed-direction LSTM (*sd*LSTM) and the previous velocity decoding LSTM as *v*LSTM.

II. DATA DESCRIPTION

The public data provided by the Collaborative Research in Computational Neuroscience – Data sharing (CRCNS) was used to test the proposed decoder [7]. The description of the data is given below.

A rhesus monkey performed the eight-target center-out task, where the monkey grasped and moved the two-link manipulandum to control the computer cursor. The eight targets were placed around a circle with a radius of 10 cm with a 45-degree interval. The home position at which the monkey started each trial was placed at the center of the circle. In each trial, the monkey held the manipulandum at the home position with a duration randomly given between 0.5 ~ 0.6s. Then, one of the eight targets, which was a square of 2 x 2 cm^2, was randomly highlighted and the center position became dark. The monkey reached the target and held over it for a random period between 0.2 s and 0.4 s to complete the trial. The eight targets were given equally in a random order. We analyzed the data of the first experiment recorded from the monkey C in the data set.

Fig. 1 shows the arm trajectories of the monkey in the center-out task. The monkey did not have to reach the target center completely because of the target size. The total number of trials was 194 among which we only analyzed those in which the monkey successfully acquired the target (175 trials). We used 60 % of the trials for training of the neural decoders and 40 % for validation and test (20% each).

The silicon microelectrode array (96-channel, Blackrock, Inc., USA) was implanted within the arm area of the contralateral primary motor cortex (M1). The total number of recorded neurons were 196. We excluded those neurons which seldom fired and caused the singular matrix in the the Kalman filter from the analysis, resulting in 158 neurons being used for the subsequent analysis.

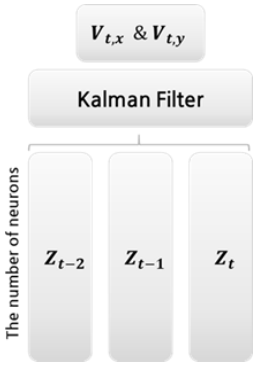

Fig. 3 The structure of the velocity Kalman filter

III. METHODS

A. Synchronization of neural data with kinematics

Fig. 2 shows how the neural data were synchronized with the kinematic data at each time instance. We first defined a time bin containing neuronal spikes as 50 ms. Then, we counted the number of spikes in each bin for each neuron. To predict the kinematic parameters at time *t*, *B* bin counts for every neuron were collectively used, where *B* was set to 3 in this study. There was no overlap between bins. The prediction was conducted every 50 ms.

B. Velocity Kalman Filter

Fig. 3 shows the structure of the velocity Kalman filter. The Kalman filter has been used for neural decoding to predict the kinematic state variables, such as position, velocity, acceleration, from neuronal observations, consisting of the prediction and update steps. The prediction step predicts the current state with the previous state, and the update step corrects the predicted state using a current neuronal observation. How much the observation is reflected on to update the predicted state is determined by the Kalman gain.

As the neural decoder, the Kalman filter can be illustrated with following equations [2]:

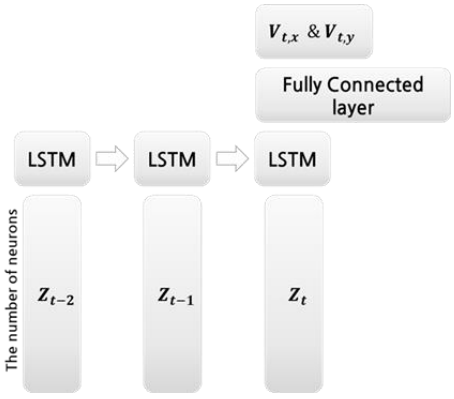

Fig. 4 The structure of the velocity LSTM

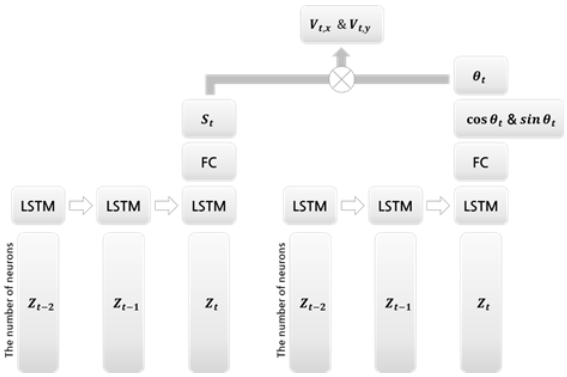

Fig. 5 The structure of the *sd*LSTM

$$Z_k = H_k X_k + q_k \quad (1)$$

$$X_{k+1} = A_k X_k + w_k \quad (2)$$

Z_k is the observed bin counts of 158 neurons over 3 bins at time k, k-1 an k-2, respectively.X_k is a state variable for the velocity of the hand. q_k is noise assumed to follow the normal distribution with zero mean and a covariance matrix Q_k. w_k is noise following the normal distribution with zero mean and a covariance matrix W_k . The matrix A represents a linear dynamics of velocity from time k-1 to k. The matrix H represents how the velocity state is encoded in firing activity. A, H, W, and Q were estimated by the least mean square solution.

C. *vLSTM*

Fig. 4 represents the structure of the velocity LSTM (*v*LSTM). Similar to the velocity Kalman filter, *v*LSTM received neuronal input data from time t-2 to t for predicting velocity at time t. Incoming neuronal input was processed in the LSTM cell and updated from the previous cell state. It was then passed to the LSTM cell of the next time step. After processing three consecutive neuronal bin count vectors sequentially, the final hidden state was passed to the fully connected layer that contained two output units to predict $V_{t,x}$ and $V_{t,y}$. The implementation of *v*LSTM was similar to that in the previous study [1]. The weights of the network were optimized with the Adam optimizer [8]. Also, the random search strategy was used to find an optimal parameter set [9]. It was sampled from in the uniform distribution of a range from 10 to 50 for hidden nodes. The number of epoch varied from 10 to 100, and the batch size from 10 to 50.

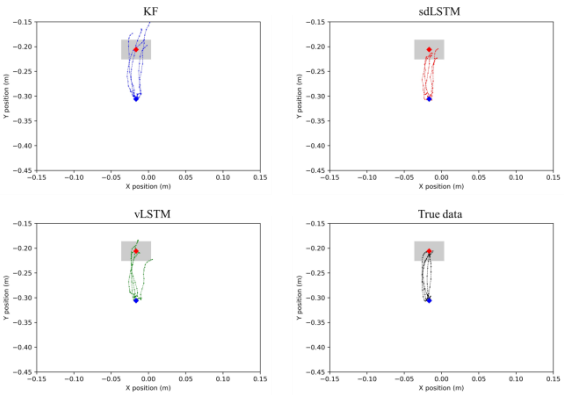

Fig. 6 The example of decoded trajectory. Home position was indicated as blue diamond and Target position was indicated as red diamond. The region that the trial is regarded as hit trial is shaded as grey.

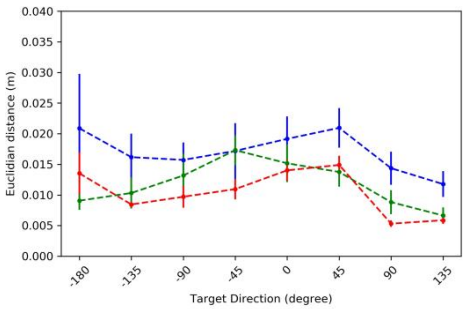

Fig. 7 Euclidean distance of the KF (blue), *v*LSTM (green), *sd*LSTM (red) in each target direction

D. *sdLSTM*

Fig. 5 represents the structure of *sd*LSTM, consisting of two parts. One part estimated the hand speed and the other estimated the hand direction. Each part passed the hidden state to the fully connected layer with a single output unit for speed estimation or two output units (cosine and sine of an angle) for direction estimation. The velocity $V_{t,x}$ and $V_{t,y}$ were then calculated by multiplying speed to a direction vector. We estimated a 2D unit-length direction vector constructed by cosine and sine functions of an angle instead of a scalar of the degree because of the periodicity. Previous studies have also built the bidirectional LSTM estimated the protein secondary structure [10], or predicted the association angle of protein backbone [11], using 2D direction vectors. *sd*LSTM was optimized with the Adam optimizer and the random search strategy. The parameter searching range was the same as *v*LSTM.

To asses the accuracy of decoding, Euclidean distance (ED) between decoded and true hand trajectories was calculated:

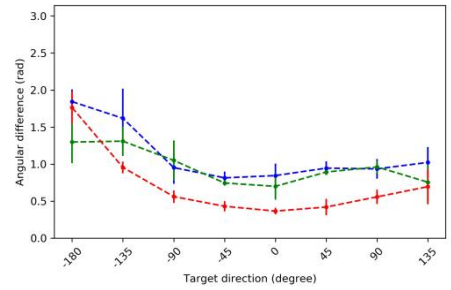

Fig. 8 Angular difference of the KF (blue), *v*LSTM (green), *sd*LSTM (red) in each target direction

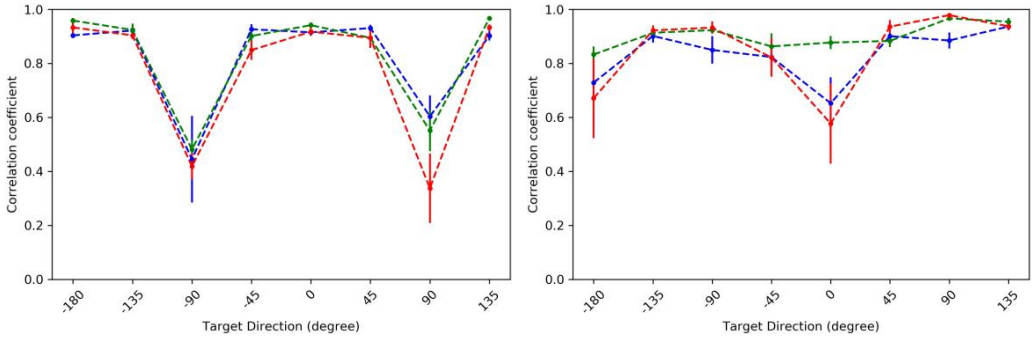

Fig. 9 Correlation coefficient of velocity x and velocity y of the KF (blue), *v*LSTM (green), *sd*LSTM (red) in each target direction

$$\mathrm{ED} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(p_{x,i}-\hat{p}_x)^2 + (p_{y,i}-\hat{p}_y)^2} \qquad (3)$$

In addition, a correlation coefficient (CC) between true and reconstructed velocity was calculated. The CC was measured on the x-, and y-coordinates, respectively.

$$CC = \frac{\sum_{i=1}^{n}(v_{x,i}-\overline{v_x})(v_{y,i}-\overline{v_y})}{\sqrt{\sum_{i=1}^{n}(v_{x,i}-\overline{v_x})^2}\sqrt{\sum_{i=1}^{n}(v_{y,i}-\overline{v_y})^2}} \qquad (4)$$

Finally, an angular difference (AD) between the reconstructed and true directions were calculated.

$$AD = \frac{1}{n}\sum_{t=0}^{n}\theta_t \qquad (5)$$

where, θ_t was calculated as.

$$\theta_t = \cos^{-1}\frac{V_t \cdot \hat{V}_t}{|V_t|\,|\hat{V}_t|} \qquad (6)$$

V_t and $\hat{V}_t$ were the velocity vectors recorded and predicted from decoding algorithms at time *t*, respectively.

IV. Results

Fig. 6 illustrates the example of the reconstructed and true trajectory. In many cases, the reconstructed trajectory by sdLSTM could reach the target more frequently than the other decoders. The target acquisition rate was 48.125, 43.125 and 31.875 percentage for *sd*LSTM, *v*LSTM, KF. The proposed *sd*LSTM showed the minimum ED on average among the decoders (Fig 7). The mean ED was 0.0103, 0.0118, and 0.017 m for *sd*LSTM, *v*LSTM and KF, respectively. Also, *sd*LSTM showed smaller AD than the other decoders on average (Fig 8). The mean AD was was 0.6994, 0.9587, and 1.104 rad for *sd*LSTM, *v*LSTM and KF, respectively. For CC, *sd*LSTM showed poor performance in some directions or similar performance in other directions compared with other decoders (Fig. 9). The mean CC for the x-velocity was was 0.755, 0.813, and 0.8047 for *sd*LSTM, *v*LSTM and KF, respectively. The mean CC for the y-velocity was was 0.8458, 0.9036, and 0.8316 for *sd*LSTM, *v*LSTM and KF, respectively. However, the target direction that elicited poor CC of *sd*LSTM did not affect reaching movements much. For example, reaching a target at -90 and 90 degrees entailed little movement along the direction of the x-axis. In the same way, reaching a target at 0 degree generated little movement along the y-axis. Therefore, *sd*LSTM that separately estimated speed naturally exhibited less correlations in these angles.

V. Conclusion

In the present study, we used LSTM to decode hand kinematics of a primate from motor cortical activity. LSTM was designed in two folds: a unified output predicting 2D hand velocity or two separate outputs predicting 2D hand direction and speed, respectively. The decoding results demonstrated that the LSTM with two separate outputs decoded hand kinematics better than the velocity decoding LSTM as well as the traditional KF.

Superior performance of *v*LSTM over KF suggests that the use of nonlinear models with a more complex structure might improve neural decoding. However, even with this deep learning algorithm, significant gaps between true hand kinematics and decoded output exist. Such gaps may not be fully overcome by enhancing decoding algorithms per se. Rather, it would be also desired to consider the properties of hand kinematics and find appropriate ways to associate them with decoding algorithms. This study shows that changing the way to cast kinematic parameters in a decoding algorithm could improve performance, supporting the argument above.

References

[1] Glaser, Joshua I., et al. "Machine learning for neural decoding." *arXiv preprint arXiv:1708.00909* (2017).

[2] Wu, Wei, et al. "Neural decoding of cursor motion using a Kalman filter." *Advances in neural information processing systems*. 2003.

[3] Li, Zheng, et al. "Unscented Kalman filter for brain-machine interfaces." *PloS one* 4.7 (2009): e6243.

[4] Gilja, Vikash, et al. "A brain machine interface control algorithm designed from a feedback control perspective." Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE. IEEE, 2012

[5] Perel, Sagi, et al. "Direction and speed tuning of motor-cortex multi-unit activity and local field potentials during reaching movements." *Engineering in Medicine and Biology Society*

(EMBC), 2013 35th Annual International Conference of the IEEE. IEEE, 2013.

[6] Amirikian, Bagrat, and Apostolos P. Georgopulos. "Directional tuning profiles of motor cortical cells." *Neuroscience research*36.1 (2000): 73-79.

[7] Flint, Robert D., et al. "Accurate decoding of reaching movements from field potentials in the absence of spikes." *Journal of neural engineering* 9.4 (2012): 046006.

[8] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[9] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (2012): 281-305.

[10] Heffernan, Rhys, et al. "Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility." *Bioinformatics* 33.18 (2017): 2842-2849

[11] Lyons, James, et al. "Predicting backbone Cα angles and dihedrals from protein sequences by stacked sparse auto-encoder deep neural network." *Journal of computational chemistry* 35.28 (2014): 2040-2046.