# Machine learning for neural decoding

Joshua I. Glaser[1,2*], Raeed H. Chowdhury[3,4], Matthew G. Perich[3,4], Lee E. Miller[2-4], and Konrad P. Kording[2-7]

1. Interdepartmental Neuroscience Program, Northwestern University, Chicago, IL, USA
2. Department of Physical Medicine and Rehabilitation, Northwestern University and Shirley Ryan Ability Lab, Chicago, IL, USA
3. Department of Physiology, Northwestern University, Chicago, IL, USA
4. Department of Biomedical Engineering, Northwestern University, Chicago, IL, USA
5. Department of Applied Mathematics, Northwestern University, Chicago, IL, USA
6. Department of Neuroscience, University of Pennsylvania, Philadelphia, IL, USA
7. Department of Biomedical Engineering, University of Pennsylvania, Philadelphia, IL, USA

* Contact: j-glaser@u.northwestern.edu

**Abstract:**
While machine learning tools have been rapidly advancing, the majority of neural decoding approaches still use last century's methods. Improving the performance of neural decoding algorithms allows us to better understand what information is contained in the brain, and can help advance engineering applications such as brain machine interfaces. Here, we apply modern machine learning techniques, including neural networks and gradient boosting, to decode from spiking activity in 1) motor cortex, 2) somatosensory cortex, and 3) hippocampus. We compare the predictive ability of these modern methods with traditional decoding methods such as Wiener and Kalman filters. Modern methods, in particular neural networks and ensembles, significantly outperformed the traditional approaches. For instance, for all of the three brain areas, an LSTM decoder explained over 40% of the unexplained variance from a Wiener filter. These results suggest that modern machine learning techniques should become the standard methodology for neural decoding. We provide code to facilitate wider implementation of these methods.

## Introduction:

Decoding is a critical tool for understanding how neural signals relate to the outside world. It can be used to determine how much information the brain contains about an external variable (e.g. sensation or movement) [1-3], and how this information differs across brain areas [4-6], experimental conditions [7, 8], disease states [9], and more. It is also useful in engineering contexts, such as for brain machine interfaces (BMIs), where signals from motor cortex are used to control computer cursors [10], robotic arms [11], and muscles [12]. Decoding is a central tool for neural data analysis.

Because decoding is simply a regression or classification problem, many methods can be used for neural decoding. Despite the many recent advances in machine learning techniques, it is still very common to use traditional methods such as linear regression. Using modern machine learning tools for neural decoding would likely significantly boost performance, and might allow deeper insights into neural function.

Here, we compare many different machine learning methods to decode information from neural spiking activity. We predict movement velocities from macaque motor cortex and sensorimotor cortex, and locations in space from rat hippocampus. In all brain regions, modern methods, in particular neural networks and ensembles, led to the highest accuracy decoding, even for limited amounts of data. We provide code so that others can easily use all the decoding methods we tested.

# Methods:

## Tasks and brain regions:

**Decoding movement velocity from the motor cortex and somatosensory cortex:** In our "random-target" experiment [8], monkeys moved a planar manipulandum that controlled a cursor on the screen (Fig. 1a). The monkeys continuously reached to new targets that were presented with a brief hold period between reaches. After training, the monkeys were surgically implanted with 96-channel Utah electrode arrays (Blackrock Microsystems, Salt Lake City, UT) to record the extracellular activity of cortical neurons. In one experiment [8], we recorded from both primary motor cortex (M1) and dorsal premotor cortex (PMd) and combined neurons from both areas. In another experiment we recorded from area 2 of primary somatosensory cortex (S1) [13]. From both brain regions, we aimed to predict the x and y components of movement velocity. The recording from motor cortex was 21 minutes, and contained 164 neurons. The mean and median firing rates, respectively, were 6.7 and 3.4 spikes / sec. The recording from S1 was 51 minutes, and contained 52 neurons. The mean and median firing rates, respectively, were 9.3 and 6.3 spikes / sec.

**Decoding position from the hippocampus:** We used a dataset from CRCNS, in which rats chased rewards on a square platform (Fig. 1b) [14, 15]. Extracellular recordings were made from layer CA1 of dorsal hippocampus (HC). We aimed to predict the x and y position of the rat. The recording from HC was 93 minutes, and contained 58 neurons. We excluded neurons with fewer than 100 spikes over the duration of the experiment, resulting in 46 neurons. These neurons had mean and median firing rates, respectively, of 1.7 and 0.2 spikes / sec.

## General Decoding methods:

**Decoding movement velocity from the motor cortex and somatosensory cortex:** We predicted the average velocity (x and y components) in 50 ms bins. Neural spike trains used for decoding were also put into 50 ms bins. In motor cortex, we used 700 ms of neural activity (13 bins before and the concurrent bin) to predict the current movement velocities, as the primary interest is in investigating how motor cortex causally affects movement. In somatosensory cortex, we used 650 ms surrounding the movement (6 bins before, the concurrent bin, and 6 bins after), as neural activity

has been shown both preceding and following movements [16].

**Decoding position from the hippocampus:** We aimed to predict the position (x and y coordinates) of the rat in 200 ms bins. Neural spike trains used for decoding were also put into 200 ms bins. We used 2 seconds of surrounding neural activity (4 bins before, the concurrent bin, and 5 bins after) to predict the current position.

**Scoring Metric:** To determine the goodness of fit, we used $R^2 = 1 - \dfrac{\sum_i \left( \hat{y}_i - y_i \right)^2}{\sum_i \left( y_i - \bar{y} \right)^2}$ , where $\hat{y}_i$ are the predicted values, $y_i$ are the true values and $\bar{y}$ is the mean value. This formulation of $R^2$ (which is the fraction of variance accounted for, rather than the Pearson's correlation coefficient squared [17]) can be negative on the test set due to overfitting on the training set. The reported $R^2$ values are the average across the x and y components of velocity or position.

**Cross-validation:** When determining the $R^2$ for every method (Fig. 3), we used 10 fold cross-validation. For each fold, we split the data into a training set (80% of data), a contiguous validation set (10% of data), and a contiguous testing set (10% of data). For each fold, decoders were trained to minimize the mean squared error between the predicted and true velocities/positions of the training data. We found the algorithm hyperparameters that led to the highest $R^2$ on the validation set using Bayesian optimization [18]. That is, we fit many models on the training set with different hyperparameters and calculated the $R^2$ on the validation set. Then, using the hyperparameters that led to the highest validation set $R^2$, we calculated the $R^2$ value on the testing set. Error bars on the test set $R^2$ values were computed across cross-validation folds.

**Bootstrapping:** When determining how performance scaled as function of data size (Fig. 5), we used a single test set and validation set, and varying amounts of training data that directly preceded the validation set. The test and validation sets were 5 minutes long for motor and somatosensory cortices, and 7.5 minutes for hippocampus. To get error bars, we resampled from the test set. Because of the high correlation between temporally adjacent samples, we didn't resample

randomly from all examples (which would create highly correlated resamples). Instead, we separated the test set into 20 temporally distinct subsets, $S_1$-$S_{20}$ (i.e., $S_1$ is from $t=1$ to $t=T/20$, $S_2$ is from $t=T/20$ to $t=2T/20$, etc., where $T$ is the end time), that were more nearly independent of each other. We then resampled combinations of these 20 subsets (e.g. $S_5$, $S_{13}$, ... $S_2$) 1000 times to get confidence intervals of $R^2$ values.

**Preprocessing:** The training input was normalized (z-scored). The training output was zero-centered (mean subtracted), except in support vector regression, where the output was z-scored. The validation/testing inputs and outputs were preprocessed using the preprocessing parameters from the training set.

*Specific Decoders:*

**Kalman Filter:** In the Kalman filter, the hidden state at time $t$ is a linear function of the hidden state at time $t-1$, plus a matrix characterizing the uncertainty. The observation (measurement) at time $t$ is a linear function of the hidden state at time $t$ (plus noise). At every time point, to update the estimated hidden state, the updates derived from the current measurement and the previous hidden states are combined. During this combination, the noise matrices give a higher weight to the less uncertain information. We used a Kalman filter similar to that implemented in [19]. In the Kalman filter, the measurement was the neural spike trains, and the hidden state was the kinematics (x and y components of position, velocity, and acceleration). We had one hyperparameter which differed from the implementation in [19]. This parameter determined the noise matrix associated with the transition in kinematic states (**Q** in [19]). We divided the empirical noise matrix of training data (used in [19]) by the hyperparameter scalar $C$. The rationale for this addition is that neurons have temporal correlations, which make it desirable to have a parameter that allows changing the weight of the new neural evidence. Interestingly, the introduction of this parameter made a big difference for the hippocampus dataset (Fig. S1). We also allowed for a lag between the neural data and predicted kinematics. The lag and hyperparameter were determined based on validation set performance.

**Wiener Filter:** The Wiener filter uses multiple linear regression to predict the output from multiple time bins of every neurons' spikes. That is, the output is assumed to be a linear mapping of the number of spikes in the relevant time bins from every neuron (Fig. 1c,d). Here, separate models were used to predict the x and y components of the kinematics.

**Wiener Cascade:** The Wiener cascade (also known as a linear nonlinear model) fits a linear regression (the Wiener filter) followed by a fitted static nonlinearity (e.g. [20]). This allows for a nonlinear relationship between the input and the output, and assumes that this nonlinearity is purely a function of the linear output. Here, as in the Wiener Filter, the input was neurons' spike rates over relevant time bins. The nonlinear component was a polynomial with degree determined on the validation set. Separate models were used to predict the x and y components of the kinematics.

**Support Vector Regression:** In support vector machine regression (SVR) [21], the inputs are projected into a higher dimensional space using a nonlinear kernel, and then linearly mapped from this space to the output to minimize an objective function [21]. Here, we used standard support vector regression (SVR) with a radial basis function kernel to predict the kinematics from the neurons' spike rates in each bin. We set hyperparameters for the penalty of the error term and the maximum number of iterations. Separate models were used to predict the x and y components of the kinematics.

**XGBoost:** XGBoost (Extreme Gradient Boosting) [22] is an implementation of gradient boosted trees. For the regression problem, gradient boosting fits many regression trees. Each subsequent regression tree is fit to the residuals of the previous fit. Regression trees create nonlinear mappings from the input to output. Here, we used the XGBoost to predict the kinematics from the neurons' spike rates in each bin. We set hyperparameters for the maximum depth of the tree, number of trees, and learning rate. Separate models were used to predict the x and y components of the kinematics.

**Feedforward Neural Network:** A feedforward neural net connects the inputs to sequential layers of hidden units via linear mappings followed by output nonlinearities. This can allow for mapping complex nonlinear functions from input to output. Here, using the Keras library [23], we created a fully connected (dense) feedforward neural network with 2 hidden layers and rectified linear unit activations after each hidden layer. We required the number of hidden units in each layer to be the same. We set hyperparameters for the number of hidden units in the layers, amount of dropout [24], and number of training epochs. We used the Adam

algorithm [25] as the optimization routine. This neural network, and all neural networks below had 2 output units. That is, the same network predicted the x and y components rather than there being 2 separate networks. The input was still the number of spikes in each bin from every neuron. Note that we refer to feedforward neural networks as a "modern" technique, despite their having been around for many decades, due to their current resurgence and the modern methods for training the networks.

**Simple RNN:** In a standard recurrent neural network (RNN), the hidden state is a linear combination of the inputs and the previous hidden state. This hidden state is then run through an output nonlinearity, and linearly mapped to the output. RNNs, unlike feedforward neural networks, allow temporal changes in the system to be modeled explicitly. Here, using the Keras library [23], we created a neural network architecture where the spiking input from all neurons was fed into a standard recurrent neural network (Fig. 1e). The units from this recurrent layer were fed through a rectified linear unit nonlinearities, and fully connected to an output layer with 2 units (x and y velocity or position components). We set hyperparameters for the number of units, amount of dropout, and number of training epochs. We used RMSprop [26] as the optimization routine.

**Gated Recurrent Unit:** Gated recurrent units (GRUs) [27] are a more complex type of recurrent neural network. It has gated units, which in practice allow for better learning of long-term dependencies. For implementation, all methods were the same as for the "Simple RNN", except Gated Recurrent Units were used rather than a traditional RNN.

**Long Short Term Memory Network:** Like the GRU, the long short term memory (LSTM) network [28] is a more complex recurrent neural network with gated units that allow long-term dependencies to be captured better. The LSTM has more parameters than the GRU. For implementation, all methods were the same as for the "Simple RNN", except LSTMs were used.

**Ensemble:** Ensemble methods combine the predictions from several other methods, and thus have the potential to leverage the different benefits of the methods contained within the ensemble. Here, using the predictions from all decoders except the Kalman filter (which had a different format) as inputs, we predicted the outputs using the feedforward neural network described above.

**Code:** Python code for all methods is available at https://github.com/KordingLab/Neural_Decoding
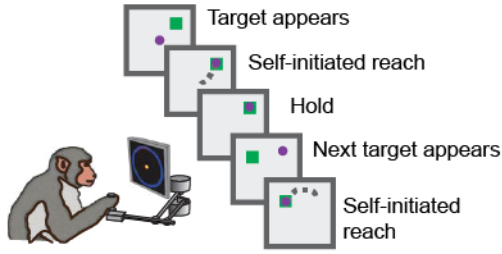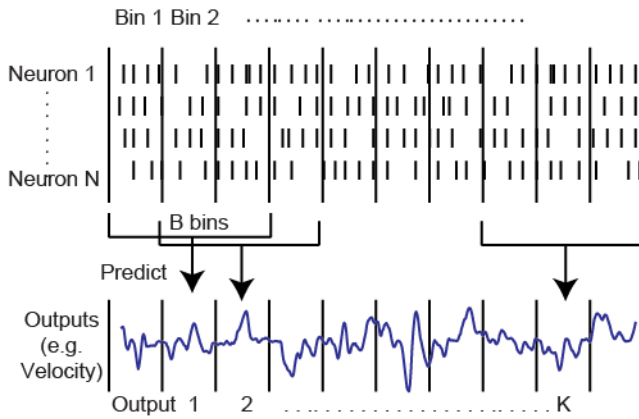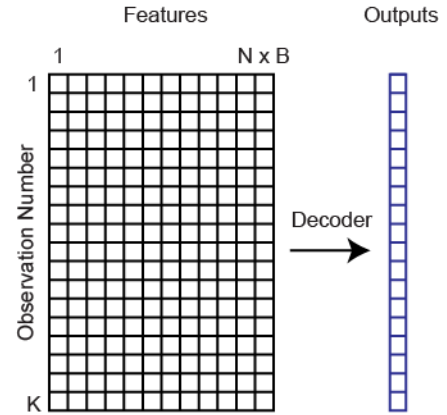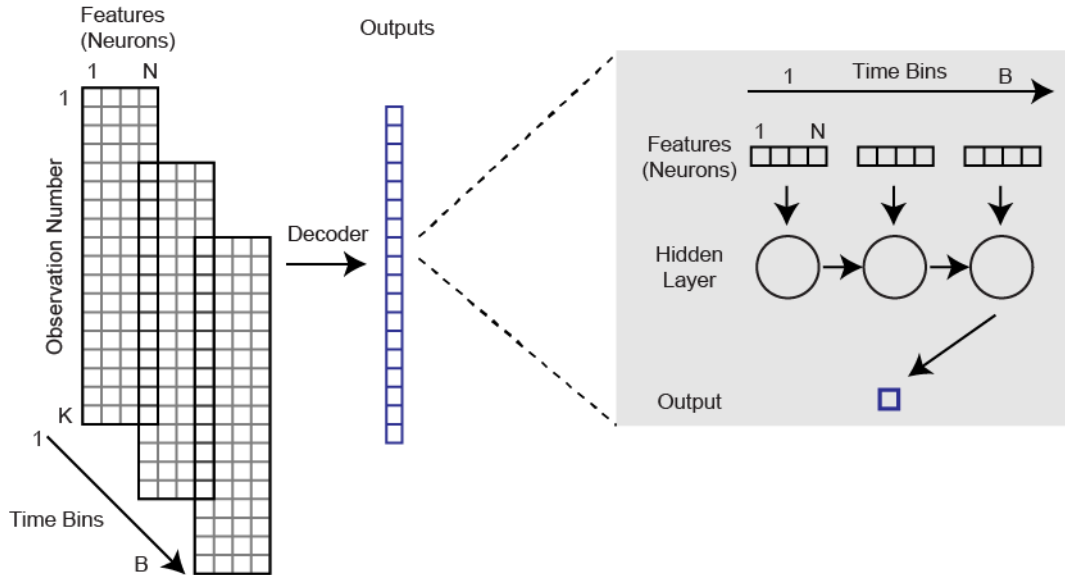
**Figure 1: Tasks and Decoding Schematic**
**a)** In the task for decoding from motor and somatosensory cortices, monkeys moved a planar manipulandum that controlled a cursor on the screen. The monkeys continuously reached to new targets that were presented, with a brief hold period between reaches [8]. **b)** In the task for decoding from hippocampus, rats chased rewards on a square platform. **c)** To decode (predict) the output in a given time bin, we used the firing rates of all *N* neurons in *B* time bins. In this schematic, *N*=4 and *B*=3 (one bin preceding the output, one concurrent bin, and one following bin). In our data, we predicted two outputs from each brain region (x and y components of velocity from motor and somatosensory cortex, and x and y components of position from hippocampus). For each region, the number of neurons and time bins

used for decoding are described in *Methods*. Also, note that this schematic does not apply for the Kalman Filter decoder. **d)** For the non-recurrent decoders (Wiener Filter, Wiener Cascade, Support Vector Regression, XGBoost, and Feedforward Neural Network), this is a standard machine learning regression problem where $N$ x $B$ features (the firing rates of each neuron in each relevant time bin) is used to predict the output. **e)** For the recurrent decoders (simple recurrent neural network, GRUs, LSTMs), to predict an output, we used $N$ features, with temporal connections across $B$ bins. A schematic of a recurrent neural network predicting a single output is on the right.
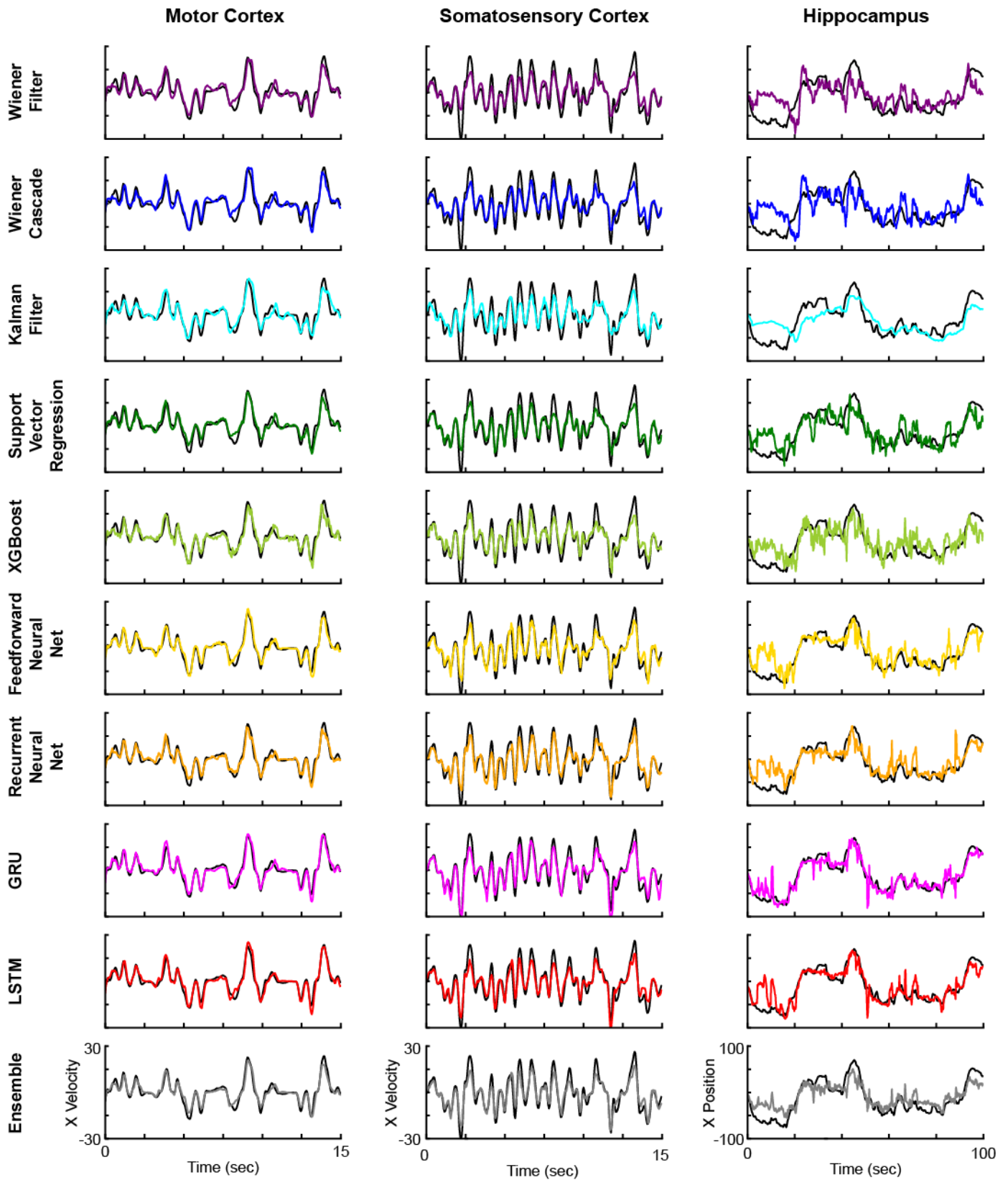
## Results

We investigated how the choice of machine learning technique affects decoding performance (Fig. 1) using a plethora of common machine learning methods. These ranged from historical linear techniques (e.g., the Wiener filter) to modern machine learning techniques (e.g., neural networks and ensembles of techniques). We tested the performance of all these techniques across datasets from motor cortex, somatosensory cortex, and hippocampus.

We aimed to understand the performance of the methods when fit to neural data. First, in order to get a qualitative impression of the performance, we plotted the output of each decoding method for each of the three datasets (Fig. 2). In these examples, the modern methods, such as the LSTM and ensemble, appeared to outperform traditional methods, such as the Wiener and Kalman filters, as the predictions were slightly closer to the true output. Next, we quantitatively compared the methods. In all three brain areas, modern machine learning methods outperformed traditional decoding methods (Fig. 3). In particular, neural networks and the ensemble led to the best performance, while the Wiener or Kalman Filter led to the worst performance. In fact, the LSTM decoder explained over 40% of the unexplained variance from a Wiener filter ($R^2$'s of 0.88, 0.86, 0.62 vs. 0.78, 0.75, 0.35). Additionally, the feedforward neural

network did almost as well as the LSTM in all brain areas. Across cases, the ensemble method added a reliable, but small increase to the explained variance. Modern machine learning methods led to significant increases in predictive power.

While modern machine learning methods yielded the best performance on our full datasets, it is possible, because of their greater complexity, that they would not work well with less data. Thus, we tested the feedforward neural network and LSTM (two modern methods that worked particularly well), along with the Wiener and Kalman filters, on varying amounts of data. Even with limited data, the modern methods worked very well. With only 2 minutes of training data for motor and somatosensory cortices, and 15 minutes of hippocampus data, both modern methods outperformed both traditional methods (Fig. 4,5). When decreasing the amount of training data further, to only 1 minute for motor and somatosensory cortices and 7.5 minutes for hippocampus data, the Kalman filter sometimes performed comparably to the modern methods. Still, the modern methods significantly outperformed the Wiener Filter (Fig. 5). Thus, even for limited data, modern machine learning methods can yield significant gains in decoding performance.
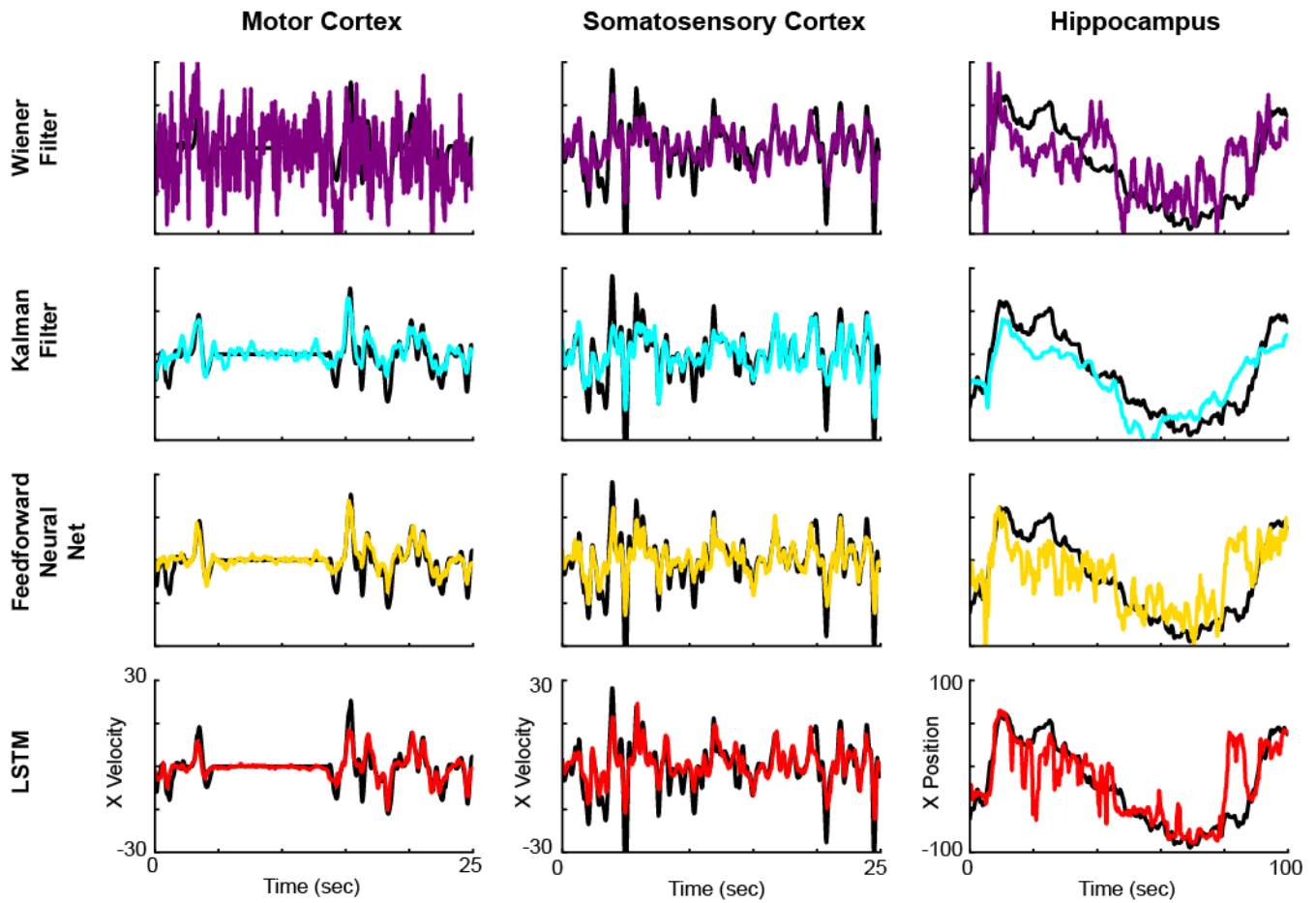
**Figure 2: Example Decoder Results**

Example decoding results from motor cortex (left), somatosensory cortex (middle), and hippocampus (right), for all ten methods (top to bottom). Ground truth traces are in black, while decoder results are in various colors.
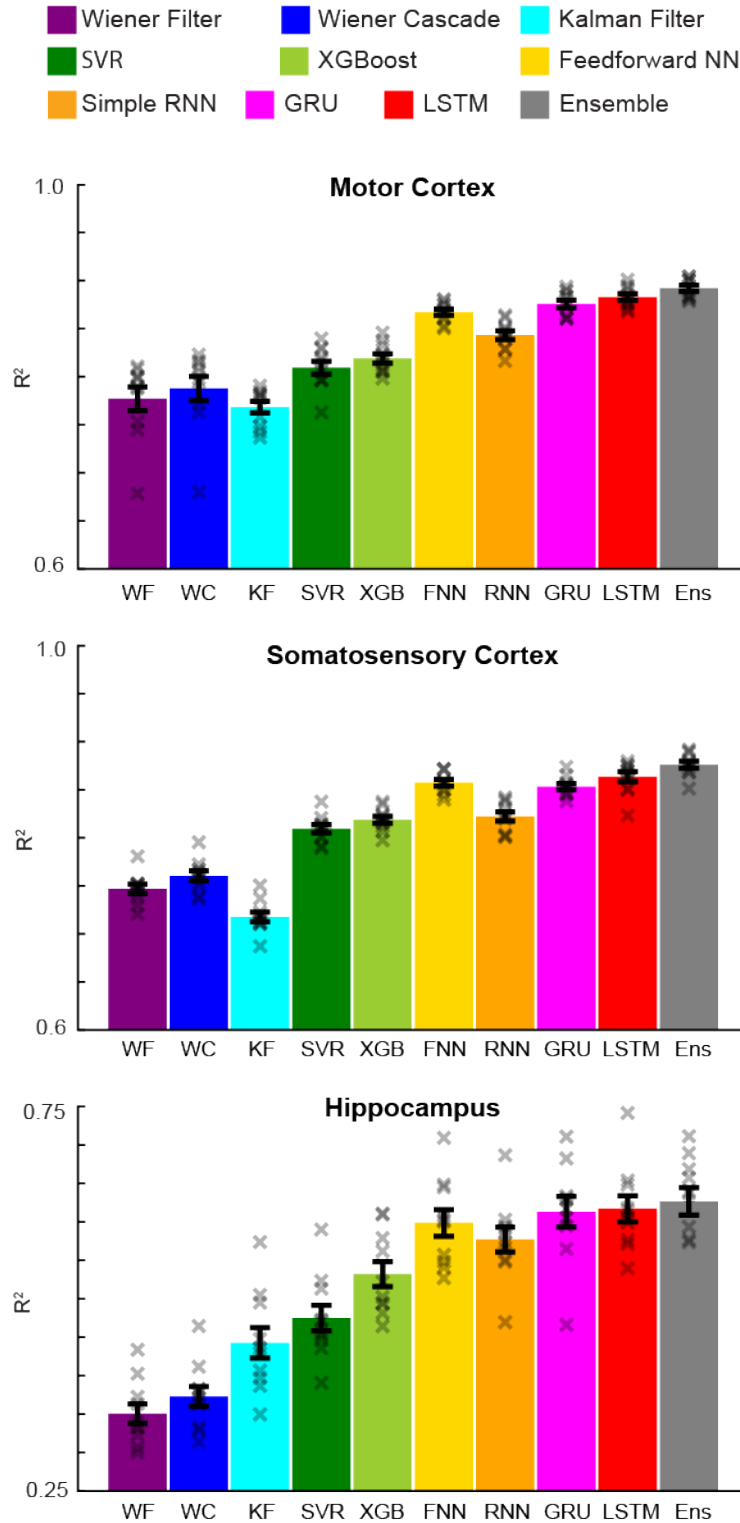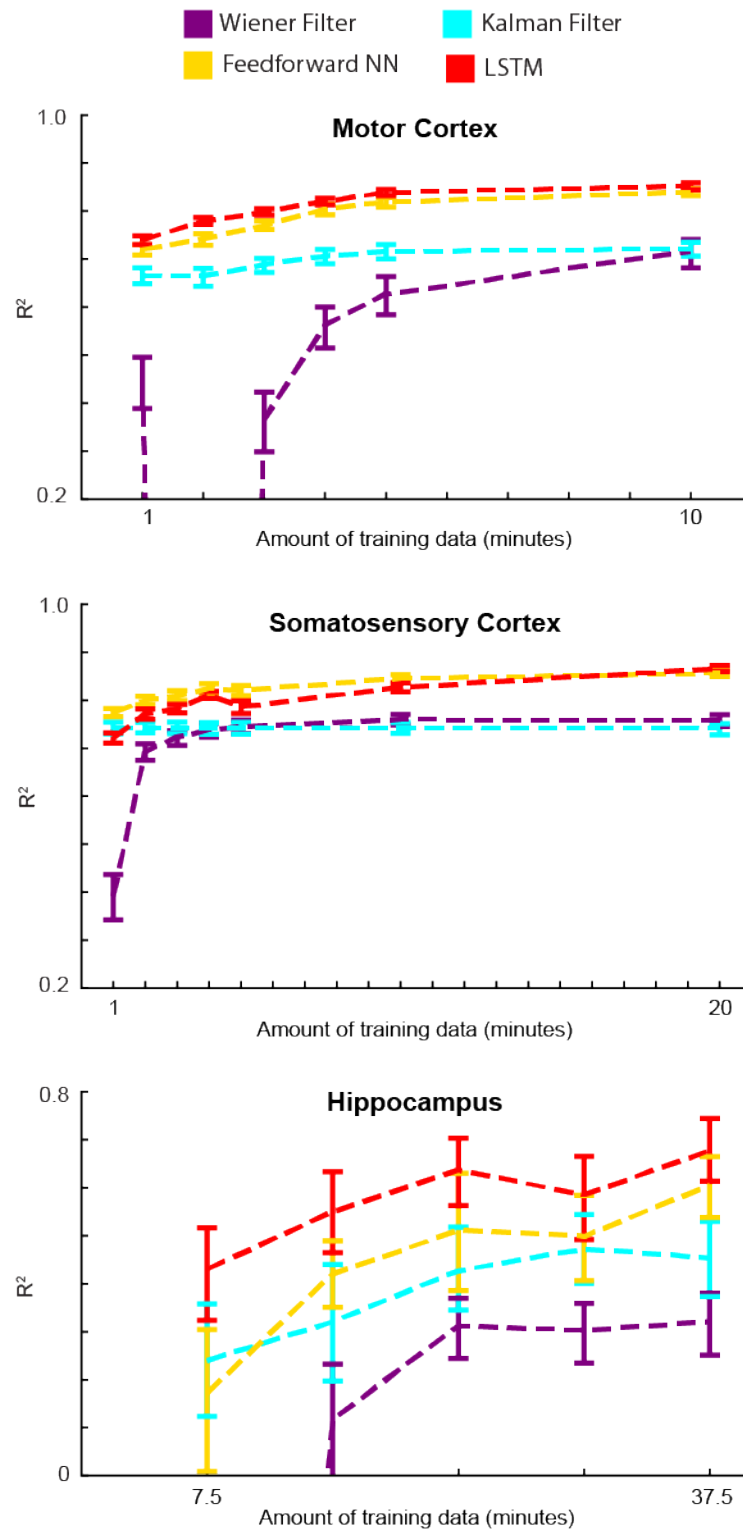
**Figure 3: Decoder Result Summary**
$R^2$ values are reported for all decoders (different colors) for each brain area (top to bottom). Error bars represent the mean +/- SEM across cross-validation folds. X's represent the $R^2$ values of each cross-validation fold. Note the different y-axis limits for the hippocampus dataset.

**Figure 4: Example results with limited training data**
Using only 2 minutes of training data for motor cortex and somatosensory cortex, and 15 minutes of training data for hippocampus, we trained two traditional methods (Wiener filter and Kalman filter), and two modern methods (feedforward neural network and LSTM). Example decoding results are shown from motor cortex (left), somatosensory cortex (middle), and hippocampus (right), for these methods (top to bottom). Ground truth traces are in black, while decoder results are in the same colors as previous figures.

**Figure 5: Decoder results with varying amounts of training data**
Using varying amounts of training data, we trained two traditional methods (Wiener filter and Kalman filter), and two modern methods (feedforward neural network and LSTM). $R^2$ values are reported for these decoders (different colors) for each brain area (top to bottom). Error bars are 68% confidence intervals (meant to approximate the SEM) produced via bootstrapping, as we used a single test set. Values with negative $R^2$s were not shown. Also note the different y-axis limits for the hippocampus dataset.

## Discussion:

Here we tested the performance of a large number of decoding techniques on three different neural decoding problems. We found that, across datasets, neural networks outperformed traditional methods. An ensemble method provided only minor additional predictive power. The strong performance of neural networks even persisted for small datasets with as little as one minute of training data.

We find it particularly interesting that the neural network methods worked so well with limited data, which is counter to the common perception. We believe the explanation is simply the size of networks. For instance, our networks have on the order of 100 thousand parameters, while common networks for image classification (e.g. [29]) can have on the order of 100 million parameters. Thus, the reasonable size of our networks (hundreds of hidden units) likely allowed for excellent prediction with limited data [30].

It is also intriguing that the feedforward neural network did almost as well as the LSTM and better than the standard RNN, considering the recent attention to treating the brain as a dynamical system [31]. For the motor and somatosensory cortex decoding, it is possible that the highly trained monkeys yielded a stereotyped temporal relationship between neural activity and movement that a feedforward neural network could effectively capture. It would be interesting to compare the performance of feedforward and recurrent neural networks on less constrained behavior.

In order to find the best hyperparameters for the decoding algorithms, we used a Bayesian optimization routine [18] to search the hyperparameter space (see *Methods*). Still, it is possible that some of the decoding algorithms did not use the optimal hyperparameters, which could have lowered overall accuracy. Moreover, for several methods we did not fit all available hyperparameters. We did this in order to simplify the use of these methods, in order to decrease computational runtime during hyperparameter optimization, and because adding additional hyperparameters did not appear to improve accuracy. For example, for the neural nets we used dropout but not L1 or L2 regularization, and for XGBoost we used less than half the available hyperparameters for avoiding overfitting. While our preliminary testing with additional hyperparameters did not appear to significantly change the results, it is possible that we have not achieved optimal performance of our methods.

While we have tested standard algorithms on three different datasets, it is possible that the relative performance of algorithms differs on other datasets. However, many datasets in neuroscience share basic properties with those we used. Most are similar in length (tens of minutes to a couple hours), simply because the length of a recording session is usually limited by both the patience of the animal and the experimentalist. Moreover, most variables of interest have similar relevant timescales, where movement, speech, vision, and many other phenomena unfold on a timescale of hundreds of milliseconds to seconds. We thus expect that similar results would be obtained for other spiking datasets.

We have decoded from spiking data, but it is possible that the problem of decoding from other data modalities is different. One main driver of a difference may be the distinct levels of noise. For example, fMRI signals have far higher noise levels than spikes. As the noise level goes up, linear techniques become more appropriate, which may ultimately lead to a situation where the traditional linear techniques become superior. Applying the same analyses we did here across different data modalities is an important next step.

All our decoding was done "offline," meaning that the decoding occurred after the recording, and was not part of a control loop. This type of decoding is useful for determining how information in a particular brain area relates to an external variable. However, for engineering applications such as BMIs [32, 33], the goal is to decode information (e.g., predict movements) in real time. Our results here may not apply as directly to online decoding situations, since the subject is ultimately able to adapt to imperfections in the decoder. In that case, even relatively large decoder performance differences may be irrelevant. An additional challenge for online applications is computational runtime, which we have not addressed here. In the future, it would be valuable to test modern machine

learning techniques for decoding in online applications (as in [34]).

While modern machine learning methods provide an increase in decoding accuracy, it is important to be careful with the scientific interpretation of decoding results. Decoding can tell us how much information a neural population has about a variable $X$. However, high decoding accuracy does not mean that a brain area is directly involved in processing $X$, or that $X$ is the purpose of the brain area. For example, with a powerful decoder, it could be possible to accurately classify images based on recordings from the retina, since the retina has information about all visual space. However, this does not mean that the primary purpose of the retina is image classification. Moreover, even if the neural signal comes before the external variable, it does not mean that it is causally involved. For example, information could be in somatosensory cortex prior to movement due to an efference copy from M1. Thus, researchers should constrain interpretations to being about the information in neural populations, and how it may vary across brain regions, experimental conditions, or time intervals.

We decoded continuous valued variables. The same methods can be used for classification tasks, which often use classic decoders such as logistic regression and support vector machines. While here we have not demonstrated the benefit of modern machine learning methods for classification, our available code can easily be modified to allow users to do classification.

Neural engineering has a history of developing specialized algorithms meant to increase the performance of decoders [35-37]. However, these algorithms are not typically tested against state of the art machine learning algorithms. Along with this manuscript, we have released a package to do neural decoding using all the described methods, making it is easy to compare with any new algorithm. Our hunch is that it will be hard for specialized algorithms to compete with the standard algorithms developed by a massive community in machine learning.

# References

1.      Raposo D, Kaufman MT, Churchland AK. A category-free neural population supports evolving demands during decision-making. Nature neuroscience. 2014;17(12):1784-92.
2.      Rich EL, Wallis JD. Decoding subjective decisions from orbitofrontal cortex. Nature neuroscience. 2016;19(7):973-80.
3.      Hung CP, Kreiman G, Poggio T, DiCarlo JJ. Fast readout of object identity from macaque inferior temporal cortex. Science. 2005;310(5749):863-6.
4.      Quiroga RQ, Snyder LH, Batista AP, Cui H, Andersen RA. Movement intention is better predicted than attention in the posterior parietal cortex. J Neurosci. 2006;26(13):3615-20.
5.      Hernández A, Nácher V, Luna R, Zainos A, Lemus L, Alvarez M, et al. Decoding a perceptual decision process across cortex. Neuron. 2010;66(2):300-14.
6.      van der Meer MA, Johnson A, Schmitzer-Torbert NC, Redish AD. Triple dissociation of information processing in dorsal striatum, ventral striatum, and hippocampus on a learned spatial decision task. Neuron. 2010;67(1):25-32.
7.      Dekleva BM, Ramkumar P, Wanda PA, Kording KP, Miller LE. Uncertainty leads to persistent effects on reach representations in dorsal premotor cortex. eLife. 2016;5:e14316. doi: 10.7554/eLife.14316.
8.      Glaser JI, Perich MG, Ramkumar P, Miller LE, Kording KP. Population Coding Of Conditional Probability Distributions In Dorsal Premotor Cortex. bioRxiv. 2017:137026.
9.      Weygandt M, Blecker CR, Schäfer A, Hackmack K, Haynes J-D, Vaitl D, et al. fMRI pattern recognition in obsessive–compulsive disorder. Neuroimage. 2012;60(2):1186-93.
10.     Serruya MD, Hatsopoulos NG, Paninski L, Fellows MR, Donoghue JP. Brain-machine interface: Instant neural control of a movement signal. Nature. 2002;416(6877):141-2.
11.     Collinger JL, Wodlinger B, Downey JE, Wang W, Tyler-Kabara EC, Weber DJ, et al. High-performance neuroprosthetic control by an individual with tetraplegia. The Lancet. 2013;381(9866):557-64.
12.     Ethier C, Oby ER, Bauman MJ, Miller LE. Restoration of grasp following paralysis through brain-controlled stimulation of muscles. Nature. 2012;485(7398):368-71. doi: 10.1038/nature10987. PubMed PMID: 22522928; PubMed Central PMCID: PMCPMC3358575.
13.     Benjamin AS, Fernandes HL, Tomlinson T, Ramkumar P, VerSteeg C, Miller L, et al. Modern machine learning far outperforms GLMs at predicting spikes. bioRxiv. 2017:111450.
14.     Mizuseki K, Sirota A, Pastalkova E, Buzsáki G. Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop. Neuron. 2009;64(2):267-80.
15.     Mizuseki K, Sirota A, Pastalkova E, Buzsáki G. Multi-unit recordings from the rat hippocampus made during open field foraging. 2009. doi: http://dx.doi.org/10.6080/K0Z60KZ9.
16.     London BM, Miller LE. Responses of somatosensory area 2 neurons to actively and passively generated limb movements. Journal of neurophysiology. 2013;109(6):1505-13.
17.     Fagg AH, Ojakangas GW, Miller LE, Hatsopoulos NG. Kinetic trajectory decoding using motor cortical ensembles. IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2009;17(5):487-96.
18.     Snoek J, Larochelle H, Adams RP, editors. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems; 2012.
19.     Wu W, Black MJ, Gao Y, Serruya M, Shaikhouni A, Donoghue J, et al., editors. Neural decoding of cursor motion using a Kalman filter. Advances in neural information processing systems; 2003.
20.     Pohlmeyer EA, Solla SA, Perreault EJ, Miller LE. Prediction of upper limb muscle activity from motor cortical discharge during reaching. Journal of neural engineering. 2007;4(4):369.
21.     Chang C-C, Lin C-J. LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST). 2011;2(3):27.
22.     Chen T, Guestrin C, editors. Xgboost: A scalable tree boosting system. Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016: ACM.
23.     Chollet F. Keras. 2015.
24.     Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research. 2014;15(1):1929-58.
25.     Kingma D, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014.
26.     Tieleman T, Hinton G. Lecture 6.5-RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning. 2012.
27.     Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:14061078. 2014.
28.     Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997;9(8):1735-80.
29.     Krizhevsky A, Sutskever I, Hinton GE, editors. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems; 2012.

30.     Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:161103530. 2016.

31.     Shenoy KV, Sahani M, Churchland MM. Cortical control of arm movements: a dynamical systems perspective. Annu Rev Neurosci. 2013;36:337-59. doi: 10.1146/annurev-neuro-062111-150509. PubMed PMID: 23725001.

32.     Kao JC, Stavisky SD, Sussillo D, Nuyujukian P, Shenoy KV. Information systems opportunities in brain–machine interface decoders. Proceedings of the IEEE. 2014;102(5):666-82.

33.     Nicolas-Alonso LF, Gomez-Gil J. Brain computer interfaces, a review. Sensors. 2012;12(2):1211-79.

34.     Sussillo D, Nuyujukian P, Fan JM, Kao JC, Stavisky SD, Ryu S, et al. A recurrent neural network for closed-loop intracortical brain–machine interface decoders. Journal of neural engineering. 2012;9(2):026027.

35.     Kao JC, Nuyujukian P, Ryu SI, Churchland MM, Cunningham JP, Shenoy KV. Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. Nature communications. 2015;6.

36.     Corbett E, Perreault E, Koerding K, editors. Mixture of time-warped trajectory models for movement decoding. Advances in Neural Information Processing Systems; 2010.

37.     Kao JC, Nuyujukian P, Ryu SI, Shenoy KV. A high-performance neural prosthesis incorporating discrete state selection with hidden Markov models. IEEE Transactions on Biomedical Engineering. 2017;64(4):935-45.
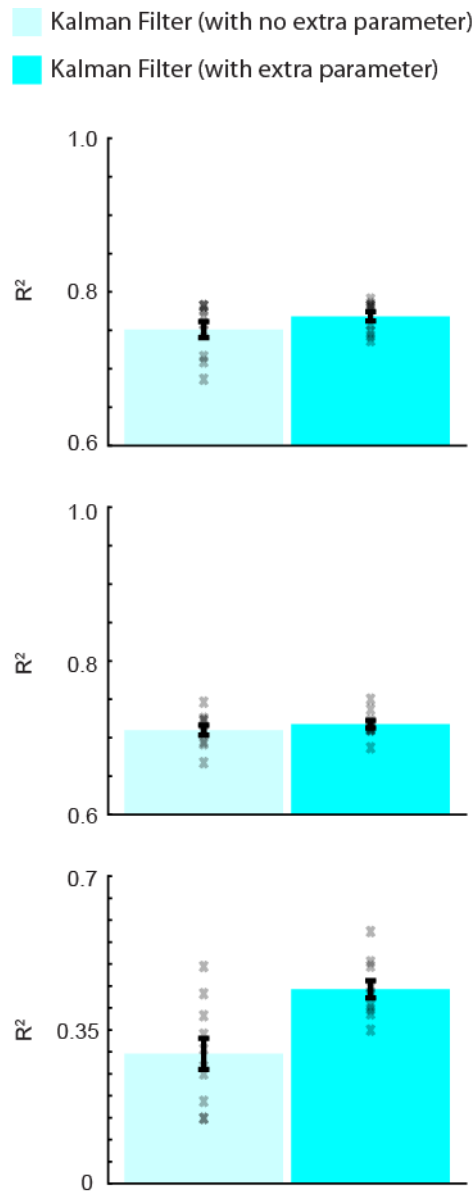
**Supplemental Figure 1. Kalman Filter Versions**
$R^2$ values are reported for different versions of the Kalman Filter for each brain area (top to bottom). On the left (in more transparent cyan), the Kalman Filter is implemented as in [19]. On the right (in more opaque cyan), the Kalman Filter is implemented with an extra parameter that scales the noise matrix associated with the transition in kinematic states (see *Methods*). This version with the extra parameter is the one used in the main text. Error bars represent the mean +/- SEM across cross-validation folds. X's represent the $R^2$ values of each cross-validation fold. Note the different y-axis limits for the hippocampus dataset.