Stabilization of Brain-Machine Interface Systems via Alignment to Baseline

Thesis by

Tara S. Porter

In Partial Fulfillment of the Requirements for
the degree of
Bachelor of Science in Electrical Engineering

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2021

June 11, 2021

Tara S. Porter
ORCID: 0000-0002-2145-9019

ACKNOWLEDGEMENTS

I would like to thank Benyamin Haghi for his dedicated mentorship over the past school year, meeting with me weekly and helping me to find the next direction to explore. This project would not have been possible without his guidance and kindness.

In addition, I give my sincere thanks to Professor Azita Emami for the opportunity to work in her lab with her students, and for trusting me with this excellent opportunity to learn and grow.

Finally, I would like to thank the whole BMI subgroup of the Emami and Andersen labs for their guidance and support each week and throughout the year – Dr. Spencer Kellis, Dr. Tyson Aflalo, Dr. Sahil Shah, Steven Bulfer, and Kelly Kadlec.

ABSTRACT

Research in the brain-machine interface has the potential to transform the lives of individuals with limited motor capabilities to allow for greater independence. By directly accessing signals in the brain, it is possible to train a decoder to identify intended motion and allow the user to control a prosthetic limb or computer cursor by simply thinking about the motion. However, neural data recorded from implanted electrodes is highly unstable over time and across multiple sessions, leading to a severe drop in decoding performance as the test data becomes more distant from the data on which the decoder was trained. Here, we investigate a method to stabilize neural spike data from human trials of a center-out cursor control task before it is passed to a linear decoder, using the techniques of factor analysis and Procrustes alignment. We find that for highly variable human neural data from experiment dates that are far apart, the method does not help the decoder better predict cursor kinematics. However, when factor analysis weights are averaged over multiple baseline days, the performance of the decoder significantly increases with Procrustes alignment, which gives a promising method to limit recalibration and retraining of neural decoders by prolonging their higher accuracy performance over time.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Each year, around 17,900 new spinal cord injuries (SCI) occur. Overall, it is estimated that 300,000 people in the United States are living with such injuries, with only 0.6% maintaining normal motor function. Most patients with SCI struggle with complete or partial paralysis of the limbs (National Spinal Cord Injury Statistical Center, 2021). One proposed mechanism to help patients struggling with paralysis gain more independence is the brain-machine interface (BMI). A BMI refers to a system that is capable of harnessing information from the brain to control an external software or hardware device, such as a computer cursor or prosthetic limb. By tapping into the signals produced by the brain, a whole world of possibilities opens with the potential to allow paralyzed individuals to control assistive technology by simply thinking about it.

Recent discoveries in BMI have pointed to the idea that neural activity lies in a low-dimensional space, termed the neural manifold (Coallier et al., 2015; Degenhart et al., 2020). This surface, or neural manifold, has axes representing the activity of individual neurons within the population, and it is typically obtained using techniques of dimensionality reduction. Dimensionality reduction in the context of BMI refers to a group of techniques that isolate the dominant patterns of covariance between neurons to summarize a majority of the neuronal population activity within fewer variables (Coallier et al., 2015). For example, we can use the technique of factor analysis to reduce the dimensionality of our neural data from 192 dimensions (one dimension for each electrode channel) down to just 25 dimensions that capture the majority of the variability on a lower-dimensional surface contained within the full neural space. This projection of the neural data onto the low-dimensional manifold helps to stabilize decoder performance over time.

Even with dimensionality reduction, a major challenge in the BMI field is the great variation in the recorded neural data from day to day, which makes it difficult to train a decoder on an earlier

day and to maintain accurate performance far into the future. While certain neural network architectures such as recurrent neural networks have been shown to be promising as BMI decoders, there remains the challenge of instability over time due to neuronal death, shifting of electrode positions within the brain, noisy channels, and whether the participant is feeling tired or distracted on a given day. Because of these various factors, we find that one day, a single electrode may be highly correlated with the intended motion. The next day, it may read noise. Naturally, the instability of neural data poses a great challenge for being able to accurately decode cursor kinematics over time. Here, we seek to recover a more stable version of the neural data which, when used to train the decoder, will allow for higher performance over time in decoding kinematics.

In this work, we address the problem of instability using real neural data from two tetraplegic human participants performing a center-out cursor control task (Figure 2). During the experiments, the participants control a cursor's movement from a central circular target on a computer screen to one of eight peripheral targets selected semi-randomly and located equidistantly around a unit circle. The kinematics data for the cursor is recorded over time as positions and velocities in two dimensions (x, y, v_x, v_y), and this data is paired with the implanted electrode readings of neuron firings (neural spike data). We perform our analysis using a linear decoder as a simple model of a BMI decoder, which is trained on the neural spike data using the kinematics data as ground truth labels. We investigate methods to stabilize the decoder's performance over time by preprocessing the neural data before passing it to the decoder. Using factor analysis to reduce the dimensionality of the neural data, we then align the future dataset to a baseline dataset on which the linear decoder is trained. We examine the effect of alignment to a baseline on the ability of the decoder to predict intended cursor movement over time.

Chapter 2

BACKGROUND

2.1 Key Concepts

Brain-machine interface (BMI). Also called a brain-computer interface, this refers to a system that is capable of translating neural data to a software or hardware output (Figure 1). It is often used with assistive technology, allowing individuals with paralysis to control prosthetic limbs or computers using brain signals. Here, we study the use of brain-machine interfaces to decode cursor kinematics from a center-out cursor control task.

Figure 1. Basic building blocks of a brain-machine interface. First, raw neural data is acquired, and features such as spike counts are extracted from it. The features are then stabilized or transformed in some way and passed to the decoder to predict the intended motion based on the neural signal.

The first step of a brain-machine interface system is the acquisition of raw neural data. Implanted electrodes are commonly used to read in signals from neuron firings. Promising new research instead uses ultrasound data from blood flow in the brain as input to a BMI (Norman et al., 2020), though this field of research is quite new. In this study, we stick to electrode data from two 96-channel electrode arrays implanted in the brains of human participants, giving a total of 192 channels of raw broadband data from the brain. The optimal location for placement of the electrode arrays has been the subject of study, but recent work has shown that the posterior parietal cortex (PPC), a brain region responsible for movement planning, contains valuable information to help produce control signals for neural prosthetics (Aflalo et al., 2015) and computer cursor movement (Shah et al., 2019).

The next building block is the extraction of features from raw neural data. Features can be as simple as threshold-crossings or spike counts within each time count (e.g., 50-millisecond bin), or more complex representations of the neural data such as wavelet transforms. In this study, we use neural spike data, which is obtained from thresholding the raw data to count the number of spikes within each bin. The spike count has corresponding velocity data that is used to train the decoder to ultimately predict the kinematics based only on the neural spike data.

The intermediate building block before passing data to the decoder is feature selection. This step is the main focus of our study. We investigate methods of transforming and aligning the neural features before they are decoded in an effort to stabilize the performance of the decoder over time. At this step, we apply factor analysis and the Procrustes algorithm (see below) to reduce the dimensionality of the neural data and align it to a baseline day.

Finally, the data is passed to a decoder that is trained to predict kinematics of the cursor based on the transformed and aligned neural spike data. Choice of decoder is another area with a lot of ongoing research. A standard model for a decoder is the Kalman filter, which linearly models the relationship between the kinematics and neural data (Shah et al., 2019). The most promising decoders have come from machine learning architectures such as deep neural networks (DNNs) and recurrent neural networks (RNNs), including those with Long Short-Term Memory (LSTM), which have proven most effective of all in predicting kinematics over time (Shah et al., 2019). Deep recurrent neural networks also perform well with BMI systems (Haghi et al., 2019; Sussillo et al., 2016). In this study, we use a simple linear decoder to examine the dynamics over time. The end goal of a BMI is to have a trained decoder that is then able to predict intended motion for days after it has been trained based solely on neural data.

Linear decoder. A linear decoder translates neural signals to predicted kinematics using linear regression. In linear regression, data is assumed to have a linear relationship with the label or ground truth. In the case of these experiments, neural spike data u_i from each of the d electrode channels are mapped to the corresponding velocity labels f with learned weights β. The weights are trained using the Python linear regression function, which solves the following equation:

$$f = u_1 \cdot \beta_1 + u_2 \cdot \beta_2 + \cdots + u_d \cdot \beta_d + \epsilon$$

For m timepoints and $d = 192$ electrodes, the dimensions for the kinematics data are $f_{2\times m}$ (containing velocities for both x and y directions at each timepoint) and for the neural data they are $u_{192\times m}$ (containing the voltage reading at each timepoint from all 192 electrode channels implanted in the brain).

It is important to note that for the purposes of this study, we have used a very simplified decoder relying only on linear regression rather than a more sophisticated neural network model. While the simpler model has overall lower levels of decoding accuracy, we are still able to gain insight into the effect of alignment on decoding accuracy by comparing the relative R^2 values over time.

Factor analysis (FA). Factor analysis is a technique of dimensionality reduction. Broadly, dimensionality reduction techniques in the context of BMI are used to find a low-dimensional representation of high-dimensional neural data that condenses the information based on how neuron firing rates covary. Factor analysis more specifically is a linear technique that assumes that latent variables exist to account for correlations among observed variables. With FA, we try to capture the maximum variability of the data with the fewest latent variables, where each variable has as little overlap as possible (Majumdar, 2018). Mathematically, we can define FA with the following equation:

$$u - \mu = \Lambda \cdot z + \epsilon$$

where u is the neural spike data from all electrode channels, μ is the vector of its means, Λ is the matrix of factor analysis weights, or loadings, that define the relationship between the raw data and its latent factors, z is the matrix of latent factors, and ϵ is an error term.

FA takes d-dimensional data and reduces the number of dimensions to a smaller number of dimensions n. For example, our neural data has $d = 192$ dimensions when it is taken directly

from the 192 electrodes, with m timepoints. If we apply factor analysis with $n = 25$ factors, we obtain a new transformed dataset $z_{d\times n}$ from the following:

$$u_{d\times m} - \mu_d = \Lambda_{d\times n} \cdot z_{n\times m} + \epsilon$$

where the neural data u is $192 \times m$, the FA loadings Λ are 192×25, and the latent variables z are $25 \times m$. So, by using the new z rather than u, we have effectively reduced the dimension of the data from 192 to 25 while maintaining most of its variability.

In practice, we use the factor_analyzer package in Python, which learns the weights Λ from u. We can then use the weights to transform u into the latent variables z, and train the decoder on z as the neural data input.

Procrustes problem. The Procrustes problem is an optimization problem that solves for a rotation matrix to align a second matrix Λ_2 to a baseline matrix Λ_1 by minimizing the square of the l_2 norm. Formally, it is defined by the following equation:

$$\hat{O} = argmin_{O:OO^T=I} \left|\left|\Lambda_1 - \Lambda_2 O^T\right|\right|^2$$

Here, $\hat{O} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix with the same dimensions as the number of latent variables n in factor analysis. We test this algorithm using data from the human participants on different days. For example, if we want to align data from day 5 to day 1, we apply factor analysis to data from day 1 and train the linear decoder on the z_1 obtained from FA. Then we want to test the decoder on the future day, so we apply FA to day 5 to obtain the loading matrix Λ_5, and then we rotate the factor analyzer's weights to better align to the baseline using $\Lambda_5' = \Lambda_5 \cdot \hat{O}^T$, where $\hat{O}$ is found using Procrustes with Λ_1 as the baseline. We then use FA with aligned weights to transform the neural data u_5 to the lower dimensional z_5' and input this z_5' to the linear decoder to test its performance in predicting cursor kinematics.

R^2 Accuracy Metric. R^2 accuracy is defined as the square of the correlation coefficient, which corresponds to how well the linear decoder's velocity predictions change with the ground truth. R^2 values range from 0 to 1, with 1 indicating that the prediction perfectly matches the ground truth (or is the exact negative of the ground truth), while a value of 0 indicates that there is absolutely no correlation. For this study, we use a combined R^2 metric to measure the accuracy of the linear decoder; to account for both x and y dimensions, we define the combined accuracy as

$$R^2_{combined} = \sqrt{\frac{(R^2_x)^2 + (R^2_y)^2}{2}}$$

which relates to the average R^2 accuracy from both directions of motion.

2.2 Description of Data

Dataset 1: 2019, Patient 1

Throughout this study, we use neural spike data from 12 days in 2019 (Table 1) taken from two 96-channel electrode arrays implanted in the brain of a tetraplegic patient from Rancho Los Amigos National Rehabilitation Center.

Day Name	Actual Experiment Day (YYYYMMDD)
Day 1	20190125
Day 2	20190215
Day 3	20190314
Day 4	20190402
Day 5	20190507
Day 6	20190625
Day 7	20190723
Day 8	20190806
Day 9	20190820
Day 10	20191008
Day 11	20191115
Day 12	20191217

Table 1. Conversion table showing the day name and the actual date of the experiment during which that day's data was collected from the patient. Throughout the study, we refer to the dates with the designated day name.

Patient 1 is a 54-year-old tetraplegic human research participant. The patient had Utah electrode arrays implanted into the hand-knob of the motor cortex and the superior parietal lobule of the posterior parietal cortex (Neuro-Port, Blackrock Microsystems). Both arrays had 96-channels each, giving a total of 192 channels from which broadband data was sampled at a frequency of 30,000 samples per second.

During the experiment, the participant performed the center-out task where they moved a cursor in the x-y directions on a computer screen outward from a central target to one of 8 outer targets situated around a circle and back to the center (Figure 2). Each trial is defined to be one trajectory outward to a target or inward back to the central target. The cursor's movement across the screen is updated every 30 milliseconds, and the patient typically imagines moving the cursor for blocks of around three minutes. Trajectories were extracted from the trials at the point 200 milliseconds after the target had been presented up until 100 milliseconds before the cursor overlapped with the target in an attempt to isolate when the participant's intent was most well-defined. Neural features were then regressed against cursor velocity, which was modeled as a constant for simplicity. Data for this research participant was collected over 22 sessions containing 44 blocks of trials. 12 of these sessions (scattered across 2019) were used in the following analysis.

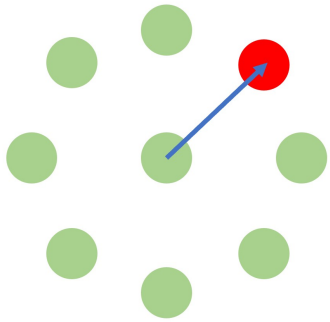


Figure 2. Center-out cursor experiment setup. Semi-randomly selected outer targets light up in red, and the participant thinks about moving the cursor from the central target to the outer target roughly along the blue trajectory, for example.

After an initial period of calibration during which the computer controls the cursor to move to different targets while the participant imagines controlling it, the research participant then

switches to controlling the cursor themself with signals from the brain as they imagine moving a joystick in the direction of the target, for example. From the calibration, it is possible to decode the movement intention from brain signals in the short timescale of the experiment so that the participant is able to control the cursor in the open loop. It is signals from these open-loop trials when the participant was controlling the cursor that we use in our analysis to explore methods of maintaining decoder performance over longer timescales.

Data Preprocessing

Very little is done to the raw data in terms of preprocessing. The raw neural data (.NEV) from the electrodes is simply processed to have zero mean and a standard deviation of 1, and then the NEV file is converted to a MATLAB file containing all the trials for a particular day, along with the kinematics of motion (velocities in the x and y direction corresponding to each timepoint of neural signal recordings). The MATLAB file is then read into Python and the neural spike data ($\boldsymbol{u}$) and kinematics data ($\boldsymbol{f}$) are used in the following analysis as the decoder input and ground truth, respectively.

2.3 Foundational Work: Degenhart et al., 2020

The research presented in this thesis is based on the methods of a previous study to stabilize a BMI decoder over time (Degenhart et al., 2020). The authors of the study proposed a method of stabilizing neural threshold crossing data using factor analysis and alignment of neural manifolds via the Procrustes algorithm. An overview of the methods is presented below.

The authors recorded neural data from a 96-electrode array (Blackrock Microsystems) implanted in the primary motor cortex of two male Rhesus macaques. The monkeys performed a center-out cursor task during which they were given 7.5 seconds to reach a target selected from 8 screen locations with the cursor.

A manifold-based stabilization method was then proposed for improving the stability of neural signals over time. The researchers leveraged the tendency of neural signals to lie in a lower-

dimensional manifold. By applying techniques of dimensionality reduction, it is possible to map the raw neural threshold crossings data to a lower-dimensional latent space. In this study, the technique of factor analysis was used.

For factor analysis, the latent space variable z is assumed to be normally distributed, with a matrix of weights Λ that map the latent variable back to the original neural data u such that $u - \mu = \Lambda \cdot z + \epsilon$. This study uses $n = 10$ factors, meaning that the latent variables $z \in \mathbb{R}^{10}$. Furthermore, for q electrodes recording the neural data contained in u, we have that $u \in \mathbb{R}^{q}$ and thus $\Lambda \in \mathbb{R}^{q \times 10}$.

This work used $q = 75$ electrodes in non-overlapping 45 ms bins. The neural data u was formed from threshold crossings over these electrodes. Then, given the weights matrices Λ_1 and Λ_2 obtained from performing factor analysis on two different blocks of trials, the researchers used the Procrustes algorithm to align the coordinate systems of the two manifolds. The Procrustes method solves the following optimization problem for $\hat{O} \in \mathbb{R}^{10 \times 10}$:

$$\hat{O} = argmin_{O:OO^T} \left|\left|\Lambda_1(s,:) - \Lambda_2(s,:)O^T\right|\right|_F^2$$

where s represents the indices of stable electrodes used in the study. After calling the Procrustes function, the second day's weights are aligned to Λ_1 by multiplying by the rotation matrix. So the newly aligned weights are $\Lambda_2' = \Lambda_2 \cdot \hat{O}^T$.

To identify the stable electrodes making up s, the authors describe two main steps. First, they manually removed channels with l_2 norms in either day that were less than a predefined threshold T. To do so, they took the l_2 norm of Λ_1 and Λ_2 (where each row corresponds to an individual channel) and removed the channels from consideration if that row of the norm from either day was below 0.01 counts per bin. $T = 0.01$ was set to be larger than the smallest observed norm for low-noise electrodes.

Next, the researchers iteratively removed channels from consideration in order of how poorly they aligned with the baseline day 1's weights matrix, Λ_1. First, they chose the number of channels B that they wanted to keep as the largest possible value that excluded electrodes with severe instabilities ($B = 60$ for this study). Next, they iteratively performed Procrustes alignment of day 2 with day 1, each time removing the channel from s based on the row that had the largest l_2 norm entry in $\Lambda_1(s,:) - \Lambda_2(s,:)\hat{O}^T$. In other words, they removed the channel that was furthest from Λ_1 after alignment had been performed.

In this study, the authors manually added instabilities to their data in the form of unit dropouts, baseline shifts, and tuning changes. Due to the simpler nature of having used predictable simulated instability data, we chose to investigate the efficacy of the above-described stabilization method on real human data, with real instabilities arising from physical phenomena such as electrodes shifting in the brain, neurons dying, and thoughts and neural signal strength changing from day to day. These signals can be much less stable and more difficult to decode, especially over long distances in time. In our study, we test whether the methods of Degenhart et al. (2020) work with our unstable human data.

Chapter 3

DECODING KINEMATICS WITH BASELINE ALIGNMENT

3.1 Preliminary Analysis

Given that the stabilization methods of Degenhart et. al (2020) namely rely on single channels considered individually and removed iteratively, we performed some preliminary investigations on our real human data to understand how channels compared to one another.

Single-Channel R^2 Analysis

To begin, we performed an analysis of the accuracy of each individual electrode channel. To do so, we trained a linear decoder in Python on data from each of the 192 channels separately for a given day. We used 88% of the data to train and 12% to test. We then calculated the testing data R^2 value with five-fold cross validation. The errors for x and y directions were calculated separately for each channel.

We found that the R^2 values across all channels for a given day do not typically reach a value higher than 0.2, while most channels' R^2 values are clumped closer to zero (Figure 3). These results indicate that a single channel does not contain very much predictive power, but certain channels have more ability to predict cursor kinematics than others on a single day.

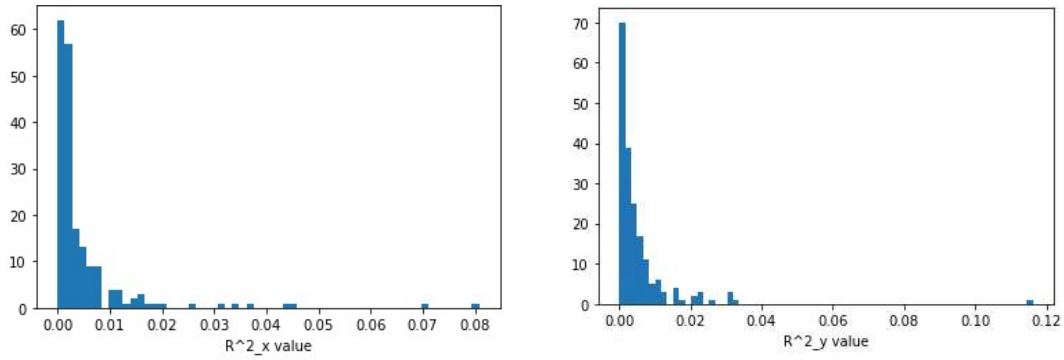


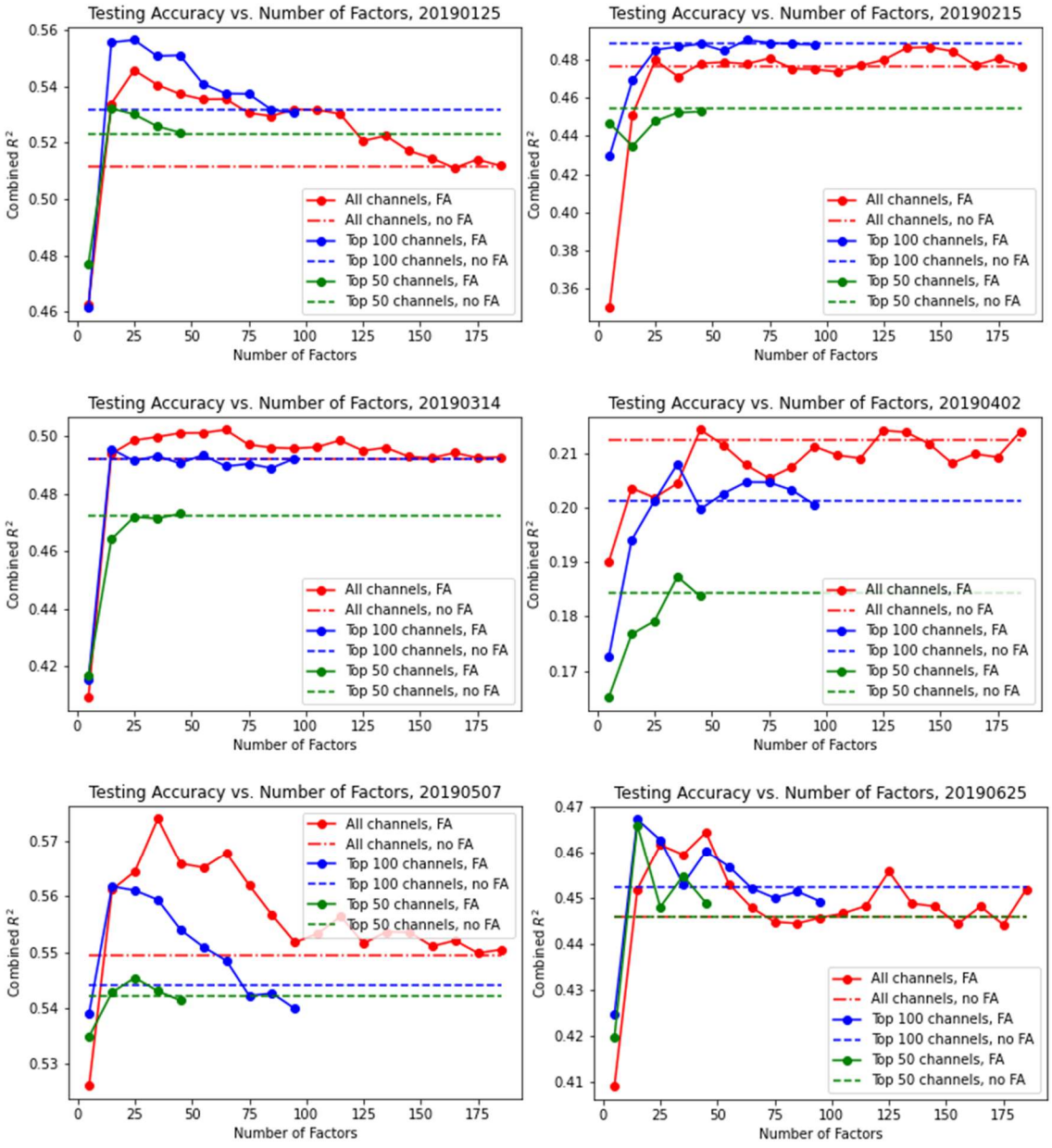
Figure 3. An example distribution of R^2 values from single channels on a single day in the x-direction (left) and y-direction (right).

At this point, it is worth investigating that some channels are stronger than others at predicting kinematics, which could potentially support the method of iteratively removing weaker channels from the data before training the decoder as described in the foundational work.

Factor Analysis with Channel Selection

Next, we investigated the effect of factor analysis with different numbers of factors n (with n ranging from 5 to 185) on the accuracy of the linear decoder. As a comparison, we first directly trained a linear decoder on individual days with raw neural spike data (u) paired with v_x, v_y labels of velocity kinematics data (f). We then applied factor analysis with varying numbers of latent factors to transform the spike data from the 192 channels to latent variables z (where $u - \mu = \Lambda \cdot z + \epsilon$), and trained the linear decoder with z instead of u, labeled with velocities f.

The above process was repeated in three different scenarios: first, passing all 192 channels to the factor analyzer. Second, passing only the top 100 channels. And finally, passing only the top 50 channels (Figure 4). Channels were ranked based on their single-channel R^2 values obtained from training the linear decoder on each channel individually. The channels were all sorted by their R^2 values, and those with the highest accuracies were selected as the top M channels (for $M = 100$ and $M = 50$). Accuracy was measured as the combined R^2 value from both coordinate directions based on the linear decoder's predictions on a subset of 12% of the data for each day.


Testing Accuracy vs. Number of Factors, 20190125
Testing Accuracy vs. Number of Factors, 20190215
Testing Accuracy vs. Number of Factors, 20190314
Testing Accuracy vs. Number of Factors, 20190402
Testing Accuracy vs. Number of Factors, 20190507
Testing Accuracy vs. Number of Factors, 20190625
Combined R^2
Number of Factors
All channels, FA
All channels, no FA
Top 100 channels, FA
Top 100 channels, no FA
Top 50 channels, FA
Top 50 channels, no FA

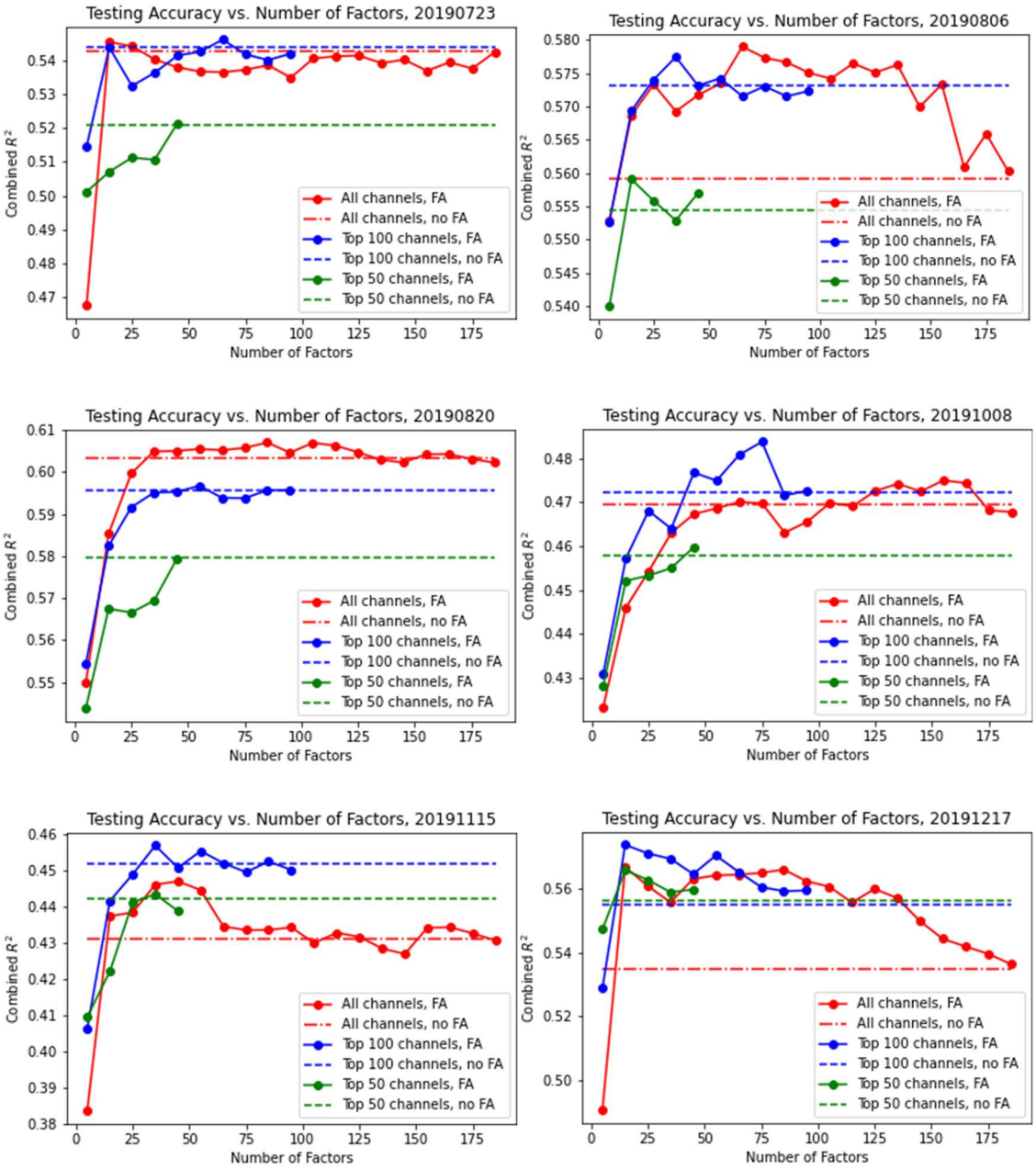


Figure 4. Graphs of each of the 12 days in 2019 used in this study. The graphs show the accuracy of the linear decoder trained on the latent variables z obtained from factor analysis on the top M channels with varied number of factors n from 5 to 185. The horizontal lines show the basic threshold that factor analysis must cross in order for the linear decoder trained from latent variables to perform as well as the one trained on raw spike data u.

The results indicate that much of the information that is important for decoding the kinematics is contained in a lower number of channels than all 192. In particular, most days show that the

top 100 channels perform equivalently or even better than all 192 channels after factor analysis. This would suggest that some of the channels negatively impact the accuracy of the decoder, and it may then be beneficial to remove channels such as in the approach of Degenhart et al. (2020). Furthermore, when all channels are used, this analysis suggests that 25 factors are sufficient in almost all cases to achieve or exceed the limit of accuracy when no factor analysis is performed to reduce the dimensionality of the data. Finally, we conclude that 50 channels would not be sufficient to capture enough information to decode the motion at the highest possible level.

In summary, in this section we found that some channels are more important for decoding the kinematics than others, and that $n = 25$ factors is sufficient for most of our 12 days in 2019 to maintain or exceed the performance of the linear decoder with no factor analysis. Furthermore, the most important information is contained in a smaller subset of the channels than all 192, making channel removal a potentially feasible method of stabilization. Using the optimal parameters that we determined above, we can now test the methods of Degenhart et. al (2020) directly on our data.

3.2 Replication of Foundational Work

We tested the methods of Degenhart et al. (2020) to stabilize the decoder with factor analysis and Procrustes alignment on our human data. If the alignment works, we expect that after channel removal by the iterative Procrustes alignment algorithm presented in the foundational work, the decoder will perform better over time than without alignment.

Channel Removal by Thresholding

The first method described by Degenhart et al. (2020) for identifying the most stable channels in the data involves manually removing channels for which the l_2 norm within the FA weights matrix is below the threshold $T = 0.01$. We took the norm of each individual row in Λ_1 and Λ_i for $i \in [2,12]$. In the foundational procedure, any row for which this norm was below T in

either weights matrix would be removed from consideration (where each row corresponds to a channel, i.e. an electrode). Across all 12 days in our dataset, we found that the minimum norm was 0.06345, while the maximum norm was 0.8859. While a smaller l_2 norm value could indicate that the channel simply contains noise or does not read any real signals across days, we chose not to remove any channels at this point since their norms were not as small as the threshold $T = 0.01$ used in the foundational work. We also decided to take data on all channels rather than trying to remove them at first in order to better understand the dynamics.

We plotted the distribution of single-channel l_2 norms to find a suitable threshold for our specific dataset (Figure 5). Rather than the original paper's threshold of $T = 0.01$, perhaps a more suitable threshold would be 0.2 or 0.25 for manual channel removal by thresholding.

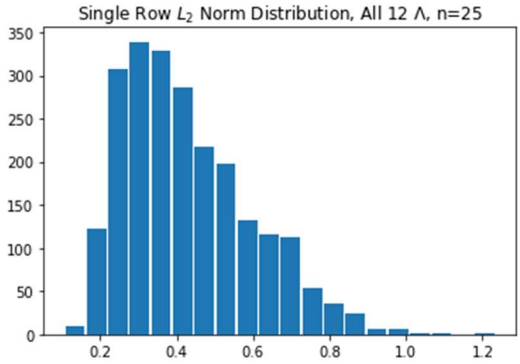


Figure 5. Histogram of l_2 norms of individual rows in all 12 weights matrices Λ obtained from factor analysis with $n = 25$ factors. The distribution suggests that a better threshold for manual channel removal may be closer to $T = 0.25$ rather than $T = 0.01$.

Procrustes Alignment of Channels to a Baseline

The second step in stable channel selection outlined by Degenhart et al. (2020) involves alignment of the second day's manifold to the baseline day using the Procrustes algorithm. To do this, we fit a factor analyzer to the first day to obtain the weights Λ_1. We then fit a factor analyzer to a second day i to obtain Λ_i. Next, we called the SciPy Procrustes function to obtain the newly aligned weights $\Lambda_i' = \Lambda_i \hat{O}^T$ that are aligned to the coordinate system of Λ_1. We first tried using $B = 60$ channels as the number of top channels to keep that are best aligned to the baseline day. We also tried $B = 100$ (Figure 6).

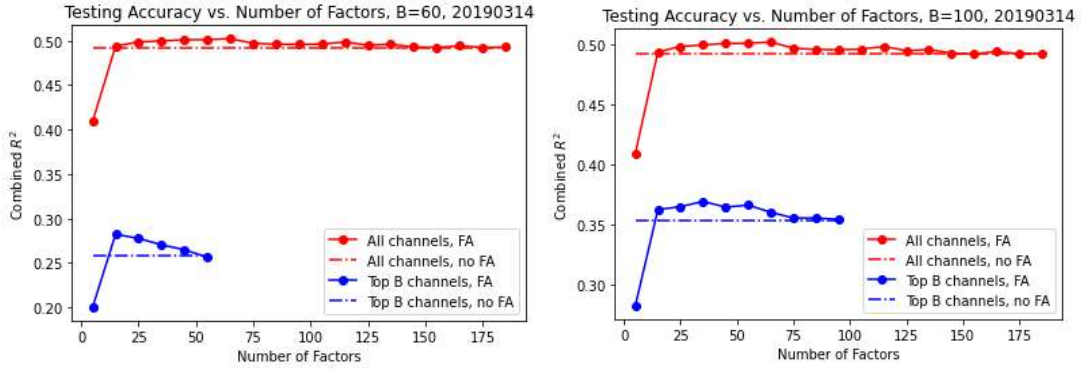


Figure 6. Combined R^2 accuracy after iterative channel removal to the top B channels via Procrustes alignment for $B = 60$ channels (left) and $B = 100$ channels (right).

Neither method performed as well as when all 192 channels were used. We decided to further investigate the individual channels to better understand why this was the case, since based on the foundational work, we would expect that after iterative channel removal the performance would improve.

3.3 Investigating the Effect of Alignment

We investigated the tradeoff between channel stability over time and accuracy of decoding. To do so, we looked at the l_2 norm of individual channels within the weights matrix Λ' after it had been aligned to a baseline via Procrustes alignment and subtracted from the baseline weights.

Comparison of Stability from Different Baseline Alignments

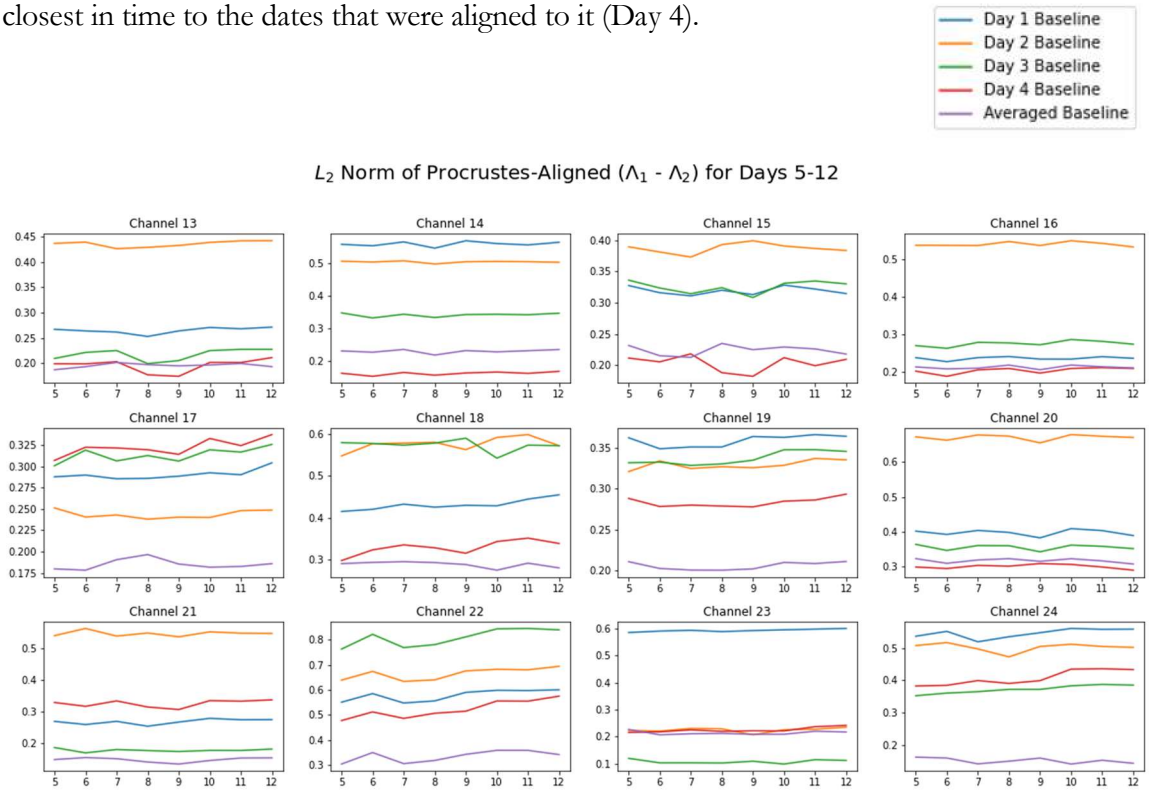
First, we looked into which baseline day had the best results in stabilizing and aligning a future date. We used a baseline selected from Day 1, Day 2, Day 3, Day 4, or the average of the first 4 days, and aligned each of Days 5 to 12 to this baseline via the Procrustes algorithm. To do so, we fit a factor analyzer in Python to the neural data from the baseline day, and another factor analyzer to the test day. We then called the Procrustes algorithm to retrieve a set of modified weights for the test day that had been aligned to the baseline day. Next, we calculated the difference between the aligned test weights and the baseline weights, $\Lambda_i' - \Lambda_{base}$ and took the

l_2 norm to measure the degree of alignment between the two days. For the average baseline, we took the average of all 4 baseline days as:

$$\frac{\sum_{b=1}^{4} \Lambda_b}{4}$$

where each Λ_b was obtained by fitting a factor analyzer to neural spike data from each baseline day b.

We then plotted the l_2 norm of the difference between the aligned weights matrix and the baseline weights for each of the 192 channels (channels correspond to individual rows in the weights matrices) for each day in 5 to 12 (Figure 7). We found that the baseline that minimized the channel norm of the weights difference was the average baseline, followed by the baseline closest in time to the dates that were aligned to it (Day 4).



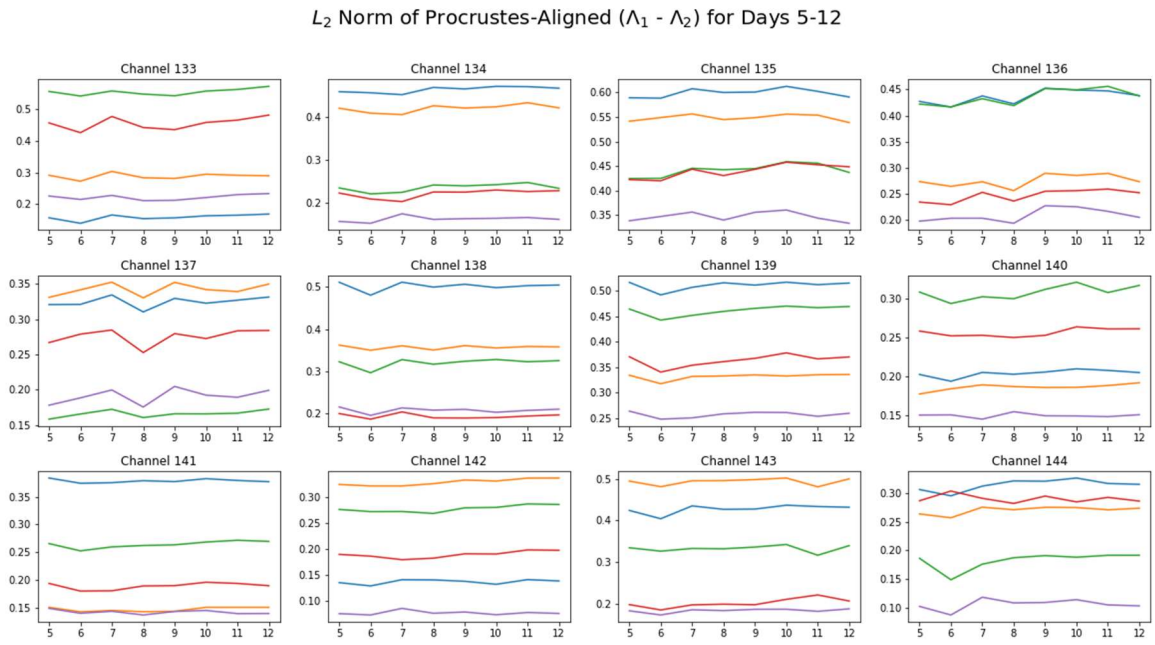


Figure 7. Example of channel norms over Days 5 to 12 when aligned to single-day baselines (Days 1-4) and the average baseline. These graphs were generated for all 192 channels.

We found that the norms of the difference between the baseline weights and the aligned test day weights were consistently lower when we used the average baseline. Among single-day baselines, we found that those days closest to the test day (Day 3 and Day 4) were most frequently the lowest norm among the channels (Figure 8).

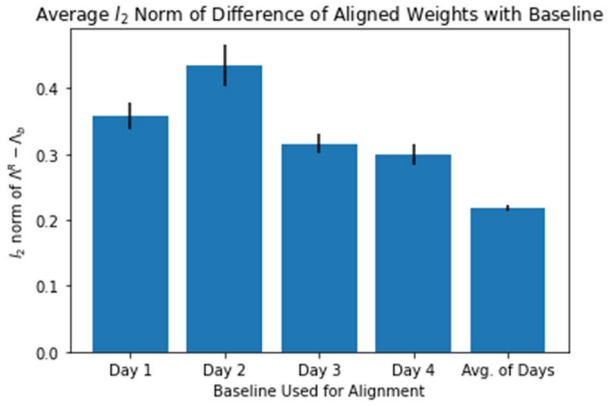


Figure 8. Summary of Figure 7 data, showing the average l_2 norm of the rows of $\Lambda_i' - \Lambda_b$. This is the average norm across the 192 neural channels across all 8 test days 5 to 12. A low l_2 norm could indicate a higher degree of alignment between the Procrustes-aligned test weights and the baseline.

A low l_2 norm could indicate a higher degree of alignment between the Procrustes-aligned Λ_i (where $i \in [5,12]$) and the baseline day Λ_b (for $b \in [1,2,3,4, Avg.1{:}4]$). Excluding Day 1, we see a downward trend in average norm across channels and test dates as the baseline day becomes closer in time to the test dates. This indicates that it may be easier to align and stabilize weights across days when the days are closer together in time.

Comparison Between Stability with Aligned vs. Unaligned

Next, we investigated whether the Procrustes alignment stabilized the channel difference l_2 norm across the test days (Days 5 to 12) compared to the norm of the difference without any Procrustes alignment. For this part of the analysis, we chose to use the average baseline data since it had performed best compared to all the baselines tested in the previous section. Applying a similar process to the previous analysis, we obtained the average baseline weights $\Lambda_{1:4}$ from factor analyzers fit to the neural data from Days 1 to 4 individually. We fit another factor analyzer to each test day in 5 to 12 to obtain the weights matrices Λ_i for $i \in [5,12]$.

We then calculated the l_2 norm of the difference $\Lambda_{1:4} - \Lambda_i$ for the unaligned version and $\Lambda_{1:4} - \Lambda_i'$ for the aligned version, where Λ_i' is the weights matrix from the factor analyzer fit to day i's neural spike data after the weights have been aligned via the Procrustes algorithm to the average baseline weights. We compared the results of aligned vs. unaligned l_2 norms in the graphs below (Figure 9).

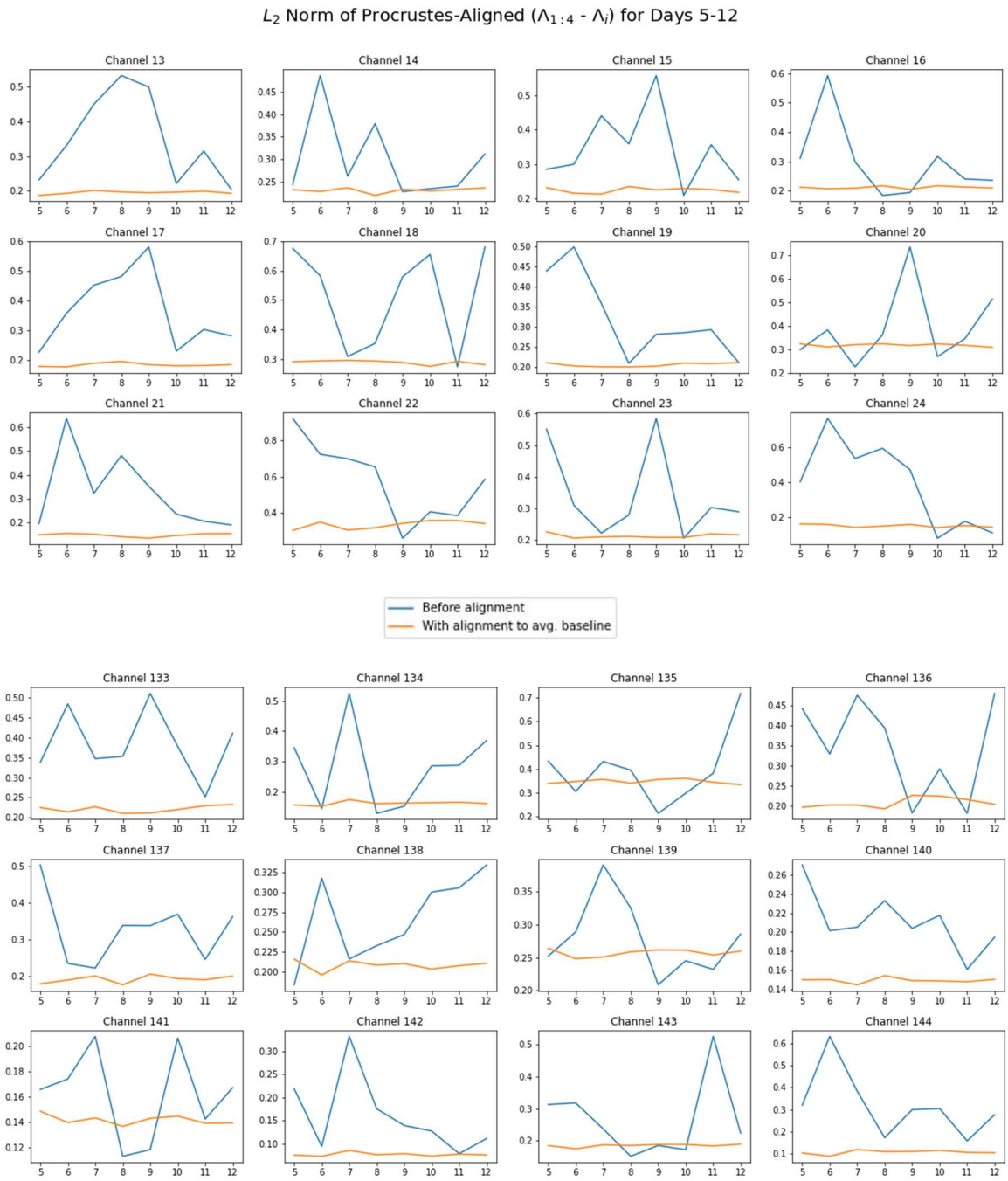


Figure 9. Comparison between the l_2 norm (y-axis) of the unaligned difference of weights matrices versus aligned difference of weights across test days 5 to 12 (x-axis). The Procrustes alignment tends to stabilize the l_2 norm across days and minimize its value compared to the unaligned norms, which suggests a higher degree of alignment to the baseline.

We found that the Procrustes alignment visibly reduces the variability across days of the l_2 norm for each channel, compared to the variation that we see in with the unaligned norms. In addition, in many cases the stabilized, aligned norm has a lower average value than the unaligned norm. This suggests that the Procrustes alignment not only serves to stabilize the weights across days, it also minimizes the difference between the baseline and corresponding aligned weights compared to the unaligned difference, which could lead to greater stability in performance after alignment.

R^2 *Performance of Channels Across Test Days*

We investigated the effect of individual channels on decoding accuracy across all 8 test days (Days 5-12). The linear decoder was trained on data from the single channel on each day individually. We then plotted the R^2 accuracy for each channel, on each day (Figure 10). We observed that while many channels remained consistently poor at predicting the kinematics behavior across the test days, certain other channels seemed to have spikes in accuracy on a few specific days, but did not remain consistently significant for predicting kinematics behavior across all days. This suggests that at this timescale of days, where in reality there is an average of 30 days between each "Day," any given channel will not necessarily be a good channel across multiple consecutive "Days." Those channels which are not noise channels only have certain high-R^2 days interspersed across all the days. This suggests that the approach of Degenhart et al. (2020) of removing channels that are not well aligned to the baseline is likely to fail at large timescales, since channels do not have stability across days in terms of decoding accuracy but rather have sporadic spikes in R^2.

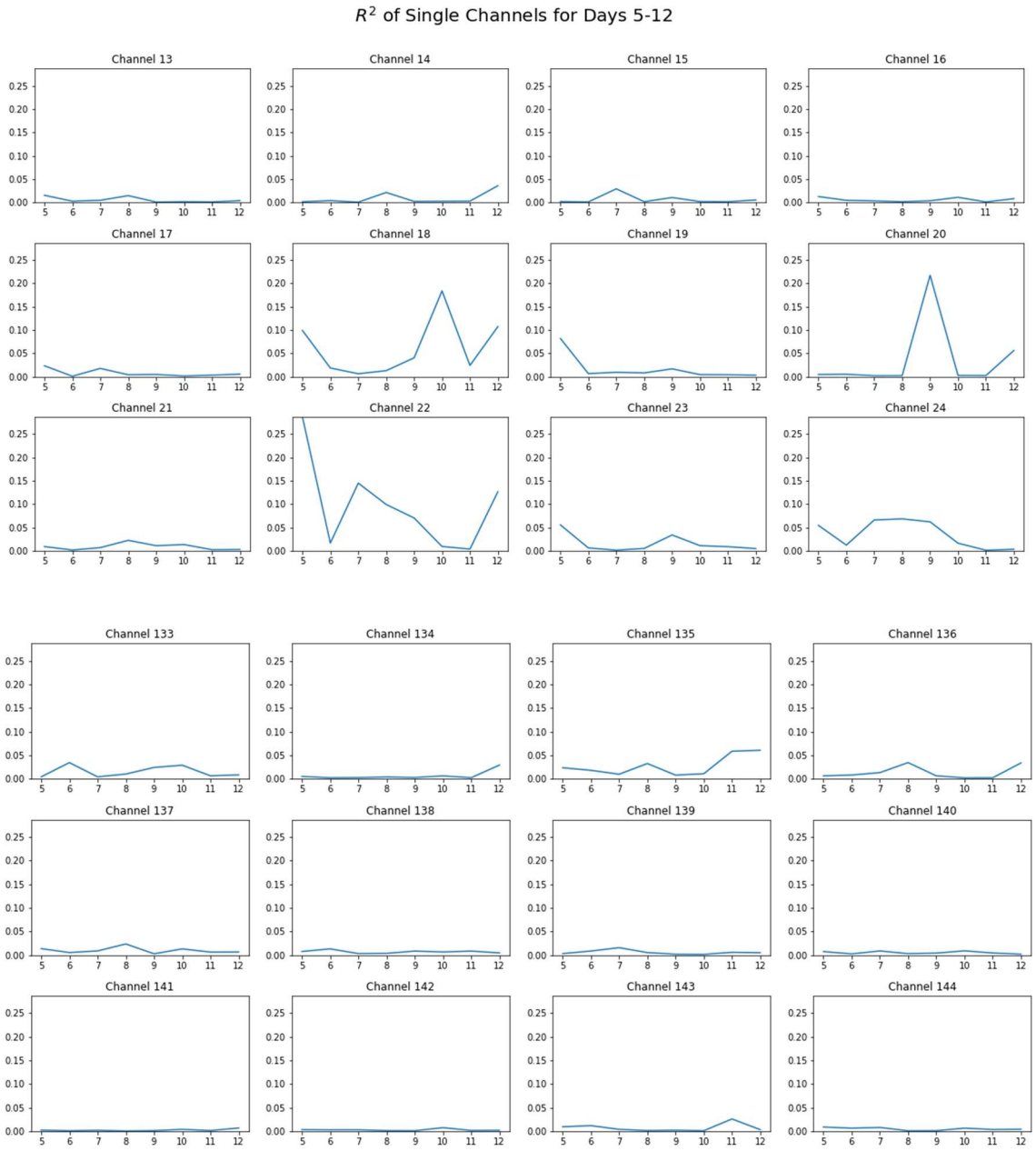


Figure 10. Example graphs showing the R^2 accuracy of the linear decoder when trained on a single channel across Days 5 to 12 (the test days). Each graph represents the performance from a single channel and its significance in helping the linear decoder to decode kinematics on each day.

We can see that a lot of channels have almost no impact on the decoding across all days, while others seem to spike only for certain days. Very few remain consistently significant. Based on the above results, we find that a bad channel on one day can easily be one of the more important

channels on the next day, so it could be detrimental to remove this channel from the dataset used to train or test the decoder.

Accuracy-Stability Tradeoff

Next, we examined the tradeoff between stability and accuracy by comparing the top channel in accuracy and the top channel in stability. The channel with the maximum single-day R^2 accuracy was Channel 22 (accuracy around 0.29 on Day 5). This channel also has a comparatively high level of linear decoding accuracy on other days, including Day 7 and Day 12. However, the l_2 norm between test days and baseline weights from the average baseline (Figure 11 in purple) is at a higher value than it is for other channels (Figure 7), indicating that the channel is not as well aligned to the average baseline as other channels are.

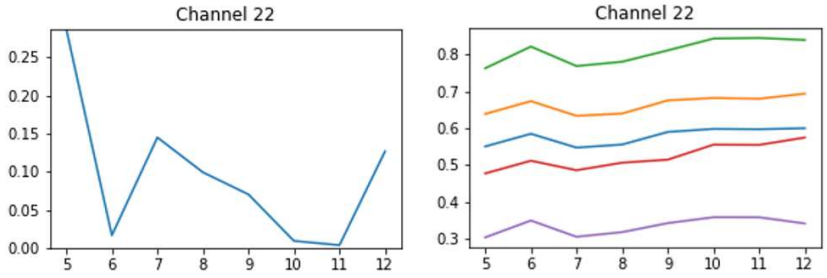


Figure 11. R^2 accuracy (left) vs. l_2 norm aligned to different baselines (right) across all 8 test days (Days 5-12). Channel 22 has the highest R^2 single-channel decoding accuracy, but the l_2 norm indicates a relatively low degree of alignment to the baseline across all test days with a norm around 0.3.

On the other hand, the channel with the lowest average l_2 norm across test days aligned to the average baseline was Channel 142 (Figure 12). Although the channel seems to have a high degree of alignment to the baseline and stability across the test days, it is clear from the R^2 graph that this channel is merely noise and contains almost no real information to help the linear decoder accurately decode kinematics across the test days.

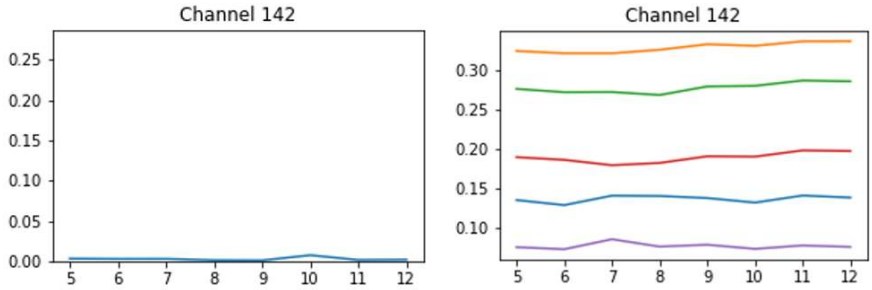


Figure 12. R^2 accuracy (left) vs. l_2 norm aligned to different baselines (right) across all 8 test days (Days 5-12). Channel 142 has the lowest average baseline l_2 norm across all test days, indicating the highest degree of alignment between the baseline and each test day, or in other words, the highest level of stability. However, the R^2 accuracy indicates very low significance of the channel in kinematics decoding. The channel may simply contain noise.

Based on the two example channels considered in this section, there seems to be a tradeoff between stability and accuracy of channels over time. Although the channel with the highest R^2 accuracy contains useful information for the decoder to predict kinematics based on neural spike data, it has poor stability and alignment to the average baseline. Similarly, although the channel with the lowest l_2 norm in $\Lambda'_{test\ day} - \Lambda_{base}$ has a high level of stability and alignment to the baseline, it has very poor accuracy performance and appears to only contain noise that is irrelevant to the decoding of kinematics.

Effect of Alignment to Baseline on Training

To verify whether the Procrustes alignment helps the decoder to perform better over time, we compared the performance of the linear decoder in four distinct scenarios, all with factor analysis using $n = 25$ factors:

1. **Trained and tested on same test day, no alignment to baseline day:** The linear decoder was trained on 88% of the factor-analysis-transformed z for a given day ($u_{192} - \mu = \Lambda \cdot z_{25} + \epsilon$), and the linear decoder predictions were tested on a different segment of 12% of the remaining data from that day that was not used for training.

2. **Trained and tested on same test day, with alignment to baseline day:** The linear decoder was trained on the Procrustes-aligned version of the transformed z for a given

day, which had been aligned to the weights of the baseline day. The decoder was again tested on the remaining data for the same day.

3. **Trained and tested on different days, no alignment to baseline day:** The linear decoder was trained on the factor-analysis-transformed data from the baseline day and tested on the test days (Day 5 to 12) data that had been transformed with its own factor analyzer (no alignment to the baseline day).

4. **Trained and tested on different days, with alignment to baseline day:** The linear decoder was trained on the transformed z_{train} of the baseline day and tested on the test day after alignment to the baseline. To align to the baseline, a factor analyzer was first fit to test day spike data, and the weights were extracted. The Procrustes algorithm was then called in Python to align the test weights to the baseline day weights. The new, aligned test weights were then set as the weights of the factor analyzer before transforming the test data to z'_{test} and testing the performance of the linear decoder that had been trained on the baseline.

Using Day 1 as the baseline, we trained and tested the linear decoder according to the four scenarios described above (Figure 13).

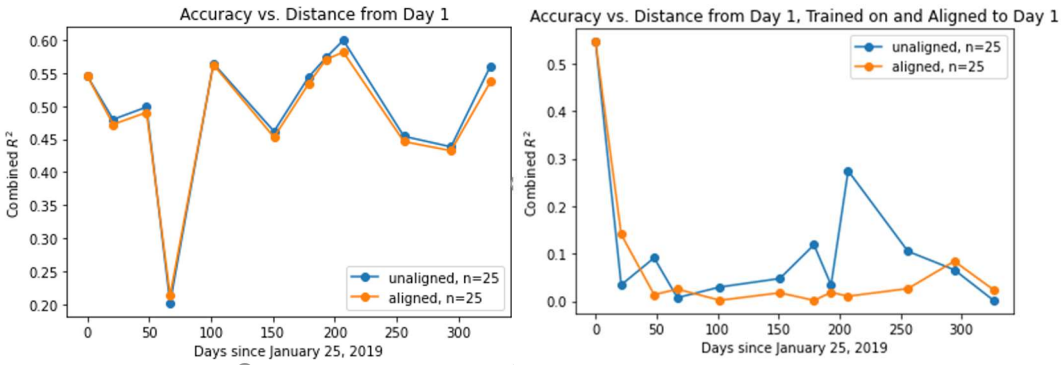


Figure 13. Left: Scenario 1 is shown in blue, while scenario 2 is shown in orange. When the linear decoder is tested and trained on the same day, it tends to perform much better than when it is trained on an earlier day than when it was tested. The process of aligning the weights to the baseline day has little to no effect. Right: Scenario 3 is shown in blue, while scenario 4 is shown in orange. When the linear decoder is trained on Day 1 and tested on the later days, the unaligned test days tend to perform at higher accuracy than those aligned to the baseline day, though both scenarios give extremely low accuracy on most days.

Based on these 12 days from the year 2019, it is evident that it infeasible to train the linear decoder on any day besides the day it is being tested on for this particular dataset. Even with the alignment of the test day's weights to the baseline weights before the factor analysis transform is applied, there is almost never an improvement in the decoding accuracy if the linear decoder has only been trained on the baseline day. In fact, on most days, the transformed data that has not been aligned to the baseline weights performs better than the aligned data (Figure 13, Right). The only day in which the aligned data performs significantly better than unaligned data is on Day 2, which is the closest day to the baseline. This suggests that the Procrustes alignment method could help the decoder to more accurately decode kinematics based on neural signals over time provided that the time difference between the days is relatively small. When the decoder was trained on Day 1 and tested on Day 2 data that had been aligned to Day 1 via Procrustes, we do see a notably higher performance than the test on Day 2 without alignment. Still, the accuracies in all cases when the decoder was trained and tested on a different day are very low, and it is difficult to come to a conclusion about the effect of the Procrustes alignment based only on this dataset, where days are distributed so far from each other.

Effect of Distance Between Days

We compared the 12 days with each other pairwise to visualize the distribution of distances between all the "Days." The distribution of the distances between each combination of days is plotted below (Figure 14). The average distance between consecutive "Days" is in reality 29.63 days.

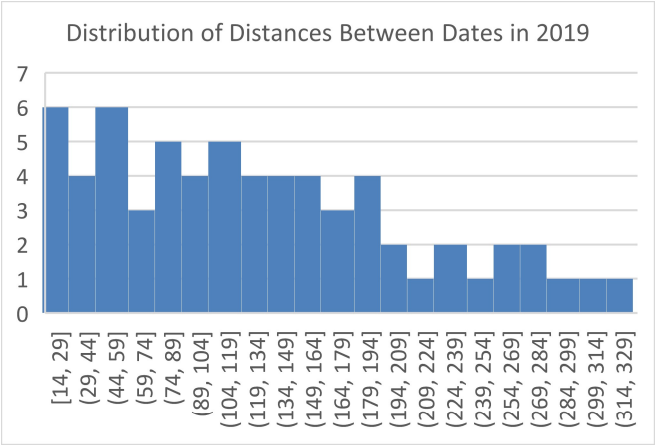


Figure 14. Histogram distribution of distances (in days) between each pair of dates in the 12 2019 data days. The mean distance between consecutive days is 29.63.

Based on the distribution, we split up the dates into the following groupings for further analysis (Table 2).

14-19 days		**21-28 days**		**32-42 days**		**46-56 days**		**77-87 days**		**101-105 days**	
14	Day 7 to 8	**21**	Day 1 to 2	**32**	Day 11 to 12	**46**	Day 2 to 4	**77**	Day 5 to 7	**101**	Day 8 to 11
14	Day 8 to 9	**27**	Day 2 to 3	**35**	Day 4 to 5	**48**	Day 1 to 3	**77**	Day 7 to 10	**102**	Day 1 to 5
19	Day 3 to 4	**28**	Day 6 to 7	**38**	Day 10 to 11	**49**	Day 5 to 6	**81**	Day 2 to 5	**103**	Day 3 to 6
		28	Day 7 to 9	**42**	Day 6 to 8	**49**	Day 9 to 10	**84**	Day 4 to 6	**105**	Day 5 to 9
						54	Day 3 to 5	**87**	Day 9 to 11	**105**	Day 6 to 10
						56	Day 6 to 9				

Table 2. Groupings of "Days" in 2019 with the corresponding distances (in actual days) between them.

For each pair of dates (Table 2), we trained the linear decoder on the first date and tested it on the second date, both in the unaligned case (scenario 3 above) and the aligned case (scenario 4 above). For the unaligned case, we trained the linear decoder on the transformed neural data (z) from the baseline day and tested it on transformed neural data from the test day that had been transformed by a separate, independent factor analyzer that had not been aligned to the baseline in any way. For the aligned case, we again trained the linear decoder on the baseline transformed data but then tested it on the test data that had been transformed using new factor analyzer weights that had been aligned to the baseline weights via the Procrustes function. Within each grouping of day ranges, we averaged the performance of the linear decoder and plotted the R^2 accuracy with error bars showing the standard deviation within each range (Figure 15). The size of each point is proportional to the number of days within that range, which respectively is [3, 4, 4, 6, 5, 5].

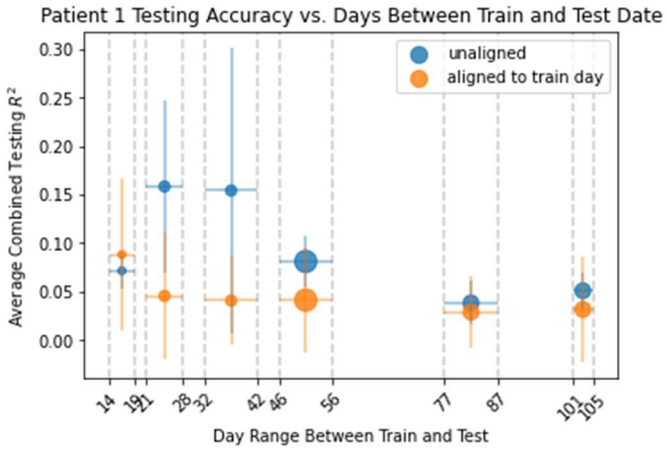


Figure 15. The R^2 accuracy performance of the linear decoder, trained on a base day and tested a number of days later, where the range of days on the x-axis shows the distance between train and test days. The points show the average accuracy within each range of dates, where the point size represents the number of dates included in each range. The blue points show the performance when the test day was transformed using factor analyzer weights that had not been aligned to the training day, while the orange dots show the performance in testing accuracy when the Procrustes alignment was applied to the weights to align them to the baseline training day.

Again, we found that the Procrustes alignment to a train day does not tend to help the linear decoder predict on future test dates. In most cases, the aligned version performs worse on average than the unaligned version, except within the smallest range of distances (14-19 days between train and test date). This suggests that either the Procrustes alignment process does not work to stabilize real neural data, or our dataset is not conducive to the method of Degenhart et al. (2020) since the data acquisition dates are distributed so far apart, and the physical changes to the brain and electrode positioning could lead to a huge lack of stability and reliability between most dates.

Final Attempt: Replication of Foundational Methods with Channel Removal

As a final test, we replicated the methods of the foundational paper using the optimal parameters we found during our previous analysis: for the 12 dates, we found that $n = 25$ factors is sufficient to meet or exceed the R^2 performance of using all 192 channels' data (Figure 4), compared to the 10 factors used in the paper for 70 channels. Furthermore, using factor analysis with $n = 25$ factors, we found that a basic threshold of $T = 0.25$ should be sufficient to manually remove channels with little activity (compared to $T = 0.01$ in the foundational work).

We chose to use this much larger threshold for manual channel removal since our real data has many more instabilities and noise than the simulated data used in the original paper. We used Day 1 as the baseline day and Days 2 through 12 as the test days. For each of the 12 days, we calculated the l_2 norm of their loading matrices Λ and removed any corresponding channels whose l_2 norm rows were lower than the manual threshold $T = 0.25$. Next, we iteratively removed channels down to $B = 140$ total channels, at each step removing the channel whose l_2 norm (corresponding to a row in the matrix $\Lambda'_{test} - \Lambda_{base}$) was highest, which indicates the lowest degree of alignment to the baseline. (We first aligned the weights matrix from factor analysis to the weights of the baseline day's factor analyzer to obtain Λ'_{test},where Λ_{base} is the baseline weights and Λ_{test} is the original weights for the test day before Procrustes alignment.)

After obtaining the top $B = 140$ best-aligned channels to the baseline Day 1, we compared the performance of the linear decoder when it was trained on factor-analyzer transformed z fit to all channel data versus factor-analyzer transformed z fit to the truncated data of the best 140 channels based on the iterative Procrustes alignment metric. In both scenarios, we compared the case of training and testing on data from the same day, versus training on the baseline Day 1 and testing on the future test day (Figure 16).

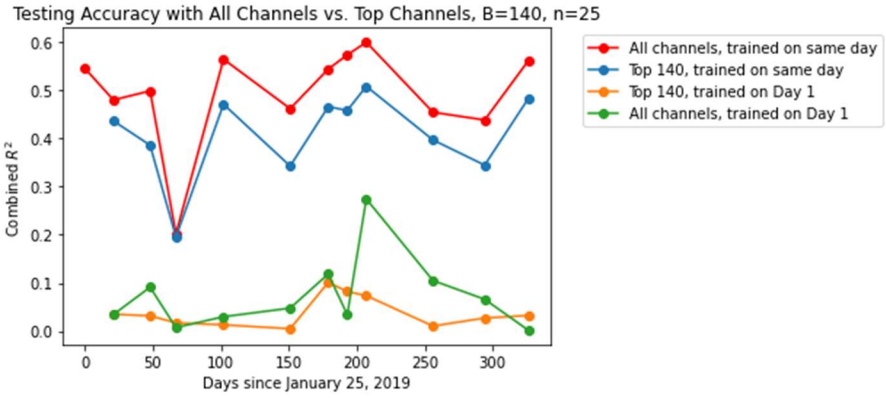


Figure 16. Accuracy of the linear decoder when the factor analyzer transformed data from all channels vs. data from the top 140 channels based on the iterative Procrustes alignment process. The red line shows the result of fitting and transforming with the factor analyzer on all 192 channels trained and tested on the same day, while the green line shows the same scenario but trained on Day 1 for all test days and tested on the test day. Similarly, the blue line shows the performance of the linear decoder when it was trained and tested on the same day using factor analyzed data of the top 140 channels, while the orange line shows the same case but when the linear decoder was trained on Day 1 and tested on each of the proceeding days.

Again, it is clear that this particular dataset does not lend itself well to the methods of the paper with factor analysis and the Procrustes algorithm for alignment to a single-day baseline. In the two cases when the linear decoder was trained on the baseline day and tested on later dates, the performance is drastically reduced, regardless of whether the Procrustes alignment had been applied, compared to training and testing the decoder on the same day. Even so, the Procrustes alignment not only does not help the decoder but actually hinders it in most cases. We now chose to move forward with a new method that incorporates multiple days into the baseline, since we found that single-day baselines perform poorly with our data.

3.4 Further Exploration: Average Baseline

During our previous analysis, we noted that the average baseline tended to have high performance compared to single-day baselines. Because we had observed this unique effect with the average baseline, we investigated whether the Procrustes alignment would have a different effect on decoding accuracy, or if it would have a similarly small or detrimental effect like single-day baselines in helping the decoder stay accurate over time.

Single-Day Baseline Results

As a check that the foundational methods indeed were not working when the decoder was trained on any single day and tested on all future days, we tried using every possible baseline day and examined the performance of unaligned factor analysis with the linear decoder versus factor analysis with alignment to the train day before inputting to the linear decoder. For each of the 12 days, we chose one day to be the baseline on which to train the linear decoder and align the FA weights via Procrustes for the aligned case. We evaluated the performance in combined R^2 for both the aligned and unaligned cases for each combination of train day paired with future test day (Table 3). The results from the first six baseline days are also plotted below (Figure 17).

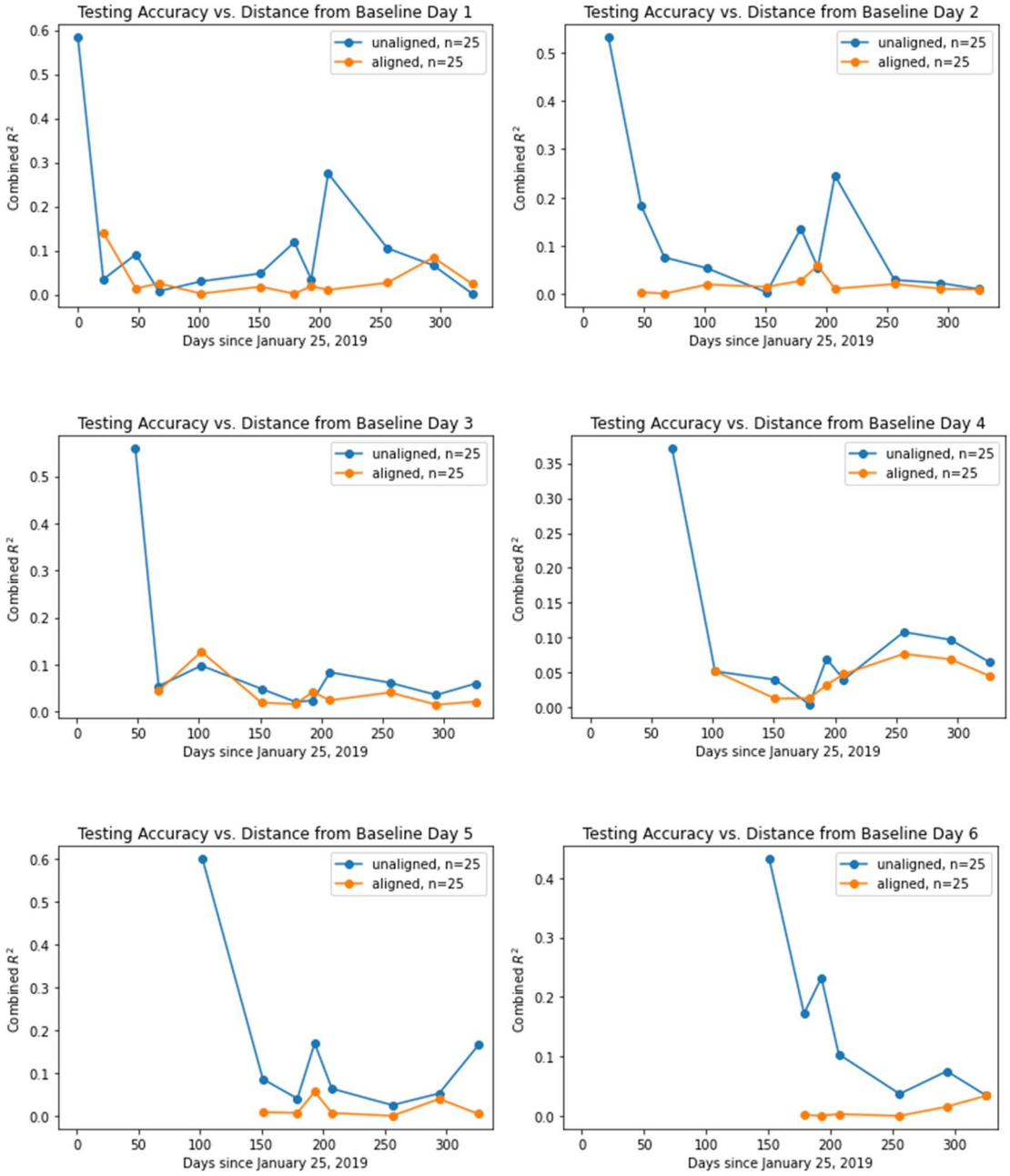


Figure 17. Single-day baseline effect of alignment on test accuracy. We find that if a single day is used to train the linear decoder and subsequent test days are aligned to the single-day baseline, there is no improvement in performance. In fact, weights not aligned via Procrustes generally lead to more successful decoding, though testing on a day that is different from the train day leads to very low accuracy in most cases.

It is clear from these results that when the linear decoder is trained on a single baseline day, the Procrustes alignment does not help it to perform better over time. We see that the aligned case

actually tends to have a lower accuracy for decoding than the unaligned case, so the foundational methods do fail in the scenario of using a single-day baseline with days that are spaced far apart such as those in our dataset.

Train Day (unaligned)/Test Day (aligned to train day)

Test Day/ Train Day	1	2	3	4	5	6	7	8	9	10	11	12
1	0.58	0.14	0.014	0.025	0.0020	0.018	0.0019	0.019	0.011	0.027	0.084	0.024
2	0.034	0.53	0.0048	0.0015	0.020	0.016	0.028	0.060	0.012	0.021	0.012	0.010
3	0.091	0.18	0.56	0.044	0.13	0.019	0.016	0.042	0.024	0.041	0.015	0.021
4	0.0075	0.076	0.054	0.37	0.052	0.013	0.014	0.033	0.048	0.077	0.069	0.045
5	0.030	0.054	0.098	0.052	0.60	0.0096	0.0081	0.057	0.0077	0.0012	0.041	0.0056
6	0.048	0.0042	0.048	0.040	0.087	0.43	0.0013	0.00047	0.0029	0.000025	0.015	0.034
7	0.12	0.14	0.021	0.0047	0.041	0.17	0.50	0.041	0.035	0.0071	0.0073	0.048
8	0.034	0.056	0.023	0.070	0.17	0.23	0.090	0.56	0.18	0.092	0.13	0.13
9	0.27	0.25	0.083	0.040	0.064	0.10	0.24	0.070	0.64	0.091	0.095	0.14
10	0.10	0.030	0.061	0.11	0.026	0.037	0.055	0.031	0.030	0.48	0.099	0.11
11	0.066	0.023	0.035	0.097	0.053	0.075	0.018	0.074	0.00067	0.32	0.49	0.011
12	0.0022	0.011	0.060	0.065	0.17	0.034	0.0080	0.12	0.048	0.0016	0.0099	0.55

Table 3. R^2 values for training on a single day and testing on each of the following days. In blue, we have the results without any alignment to the baseline, while in orange, we have the results with Procrustes alignment to the train day. The green diagonal shows the accuracy from training and testing on the same data from the same day (effectively, the training accuracy).

Table 3 illustrates that for the single-day baseline, it is not very reliable to test the decoder on a different day from the one on which it was trained. Most R^2 values remain below 0.1 for both the unaligned and aligned cases, though on average the aligned accuracies are worse than the unaligned ones, further supporting the finding that Procrustes alignment does not help the decoder remain accurate over time for our data. It is interesting to note that certain days do beat the trend, however; for example, when the decoder was trained on Day 1 and tested on the unaligned Day 9, we see a significantly higher R^2 value of 0.27. Similarly, when we trained on Day 10, the unaligned performance when testing on Day 11 was quite high at 0.32. Along with the finding that R^2 values for single channels peak sporadically across days (Figure 10), this suggests that there may be some repeatable information between certain days that helps the decoder to more accurately predict on those days that are more closely related to the training day. If this is the case, using an average baseline that factors in data from multiple days for training the decoder may help to capture information that is helpful in decoding multiple different days in a way similar to how training on Day 1 helped the decoder to better predict on Day 9.

Two-Day Average Baseline

We moved on to test how the decoder performed when instead of a single-day baseline, we took the average of the FA weights for two different days. For $m \in [1,5]$, we defined the average baseline as the average of two consecutive days' FA weights, or

$$\Lambda_{avg} = \frac{\sum_{i=m}^{m+1} \Lambda_i}{2}$$

We obtained each Λ_i by applying factor analysis to Day i's neural spike data u_i. We then created a factor analyzer with weights set to Λ_{avg} and transformed the data for both days to obtain z_m and z_{m+1}. With this, we created a concatenated matrix of latent variables $z^* = \begin{bmatrix} z_m \\ z_{m+1} \end{bmatrix}$ with the corresponding kinematics data $f^* = \begin{bmatrix} f_m \\ f_{m+1} \end{bmatrix}$. We then trained the linear decoder on z^*, f^*. We tested on all the future days for both aligned and unaligned cases. In the aligned case, we fit a factor analyzer to data from the test day to obtain Λ_{test} and modified the test weights to be

aligned to the average baseline weights with $\Lambda'_{test} = \Lambda_{test} \cdot \hat{O}^T$ via the Procrustes function. We then used these new weights to transform u_{test} to z'_{test} and predict kinematics. In the unaligned case, we simply predicted on z_{test} obtained from the FA weights Λ_{test} that had no alignment to the baseline. Figure 18 demonstrates the performance using the average weights from each pair of days from 1 to 6.

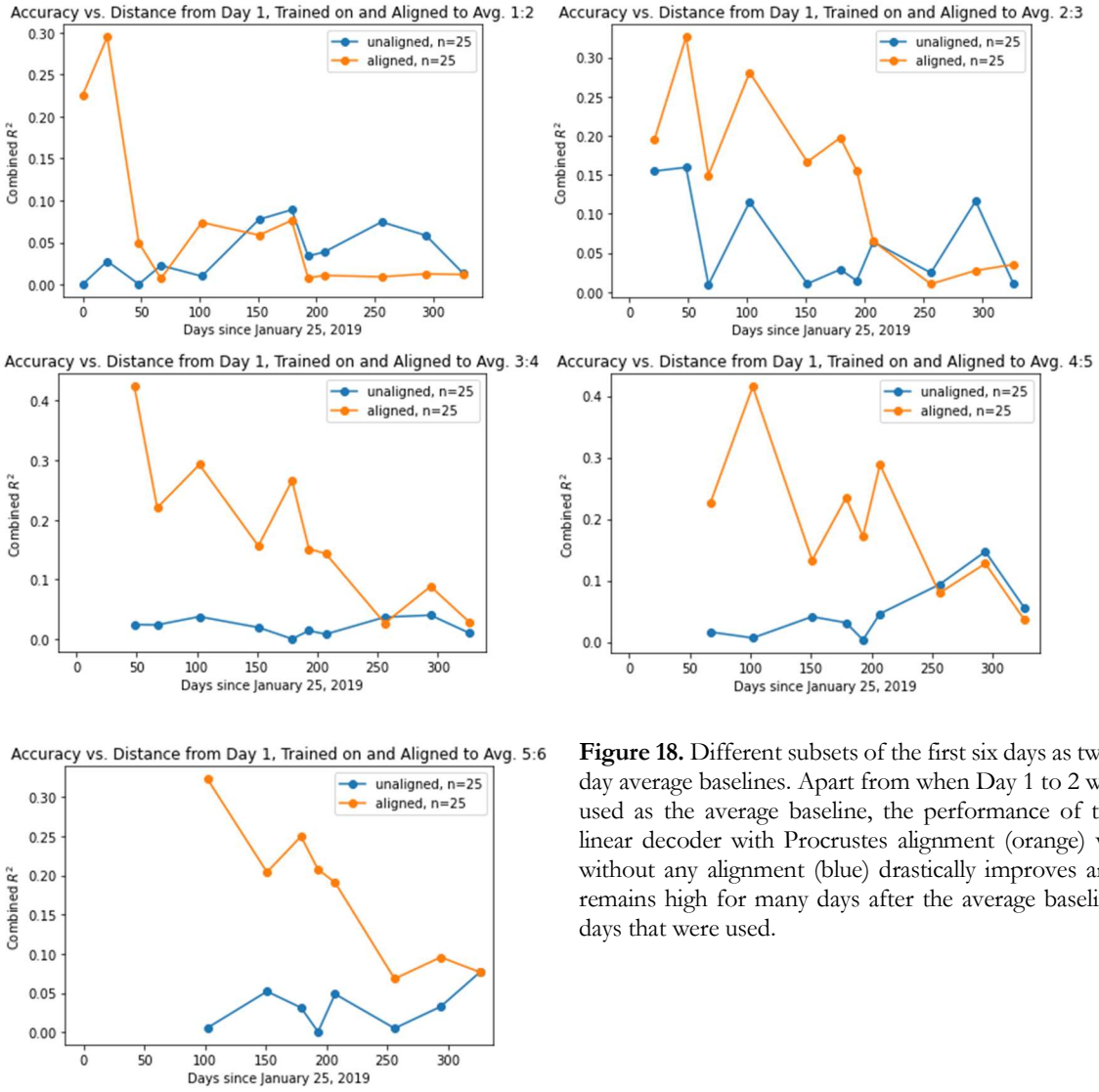


Figure 18. Different subsets of the first six days as two-day average baselines. Apart from when Day 1 to 2 was used as the average baseline, the performance of the linear decoder with Procrustes alignment (orange) vs. without any alignment (blue) drastically improves and remains high for many days after the average baseline days that were used.

Except for the case where the average baseline was taken from Day 1 to 2, we see a clear improvement in performance of the decoder over time when the test weights have been aligned

to the average baseline compared to when there has been no alignment of the factor analysis weights. The graphs show that alignment to a two-day average baseline helps the decoder perform better than the unaligned version for five to six days after the days included in the averaged weights (where the first two points on each graph are the days included in the average).

Multi-day Average Baseline

Finally, we investigated the performance of the decoder when trained on the average baseline of increasing size, for each of the cases starting from using just Day 1 as a baseline, the average of the first two days as a baseline, the average of the first three days as a baseline, all the way up to the average of the first six days as a baseline (Figure 19). To test this effect of including a different number of days in the baseline, we calculated the average baseline as

$$\Lambda_{avg} = \frac{\sum_{i=1}^{n} \Lambda_i}{n}$$

for $n \in [2,6]$ and created a factor analyzer with loadings set to Λ_{avg}. Using this factor analyzer, we transformed the neural data for each of the days from u_i to the lower dimensional z_i and concatenated the data as $z^* = \begin{bmatrix} z_1 \\ z_2 \\ \cdots \\ z_n \end{bmatrix}$, also concatenating the velocity data to obtain $f^* = \begin{bmatrix} f_1 \\ f_2 \\ \cdots \\ f_n \end{bmatrix}$. We then trained the linear decoder on z^*, f^* and tested it on each of the days. In the unaligned case, we input z_{test} to the decoder transformed from an independent, unaligned factor analyzer. For the aligned case, we used z'_{test} obtained from FA transformation with weights from the test day that had been aligned to the average baseline Λ_{avg} via Procrustes.

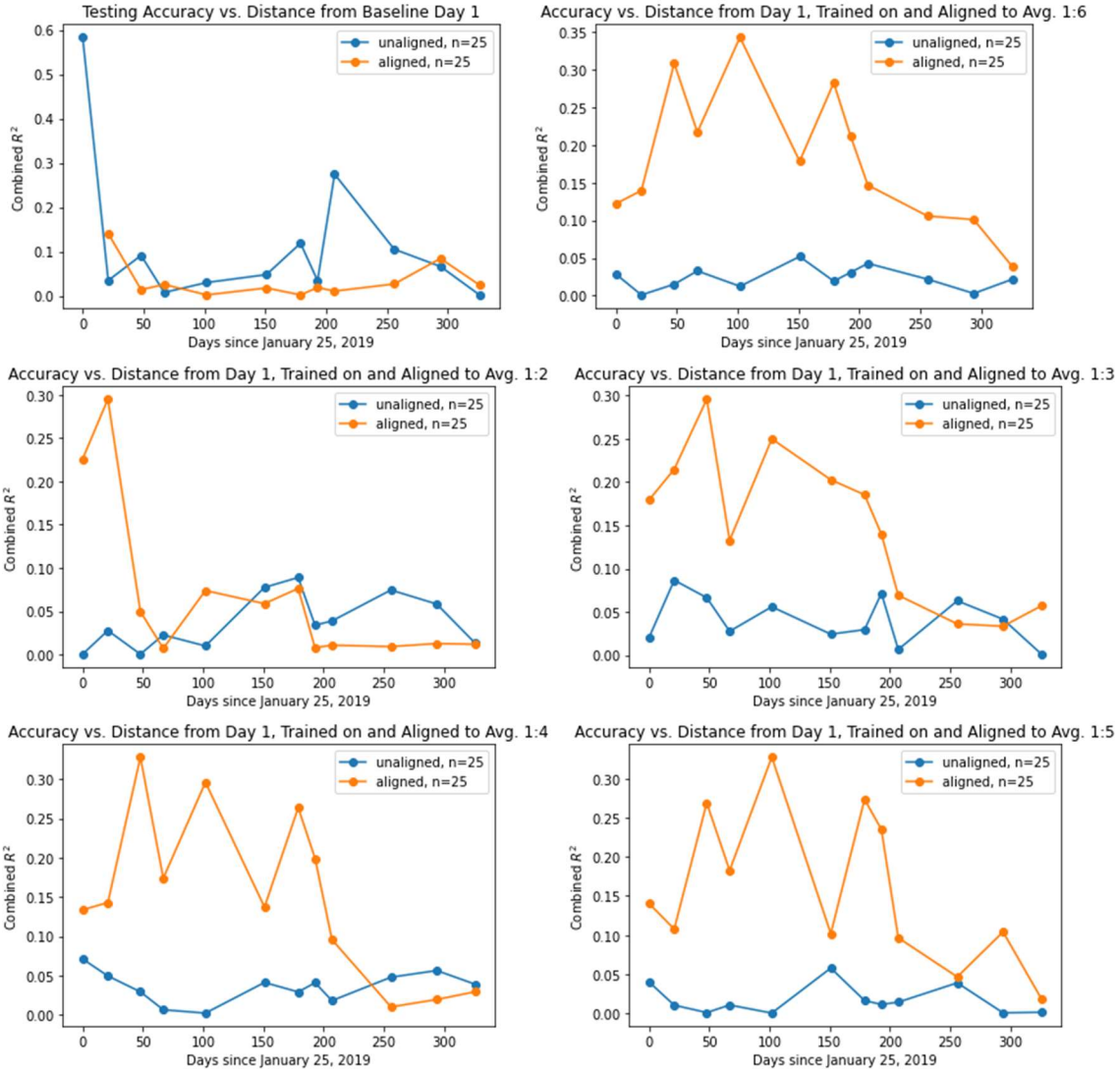


Figure 19. Left to right, top to bottom, we compare the performance of the average baseline from using the first day as baseline, the average of the first two days, first three days, first four days, first five days, and first six days. The Procrustes alignment visibly improves the performance and prolongs the decoder's retention of higher accuracy predictions.

As the number of days included in the average increases, so does the accuracy for the aligned test data. The point at which the accuracy with alignment drops below the accuracy without alignment consistently becomes a later and later date as more dates are added to the average baseline, until at 5 to 6 days average, the aligned data outperforms the unaligned data for every single test day. This is an exciting result, since it shows that in the case of the average baseline, the linear decoder actually does learn enough from being trained on the average baseline-transformed data that it can better predict the kinematics for days to come when those test days

are also aligned to the average baseline. This strongly supports the use of the Procrustes alignment over factor analysis alone, as it greatly boosts the performance of the decoder when the FA for the test day is aligned to the data on which the decoder was trained.

There are multiple possibilities for why the average baseline shows such a striking difference in performance compared to the single-day baseline. For one, using an average baseline means the training dataset is larger since it incorporates concatenated data from multiple days. This would help the decoder better predict kinematics especially in the case where it had been underfitting on the single-day data. There could also be an element of the apparent repeating patterns in the data that make certain dates more similar to others. If multiple dates are included in the training data via the average baseline method, it is likely that the linear decoder will learn how to predict on a greater number of days because of the test days that are somehow related to one of the baseline days. Another possibility is that taking the average of the weights across multiple days better represents the major neuronal population activity such that if for one day the activity is in a different direction than usual, the average baseline will still capture the dominant activity patterns for most days to allow the decoder to maintain a better accuracy for many days into the future.

Chapter 4

CONCLUSION

In this study, we investigated the methods of Degenhart et al. (2020) to stabilize neural decoding performance over time. The foundational work proposed a stabilization approach on data with simulated noise using factor analysis and Procrustes alignment to a baseline day to help remove noisy channels before decoding. However, we found that with real human data that has high instabilities and a sparse distribution of experiment dates, the methods of the foundational work are ineffective at improving the decoding accuracy. In fact, when we train and test the decoder using data from different dates, the decoding performance drastically drops regardless of alignment, indicating that the foundational methods have failed on our dataset. Additionally, we found that single channel performance varies greatly between days, with no one electrode channel remaining consistently significant across days. Thus, the foundational methods of channel removal would not be effective for multi-day testing when the days are far apart.

As such, we propose a new method that builds on the foundational work. Instead of using a single day baseline, we define an average baseline by taking the average of factor analysis weights across multiple baseline days and aligning test data to this average baseline. When the linear decoder is trained on the average concatenated data, we find that decoding with Procrustes alignment of the test data greatly outperforms predictions from unaligned test data. In addition, incorporating multiple days into the baseline prolongs the decoder's higher performance for many days after the training dates. In our case of sparse human data, we find that the methods of the foundational work are effective at improving decoder performance over time provided that the average baseline is used.

It is likely that the single-day baseline experiments presented here failed largely because of the nature of our dataset. Because of the limitations in the current pandemic situation that make it difficult to take new data with our specific research question in mind, we find ourselves with data that is spaced too far apart to model typical use of a BMI. In addition, human data may be

more unpredictable than the simulated data used in the foundational work. However, even though our data was not collected with the goal of our stability analysis in mind, it has led us to an interesting conclusion about the use of an average baseline.

In all, factor analysis and Procrustes alignment provide a promising method to allow a BMI decoder to maintain accuracy in predicting future cursor kinematics that are far away in time based solely on neural spike data. By transforming the data and using weights aligned to an average baseline, we find much higher decoding performance than in use of a single-day baseline or transformation without alignment. Since this method shows such a notable improvement in accuracy over time with a simple linear decoder, there is great potential that the method could help even better stabilize the decoder performance and increase its accuracy when more sophisticated deep learning models are used.

BIBLIOGRAPHY

Aflalo, T., Kellis, S., Klaes, C., Lee, B., Shi, Y., Pejsa, K., Shanfield, K., Hayes-Jackson, S., Aisen, M., Heck, C., Liu, C., & Andersen, R. A. (2015). Decoding motor imagery from the posterior parietal cortex of a tetraplegic human. *Science (New York, N.Y.)*, *348*(6237), 906–910. https://doi.org/10.1126/science.aaa5417

Coallier, É., Michelet, T., & Kalaska, J. F. (2015). Dorsal premotor cortex: Neural correlates of reach target decisions based on a color-location matching rule and conflicting sensory evidence. *Journal of Neurophysiology*, *113*(10), 3543–3573. https://doi.org/10.1152/jn.00166.2014

Degenhart, A. D., Bishop, W. E., Oby, E. R., Tyler-Kabara, E. C., Chase, S. M., Batista, A. P., & Yu, B. M. (2020). Stabilization of a brain–computer interface via the alignment of low-dimensional spaces of neural activity. *Nature Biomedical Engineering*, *4*(7), 672–685. https://doi.org/10.1038/s41551-020-0542-9

Haghi, B., Kellis, S., Shah, S., Ashok, M., Bashford, L., Kramer, D., Lee, B., Liu, C., Andersen, R. A., & Emami, A. (2019). *Deep Multi-State Dynamic Recurrent Neural Networks Operating on Wavelet Based Neural Features for Robust Brain Machine Interfaces* [Preprint]. Neuroscience. https://doi.org/10.1101/710327

Majumdar, C. (2018, May 29). *Dimensionality Reduction Using Factor Analysis*. Medium. https://medium.com/@chiranjit7/dimensionality-reduction-using-factor-analysis-8aa754465afc

National Spinal Cord Injury Statistical Center. (2021). *Facts and Figures at a Glance*. University of Alabama at Birmingham.

Norman, S. L., Maresca, D., Christopoulos, V. N., Griggs, W. S., Demene, C., Tanter, M., Shapiro, M. G., & Andersen, R. A. (2020). *Single Trial Decoding of Movement Intentions Using Functional Ultrasound Neuroimaging* [Preprint]. Neuroscience. https://doi.org/10.1101/2020.05.12.086132

Shah, S., Haghi, B., Kellis, S., Bashford, L., Kramer, D., Lee, B., Liu, C., Andersen, R., & Emami, A. (2019). Decoding Kinematics from Human Parietal Cortex using Neural Networks. *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, 1138–1141. https://doi.org/10.1109/NER.2019.8717137

Sussillo, D., Stavisky, S. D., Kao, J. C., Ryu, S. I., & Shenoy, K. V. (2016). Making brain–machine interfaces robust to future neural variability. *Nature Communications*, *7*(1), 13749. https://doi.org/10.1038/ncomms13749