SPIKES-FORCE DECODERS

- **Overview of the Task:**

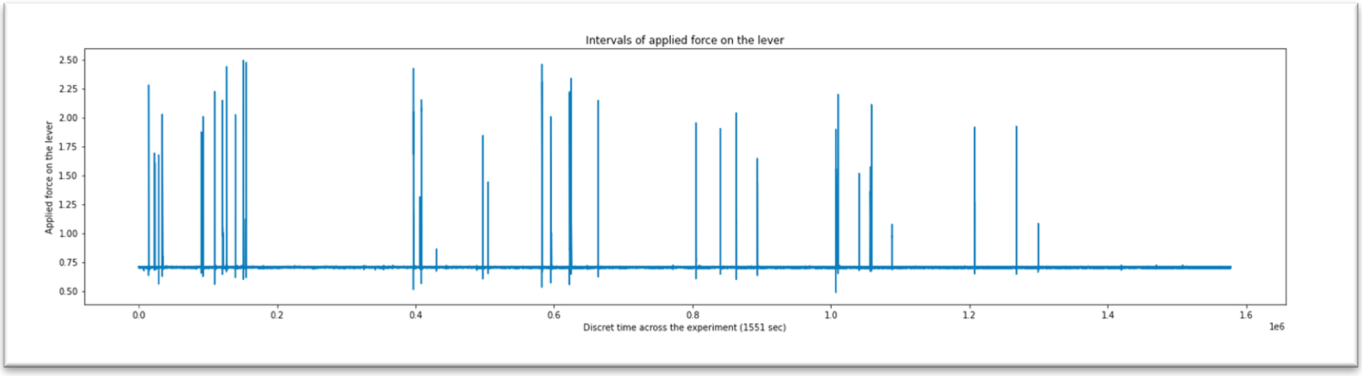
The goal of implementing those neural decoders was to assess the possibility of decoding the applied force put on by a rat on a specific knob from the neural activity recorded from his primary motor cortex (M1). The neural activity was recorded from a Microwire Array with 32 channels from TDT (Subject #194). We tested several decoders architectures in order to test the impact of different design/parameters.

- **Data Pre-Processing**

We first build a pipeline that process the raw data from the recording session (force and neural signals) in order to analyze it and generate datasets for training various decoders. The applied force on the knob was continuously sampled at 1017.253 Hz and stored in the raw data files for the whole 1551.01 seconds of the session under the variable "data.streams.knob".

- 1551.01 sec x 1017.253 Hz ≈ 1 577 600 samples of applied force
- The baseline (where no force applied) ≈ 0.75
- Statistics of the force signal to the right -->

Statistics	Value
Std	0.07854
Mean	0.70997
Median	0.7031
Size	1 577 600
Min	0.4846
Max	2.4918

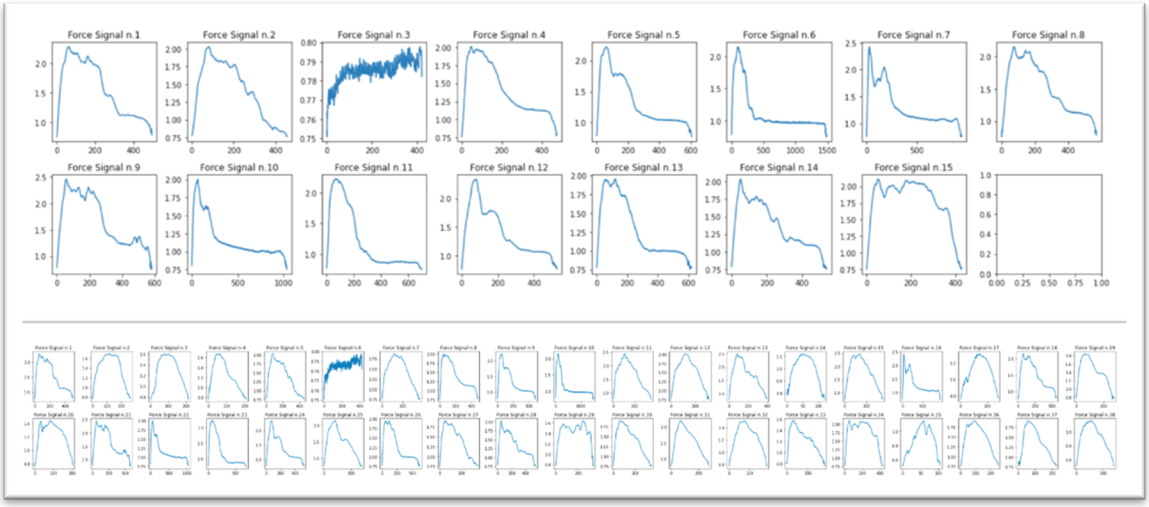


- Figure 1: Intervals of applied force on the knob. The baseline at 0.75 is clearly visible.

In the figure 1 above, we can see the knob signal sampled across all the recording session. We can clearly see the baseline on the force sensor around 0.75 and the visible spikes that represents short intervals where a force is applied on the lever.

Those intervals can be seen below on the figure 2, where we extracted for the upper part of the figure all intervals above the 0.75 baseline and with a length of 400 samples minimum, which translates to 400 / 1017.2526 Hz ≈ 0.39 sec. We obtained 15 intervals, although we classified the signal 3 [34579, 34999] as noisy due to its high noise content and its close proximity with the baseline.

The lower part of the figure 2 shows when we adjust the parameter of the function responsible for extracting the intervals above the 0.75 baseline and with a minimum length of now 100 samples.



- Figure 2: (Above) All intervals above the 0.75 baseline and with a length of 400 samples minimum.
(Below) All intervals above the 0.75 baseline with a minimum length of 100 samples.

- **<u>Models & Datasets</u>**

<u>- Decoder V1</u>

- Recurrent Model (LSTM) with a Many to Many Architecture (Regression task)
- Input X: 15 x (26 x 8) Matrices of neural activity
    - 26 channels kept
    - 8 firing rates vectors processed sequentially
    - Temporal Resolution R = 0.05 Second (50 ms)
- Output Y: 15 x (8d) Vectors of predicted forces

- Extracted all applied force intervals above 410 Samples (0.4 Seconds) and kept only their first 410 Samples
- Got 15 Intervals of Applied Force
- Separated each interval into 8 Predictions bins. Therefore, each bin was of length 50ms.
- The Mean of each Bin was computed as the target force for this sub-interval
- The corresponding neural activity of those 8 bins was extracted as 8 firing rate vectors of temporal resolution R = 50ms)

- 8 Predictions per Interval of 410 samples or 0.4 seconds of applied force.
- Force data and neural firing matrices were normalized [0-1]

- Decoder V2

- Recurrent Model (LSTM) with a Many to One Architecture (Regression task)
- Input X: 759 x (26 x 5) Matrices of neural activity
    - 26 channels kept
    - 8 firing rates vectors processed sequentially
    - Temporal Resolution R = 0.02 Second (20 Ms)
- Output Y: 759 x (1d) scalar of predicted force

- All Intervals higher than the baseline = 0.75 and higher than 100th Samples were extracted
- 37 intervals (38 in total - 1 noisy interval = 37)
- Function*: "List_Processed_Data_2 = Extract_Index(Data, Baseline=0.75, Length=100) With One Removed"*
- For all those intervals, the applied force was extracted at each 20th data point, starting at the 20th sample.
- In total, 759 applied force data was extracted

- Neural Firing Matrix Extracted:
- The Corresponding 5 firing vectors of the activity 0.1 sec before the applied force (r = 20 ms)

- Channels were investigated for removing those with zero content info (All 0s in the firing rates)
- But they all had a mean over 0 for the neural firing matrices extracted
- So, we kept them all, although some were really low on information content
- (Those with a mean below a certain threshold could have been removed)

- Decoder V3

- Modification of the V2 Decoder/Dataset
- Transformed into a classifier instead of continuous regression
- We opted for discrete classes as small differences in force values are not something very perceptible
- Hence, we thought it as an interesting approach to test

- Recurrent Model (LSTM) with a Many to One Architecture (Classification task)
- Input X: 759 x (26 x 5) Matrices of neural activity
    - 26 channels kept
    - 8 firing rates vectors processed sequentially
    - Temporal Resolution R = 0.02 Second (20 ms)
- Output Y: 759 x (Number of classes = 4d) vector predicted (probability distribution over the classes)

- We first transformed the continuous interval of force [0.75 - 2.5] into 12 interval Classes
- Only modified from V2 the label format from force scalars to a vector representing a distribution over the classes
- But the training was slow, the data was sparse, and the info content was low for this kind of number
- We then switched to a 4-class classifier... transformed the continuous interval of force [0.75 - 2.5] into 4 interval Classes

	Type	Nb of sample	Input Firing Rate Vectors	Temporal R	Nb Predicted Output
Decoder V1 (LSTM)	Regressor -- Many to Many	15 x 8	8	50 ms	8
Decoder V2 (LSTM)	Regressor -- Many to One	759	5	20 ms	1
Decoder V3 (LSTM)	Classifier -- Many to One	759	5	20 ms	4

- **Results and Analysis**

	Units (Layer)	Normalized [0-1]	Activation	Dropout	Epoch	Batch Size	MSE Test	MAE Test	Accuracy Test
Decoder V1	200 (1)	Input/Output	Softplus	0.2	100	32	**0.0404**	0.1779	____
Decoder V2	300 (1) - 100 (2)	Input/Output	ReLu	0.2	100	32	0.0474	**0.1633**	____
Decoder V3	500 (1)	Input	Softplus	0.4	200	32	_____	_____	**53.29%**

The metrics used for the V1 and V2 decoder (Regression-based) were the MSE and the MAE on the test data. For testing purpose, the MSE was used as the loss function of the decoder V1 and the MAE was used as the loss function of the decoder V2. It is important to understand that as the output forces were normalized between [0 - 1], the MAE & MSE errors are seen in a compressed space that is a bit different from the original space. As the V3 decoder was a classifier, the metric used was the accuracy on the test data. And finally, the test/train split was always kept around [0.15 - 0.20]. Now overall, the results seem acceptable for the first iterations of decoder designs. For the V1 and V2, the MSE on the test data was 0.0404 and 0.0474 respectively while the MAE on the test data was 0.1779 and 0.1633 respectively. For the V3, a test accuracy of 53.29% was obtained on the test data for a 4-class prediction.

But nonetheless, we consider those results as preliminary as we did not have time to test iterations on the decoders/parameters/datasets that we though could give better results. Indeed, many design parameters were uncertain according to us. First, the choice of a temporal resolution so short (50 ms, 20 ms, 20 ms) gave us sparse matrices to work with, which maybe was not a good choice as it limited the amount of information we had. For example, the decoder V2 and V3 had only information from 0.01 seconds before the applied force as 5 vectors of neural firing rates, which were produced with a temporal resolution of 20 ms each (0.02 sec), for a total of 100ms. This is very short (I think).

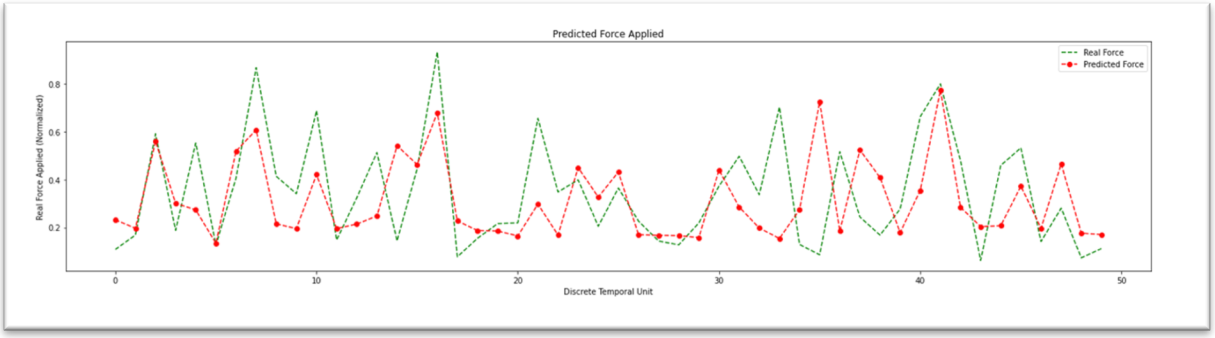
Those short values were selected as my reasoning led me to think that it may be possible that the information relevant for decoding the precise force at a specific moment may be contains only in the 0.1 seconds of neural activity before the applied force instant. Second, maybe selecting an averaged force over a specific interval seems a better idea, like we did for the decoder V1 but not V2 and V3. Also, it is possible to say that the decoders are most often overfitted due to the small amount of training sample available for those data-hungry models like the traditional LSTMs. Hence, the R2 metric was not relevant in this context and may be negative due to the model being designed to overfit the training data, although this may also be an approach that performs on the test data too.

- **Things that may be tested in future iterations:**

- Filtering/smoothing of the force signal.

The filtering/smoothing of the force signal across time could allow a much stronger learning curve for the models like we can see in figure 3 below, where the predictions from the Decoder V2 (red) are plotted against the real force labels (green). As we can see, the decoder captured a lot of the information from the supervised mapping but a filtering/smoothing of the force data volatility could boost the decoder's accuracy.

- Augmenting/Tuning the temporal resolution parameter R and the number of time bins used.
- Test the usage of non-recurrent decoder (MLP, XGBoost, ...).
- Trying a stronger degree of channel information filtering (ex: remove noisy channels below a certain threshold of information).
- Hyperparameters search (grid-search).
- Test with more channels and more data samples.



- Figure 3: Predictions from the Decoder V2 in red against the real force (label) in green. As we can see, the decoder captured a lot of the information from the supervised mapping. If iterations to the decoders were made in order to filter and smooth out the force data volatility, it is really plausible that the decoders could get a much stronger accuracy.