ARTICLE

Analysis and Correlation of Brain Motor Information in the Context of BMIs

Guilherme Costa Ferreira,[*,1] Il Memming Park,[2] and Athanasios Vourvopoulos [3,1]

[1]Instituto Superior Técnico, Lisbon, Portugal
[2]Champalimaud Research, Champalimaud Foundation, Portugal
[3]Institute for Systems and Robotics/LaSEEB

*Corresponding author: guilherme.costa.ferreira@tecnico.ulisboa.pt

Abstract

Using Brain-Machine Interfaces (BMI) to solve health problems related to the nervous system has proven a promising approach. With innovative studies that allow motor rehabilitation to companies developing and commercialising devices that grant seamless computer integration, the future solution to these problems lies with BMIs. We developed and applied two models: (a) Population vector and (2) Naïve Bayes to decode from 182 motor neurons' hand movement from the MC Maze dataset. After optimisation, we obtained 0.24 and 0.45 in R^2 correlation, respectively. They were able to shed light on many characteristics of the motor cortex, such as the tuning of the neurons and the increased amount of information they code. This information granted the possibility of improvement and adaptations of our models. However, we did not obtain correlations closer to other models applied to this dataset, as our decoders lacked developed preprocessing steps. Nonetheless, this project works as an introduction to the BMI problem.

Keywords: Brain-Machine Interfaces; Neural Decoding; Motor Cortex; Population Vector; Naïve Bayes

Abbreviations: BMI: Brain-Machine Interfaces, BCI: Brain-Computer Interfaces, ML: Machine Learning, Population Vector: PPV, Naïve Bayes: NB

1. Introduction

As medicine evolves, multiple diseases and conditions of the past become harmless by simple procedures and guidelines implementation. From discovering penicillin to eradicating smallpox, Humankind has found ways to deal with some of its fearsome enemies. Today's battlefield is less about fighting infectious diseases and more about fighting chronic, autoimmune and cancerous diseases. As this battle continues, we find the use of the term "untreatable condition" having lesser and lesser meaning.

However, there is a domain that still lags in understanding and having sound and affordable solutions: the nervous system. It is important to remember that the brain represents medicine's finest challenge [1]. To understand this complexity, we can compare the human brain to our everyday computer, as both are responsible for varied fundamental tasks such as processing, calculation, and even sensing. However, we are still very far away from understanding its complexity and being able to translate it. We tried mimicking its functioning with generative AI models. Regardless, these systems employ a city's daily electricity usage to run programs that are still far away from our highly effective and efficient organ [2] [3].

One type of disease that falls in both categories is the partial or complete paralysis of the human body since it is a condition from the nervous system commonly labelled as "Untreatable". This type of disability can be from stroke, brain trauma or spinal cord injury [4], and will leave the patient with

a profoundly limited lifestyle. Statistics point out that 1.7% of the US population (about 6 million) suffer from this condition, as the prevalence of this problem is rising due to the increasing number of strokes and accidents [5].

The growth of new devices that focus on translating brain information into meaningful data to feed new technologies might be the only hope for these people as neuron regeneration is still very difficult [6]. These devices are called Brain-Machine Interfaces (BMI) or Brain-Computer Interfaces (BCI) and have promising health applications. The origin of these devices dates back to the discovery of EEG and electrophysiological recordings. As EEG limitations started to be understood, new invasive methods were born, allowing for higher spatial resolutions. These improved resolutions allowed for single neuron data acquisition [7] and consequently more insights into brain functioning. The more recent advances allowed studies such as the one from Wagner et al., 2018 where paraplegic patients walked again. Even companies are now developing BMI devices to be industrialised and commercialised, such as Elon Musk's Neuralink's BMI [9]. Even though the focus is now on motor rehabilitation, other fields are starting to show promising results with vision and hearing rehabilitation.

As the use of AI becomes more prevalent in the development of these new devices, it is fundamental that we can still understand what these models are doing, allowing for these techniques to be more than solutions but also lenses in how this organ works.

The preferred region to extract this type of information from the brain is the motor cortex. This region, responsible for managing and passing on movement, can be split into the primary motor cortex (M1) and the supplementary motor area (M2) [10]. The M2 region is responsible for movement processing connecting directly to the M1 region. The latter can then send the signal by the spinal cord directly to effector muscles [10]. Due to this direct connection to motor movement, we can verify that some neurons in this region tune to particular movements [11]. This correlation is verified by analysing Tuning Curves - graphs that plot the neurons' activity versus the desired behaviour. If the neuron fires more during a particular movement, we can say that that neuron is tuned to that direction [11].

2. Method

This project used pre-acquired data from a public dataset. This section will briefly explain the shape of the info and how this dictates the approach. Here, we will also define both implemented models.

2.1 Dataset Structure

The dataset used is from the Neural Latents Benchmark (https://neurallatents.github.io/). Here, we used the trials from the MC_Maze. In this dataset, we can find neural firing from the M1 and M2 region from the Motor Cortex of a *Macaca mulatta* from 182 neurons over 2295 trials with a temporal resolution of 1 ms acquired with intracortical microelectrode arrays. In each trial, the monkey solves a reaching task where the target point could be obstructed by walls, forcing it to change the trajectory accordingly. The MC_Maze dataset only registres the successful trials (an example is provided in figure 1a) [12]. All coding used Python, resourcing to many libraries, primarily the NumPy package, for optimisation. The code and more documentation are available in the GitHub repository (https://github.com/GuilhermeCosta-Ferreira/PIC-Motor-Cortex-and-BMIs)

Some preprocessing was applied to the dataset. The dataset comes with timestamps from when the registry begins (`start_time`), from the presentation of the target (`target_on_time`), from the presentation of the go cue (`go_cue_time`), from when the monkey begins movement (`move onset time`) and from when the registry is over (`end_time`). However, from the moment the monkey reaches

the target to the moment the registry is over, there can be several seconds where the monkey is not making a well-planned movement. Due to that, we added the moment the monkey reached the target (`reach_time`). This timestamp is where the cursor position was the smallest to the final target (figure 1b is the corresponding raster plot of a trial after this preprocessing step).

To be able to train, update and test the models, the dataset was divided into a train, val and test set, respectively. Using the dataset's default validation set as our test set and splitting the default train set 90/10 into train and validation, respectively, we obtained slices of 67.49%, 7.49% and 25.01% (1549, 172, 574 trials).

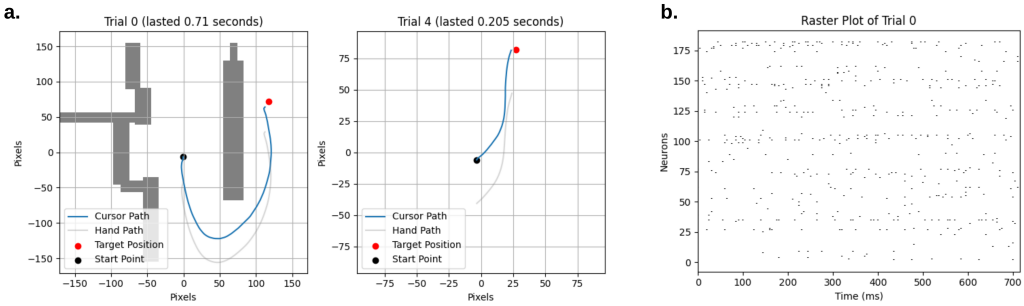

Figure 1. (a) Trajectory plots for two types of trials. In the first trial (trial 0) we have a maze situation, where walls are in front of the target. Trial 4 is an example of a trial where there were no walls. The dataset has both data from the cursor position (blue) and the wrist of the monkey (grey). **(b)** Visualisation of trial 0 raster plot from movement onset through the moment the monkey reached the target

Due to Machine Learning (ML) algorithms not liking 0s and 1s (too sparse data), we added, when needed, the option to turn spike data into firing rate by passing a moving average. We discuss the window size later on. The data was re-binarised from 1 ms to 5 ms, as recommended in the Benchmark's Demo Notebook, to handle the missing values.

2.2 Population Vector

The Population Vector Decoder (PPV) is the first model we tried for this problem. The decoder presented by Georgopoulos (1988) is the base for our model. This approach introduces the concept of tuning curves mentioned early on.

This representation of neural activation for motor tasks holds itself over two assumptions: (1) Some neurons tune to a particular single direction; (2) These neurons are independent and can be summed up into a population to predict/describe the executed movement. There are, however, some differences from the Georgopoulos' (1986) model. In that study, the monkey did a 3D reaching task with 282 neurons providing the data [13], while here we are working with a 2D plane of movement (screen) and a hundred neurons short.

The first step was to get each neuron's preferred movement's angle through tuning curves. This step is achieved by first getting the vector M for each time bin (equation 1).

With the vector M now defined, we can find the relation of the firing rate $d(\overrightarrow{M})$ with the cosiness of the movement's direction (equation 2). To build the decoder, we need to estimate the b coefficients. First, we binned the angle domain and then got the mean firing rate for each angle. Second, we ran a linear regression to obtain the estimated values. With the coefficients estimated, we got each neuron's preferred direction C (equation 3). We can now update the equation for $d(\overrightarrow{M})$, where θ_{CM}

$$M = \begin{bmatrix} m_x \\ m_y \end{bmatrix} = \begin{bmatrix} \cos(\chi) \\ \cos(\psi) \end{bmatrix} \quad (1)$$

$$d(\overrightarrow{M}) = b + b_x m_x + b_y m_y \quad (2)$$

$$C = \begin{bmatrix} c_x \\ c_y \end{bmatrix} = \frac{1}{k} \begin{bmatrix} b_x \\ b_y \end{bmatrix}, \text{ where } k = \sqrt{b_x^2 + b_y^2} \quad (3)$$

$$d(\overrightarrow{M}) = b + k\cos(\theta_{CM}) \quad (4)$$

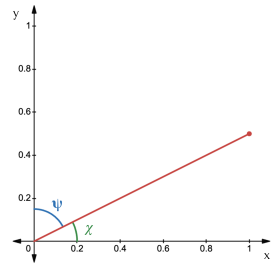

Figure 2. Single neuron's mathematical formulation for tuning curve and representation of the direction of movement of one bin with angles χ and ψ

is the angle between the direction M and the preferred direction C (equation 4). Figure 3a has the data and fitted tuning curve of neurons 2251 and 2961 with this formulation. [1]

Now that we have each neuron's parameters, that is, now that we fitted a cosine function to our data, we can get the overall population vector $P(\overrightarrow{M})$.

This way, it becomes evident that Population Vector $P(\overrightarrow{M})$ is just the sum of all neurons contribution $N_i(\overrightarrow{M})$ (equation 5 and figure 3b). Each neuron's contribution is calculated by equation 6.

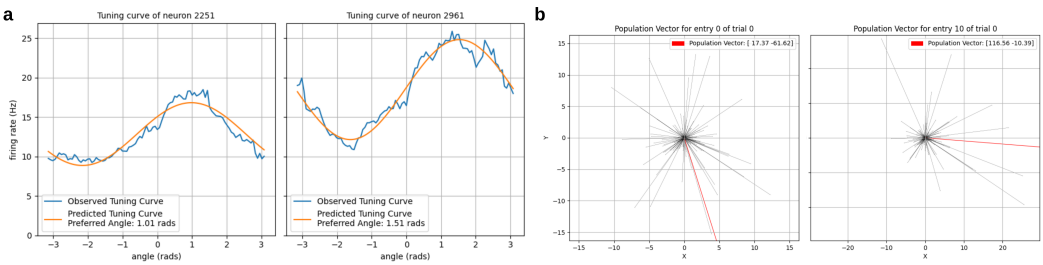

Figure 3. (a) Tuning curve visualisation for neurons 2251 and 2961 **(b)** Representation of the Population Vector calculation as a sum of different sized preferred directions

Once we apply this over every time bin of a trial, we can get the predicted angles of each movement (equation 7) as shown in figure 3b. Since the $P(\overrightarrow{M})$ is not normalised, we also obtain the vector's magnitude. However, a coefficient would be needed to transform the sized vector into a velocity vector. This way, we can rebuild the trajectory.

$$P(\overrightarrow{M}) = \sum_{i=1}^{182} N_i(\overrightarrow{M}) \quad (5) \qquad N_i(\overrightarrow{M}) = (d_i(\overrightarrow{M}) - b_i) \cdot C_i \quad (6) \qquad P(\overrightarrow{M}) = \sum_{i=1}^{182} (d_i(\overrightarrow{M}) - b_i) C_i \quad (7)$$

This model still has an issue: not all neurons tune to a particular direction. To deal with those untuned neurons, we defined a threshold to remove neurons that have fitted tuning curves with Euclidian distances to the original tuning curves higher than the set value (poorly fitted neurons).

There are some parameters, called hyperparameters, that we can optimise. For this model, this includes (1) Time Resolution (the size of the window used to calculate the firing rate with the moving average), (2) Tuning Resolution (The number of bins used to divide the angle domain) and (3) Neurons Removed Threshold. These results are in figure 5.

[1] Even though there are 182 neurons, they have different names. To check the numeration, you can check the appendix table 1

2.3 Naïve Bayes

The other model applied to the MC_Maze dataset was a Naïve Bayes Model (NB), based on Zhang et al. (1998) as a successor to the PPV. For this approach, we used tuning curves again, but instead of using angle as the variable, we used the velocity. Since this is a Classifier, it will only output some value from the binned input from the tuning curve. If we were working with angles, we would just be predicting the angle of movement, rendering it impossible to predict a trajectory. That way, we would have needed to predict independently the magnitude of movement. Because velocity is a vector, we had to decompose it into velocity in the x and y directions. This way, instead of having tuning curves, we had tuning surfaces. Due to more complex structures, instead of linear regression to predict the surface parameters, the surface is fitted using `curve_fit()` from the SciPy package, which uses non-linear least squares [15].

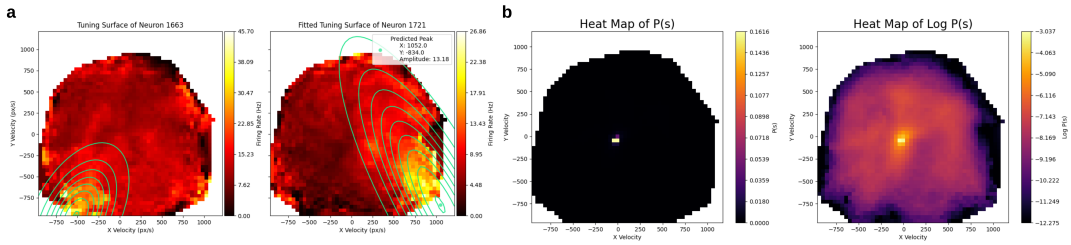

Figure 4. **(a)** Tuning surface visualisation for neurons 1721 and 1663 fitted with a Gaussian surface in a 50 binned Ω **(b)** Representation of the $P(\overrightarrow{s})$ for a model with 50 bins of resolution. The first graph is not logged, while the second one is

This time, to fit the data, we used a Gaussian surface. We chose this shape because it is similar to the bump we saw in the tuning surfaces and can be stretched and rotated for less peak-like tuning shapes. For an example of how we generated the bell curve and the data, there is figure 4a.

To tackle this problem, we need some assumptions: (1) All neurons are independent of each other, and (2) we can describe the neurons firing pattern with a Poisson distribution. The first assumption will allow us to deal with the high dimensionality of the dataset (multiple neurons) being the origin of the term Naïve in the model's name. The Fano Factor intuition from Eden & Kramer, 2010 sustains the second assumption. This factor is the variance of a spike count over the mean of spikes in a given time window ratio, and it is always very close to 1. The variance and the mean of a Poi(X) is λ, making the corresponding Fano factor close to the one of a neuron.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{8}$$

$$P(\overrightarrow{s}|\overrightarrow{r}) = \frac{\prod_{i=1}^{182} P(r_i|\overrightarrow{s})P(\overrightarrow{s})}{P(\overrightarrow{r})} \tag{9}$$

$$\overrightarrow{s}_{\text{pred}} = \arg\max_{s\in\Omega} \prod_{i=1}^{182} P(r_i|\overrightarrow{s})P(\overrightarrow{s}) \tag{10}$$

$$P(r_i|\overrightarrow{s}) = \frac{e^{-f_i(\overrightarrow{s})} f_i(\overrightarrow{s})^{r_i}}{r_i!} \tag{11}$$

$$P(\overrightarrow{s}_i) = \frac{\#\overrightarrow{s}_i}{\#\text{entries}} \tag{12}$$

$$P(r_i|\overrightarrow{s}) = \frac{e^{-(f_i(\overrightarrow{s})\tau)} (f_i(\overrightarrow{s})\tau)^{r_i}}{r_i!} \tag{13}$$

This model is based on the Bayes theorem (equation 8). In this case, we have a spike count $\overrightarrow{r}$, and we want to get the probability of the movement going in direction $\overrightarrow{s}$, obtaining equation 9. To run this as a classifier, we binned the velocities, where we treated the resolution (number of bins per direction) as an optimisable hyperparameter (the universe of binned velocities is called Ω).

Maximising the probability, we get the velocity of the movement. Because the prior $P(\overrightarrow{r})$ is the same when comparing different $\overrightarrow{s}$ in Ω, we can ignore this parameter, getting equation 10.

It is now needed to calculate each prior and posterior. Because we defined Poi(X) as the distribution, we obtain the posterior $P(r_i|\overrightarrow{s})$ by resorting to equation 11, where $f_i(\overrightarrow{s})$ is the tuning surface estimated. Due to Poisson being a discrete distribution, we need to convert the firing rate used to make the tuning curves back into spike count. We achieve this by multiplying $f_i(\overrightarrow{s})$ with τ (equation 13). This parameter is the duration in seconds of the moving average window.

There are again hyperparameters that we can optimise. For this model, this includes (1) Time Resolution, (2) Tuning Resolution (The number of bins used to divide the velocity domain) and (3) Neurons Removed Threshold. These results are in figure 5.

2.4 Hyperparametrization Hacking

To choose the different hyperparameter values, we ran multiple tests with various values and chose the ones that provided the best results.

We defined the best value as the one that minimises the Coefficient of Variation, that is, the one that reduces the relation between the variance and mean of the designated evaluation metric. In this case, we used R^2 as the model performance metric.

For the final results, we also used the R^2 metric as this is the most common evaluation metric used in other models, allowing for comparisons with other studies [17].

3. Results

This section first has the hyperparametrization hacking for each model and then the comparison between designed and state-of-the-art models.

For both models, we had to optimise three hyperparameters: (1) Time Resolution, (2) Neurons Removal Threshold and (3) Tuning Resolution (figure 5). The appendix has all the computational specs specified.

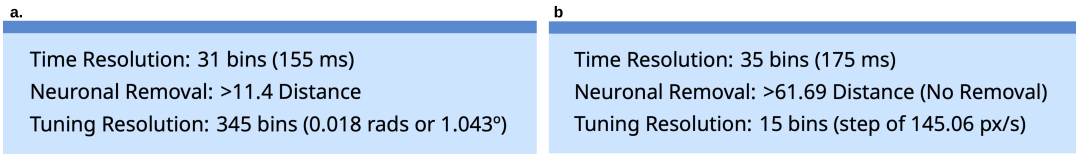

Figure 5. (a) Hyperparameters optimized values for the Population Vector Model **(b)** Hyperparameters optimized values for the Naïve Bayes Model

Once the hyperparameters were optimised we got R^2 for both final models represented in figure 6.[2] These values were obtained by running a 10-fold separation of the dataset, originating ten different training sets that we used to train the optimised models. The decoders were all applied to the test set. The standard error was also obtained [18].

As one can check, we obtained 0.24 R^2 for the PPV, while the NB obtained a 0.45 value. It is relevant to verify that we can say with confidence that the Population Vector is better than nothing (p-value < 0.01) and that Naïve Bayes is better than the Population Vector (p-value < 0.01).

[2]The represented R^2 is the mean R^2 of both x velocity and y velocity predictions

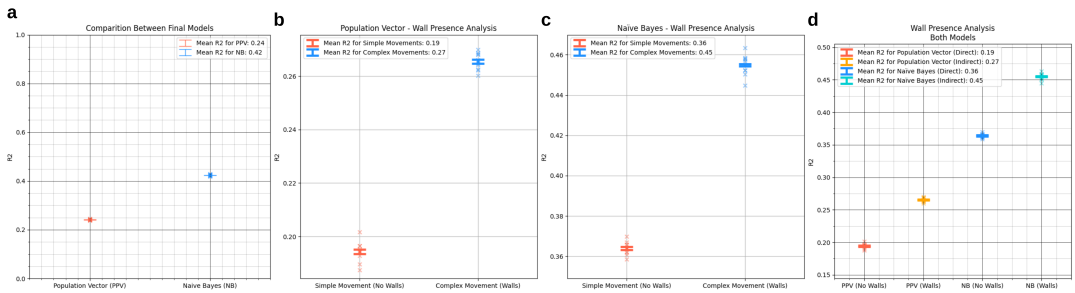

Figure 6. **(a)** Population Vector final model (R^2 = 0.24) and Naïve Bayes model (R^2 = 0.45) performance in scale **(d)** Population Vector model wall comparison. This model performed better when there were walls to avoid (0.27 over 0.19 in R^2) **(e)** Naïve Bayes model wall comparison. This model also performed better when there were walls to avoid (0.45 over 0.36 in R^2) **(f)** Both models result in the same scale for comparison

We run another test with the optimised models. This time, we split the test dataset into direct and indirect datasets. The first one comprises all the trials with no walls, so the trials were quicker with faster movements (like Trial 4 from figure 1a). The indirect set has the remaining trials where the monkey was slower to avoid the walls (like Trial 0 from figure 1a). The model's performance was different, with both having better results when the monkey had obstacles. Figure 6e as the mentioned results, where there is statistical validity (p-value < 0.01).

To allow better inspection of the model's performance, the trajectories were also plotted (figure 7). In this plot, the separation between direct and indirect trials is evident by the presence or the lack of obstacles. It is worth noticing that an arbitrary and not-optimised coefficient achieved the PPV trajectories. This coefficient converted the directional vector into a velocity vector.

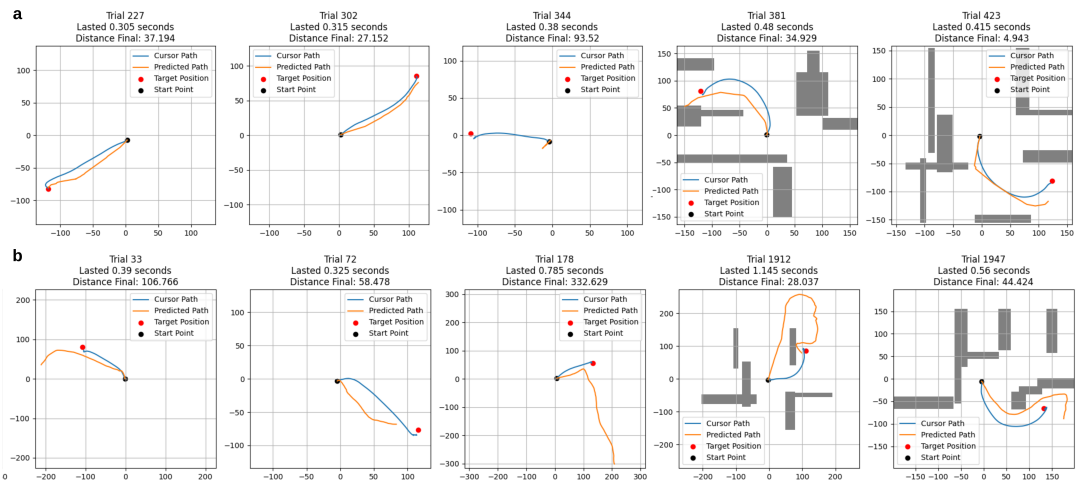

Figure 7. **(a)** Example of predicted trajectories by the optimised Population Vector Model on direct and indirect trials **(b)** Example of predicted trajectories by the optimised Naïve Bayes Model on direct and indirect trials

4. Discussions

It is now mandatory to understand what went well and what went wrong with these models.

4.1 Why did Naïve Bayses perform better than Population Vector?

Let's first start with the first decoder. Some tuning curve shapes obtained verify the intuition that neurons tune to a particular direction. However, it is fundamental to highlight that multiple neurons do not tune to anything. These neurons are likely part of the M2 region as this part of the motor cortex is responsible for motor preprocessing information. Neurons from this region tune better as they can send direct impulses to the spinal cord [10]. This limitation will compromise the data the model can use to decode the hand position as fewer neurons will contribute to both decoders.

The rationale behind tuning curves seems to make sense. Nonetheless, the performance is far from what one could expect in hindsight, especially when comparing both models. This difference has to do with their main distinction. The Population Vector is too strict in how it handles the tuning information. This model will assign one and only one direction to each neuron, reducing their contribution to different-sized but directionally equal vectors.

This difference is well reflected by the improved performance of the Naïve Bayes model as this decoder changes the way we approach tuning assignment to the neurons. Instead of assigning one preferred direction, we now assign a probabilistic field. In other words, the neuron is now more *likely* to code some particular behaviour. This flexibility improved the model's performance. However, again, we are still far from acceptable decoded trajectories.

4.2 How does this relate to other models?

If we check EvalAI (https://eval.ai/web/challenges/challenge-page/1256/overview) best Neural Latent Models used in this dataset, we can verify the best model to have a R^2 of 0.85 [19]. So what is lacking in ours?

It is mandatory to understand what these EvalAI models are. After a significant first step, these R^2 values were obtained by applying a simple linear regression with ridge regression [12]. These models are called Neural Latent Models, which are, in a sense, models used for data preprocessing.

In our models, the only preprocessing used was the use of a simple moving average to turn 0s and 1s into a firing rate, but that is still a quick fix that might be hurting our performance.

4.3 How should we deal with the Firing Rate?

To improve our model's performance, an upgrade to the preprocessing step is needed. First, we analyse the limitations of using a moving average to obtain the firing rate. By using a simple moving average, we introduce two problems. If we are working on a practical working model, we would add extra latency to our movement predictions since the user would have to wait for the moving average to reach the desired position (about 150 ms, depending on the model). This problem could be avoided by resorting to some clever tricks. The second problem, however, is more challenging to deal with, as the method works like a functional low-pass filter, removing noise and possibly some information.

We discussed some possible solutions, one being averaging not over time but rather over a population of neurons. Because neurons are not independent like we assumed for our model development, we wanted a way to group them into look-a-like neuron populations. The firing rate could then be obtained by averaging over this population, effectively mitigating the limitations of our current solution.

However, there are still some challenges that can be considered work for the future. How would we group them? In the NB, what distribution would be used for the posterior, as we would now need a multi-variable distribution?

4.4 How should we deal with the different speeds?

Another topic of discussion is the difference in speed during trials. Why do the models perform differently when the trial is a simple reaching task over when the monkey needs to avoid the walls? The appendix figure 7 demonstrates this difference in performance where direct and indirect trials' trajectories are present side by side. In some direct trials, the decoder seems unable to start moving. It is worth noticing that this phenomenon is only present in the NB direct trials, while in the PPV direct trials, the reduced correlation seems to be due to the lack of the model's precision.

The answer seems to relate to the underlying characteristic of slow vs quick movements. Because slower movements take time, they have more entries in the dataset biasing the Naïve Bayes model to this type of movement, as figure 4b and c suggests.

This, however, does not explain the same bias in the Population Vector model. A more in-depth analysis is needed, where the previous problem of applying a low-pass filter might have a role.

5. Conclusion

As paralysis and other types of motor impairment become more prevalent in the population, solutions are required to improve the quality of life of these people. The use of BMI to solve these problems shows great promise, with many relevant studies elucidating its capabilities. The use of these devices to directly or indirectly recover motor control dominates the race of BMI development.

With the help of tuning curves, one can find models that run away from the use of neuronal networks, allowing simultaneous learning of the brain network that, consequently, unlocks better and planned upgrades to those same models.

With the use of simple technical or biological models such as the Population Vector and Naive Bayes, one can have a first contact with this problem with a clearer understanding of the tuning curves producing average results.

More than model development, this problem needs to be addressed on more than that front by developing more thought and complex preprocessing steps.

References

[1] Russell A. Poldrack and Martha J. Farah. "Progress and challenges in probing the human brain". en. In: *Nature* 526.7573 (Oct. 2015), pp. 371–379. ISSN: 1476-4687. DOI: 10.1038/nature15692.

[2] Alex de Vries. "The growing energy footprint of artificial intelligence". English. In: *Joule* 7.10 (Oct. 2023), pp. 2191–2194. ISSN: 2542-4785, 2542-4351. DOI: 10.1016/j.joule.2023.09.004.

[3] Vijay Balasubramanian. "Brain power". In: *Proceedings of the National Academy of Sciences of the United States of America* 118.32 (Aug. 2021), e2107022118. ISSN: 0027-8424. DOI: 10.1073/pnas.2107022118.

[4] Arsalan Alizadeh, Scott Matthew Dyck, and Soheila Karimi-Abdolrezaee. "Traumatic Spinal Cord Injury: An Overview of Pathophysiology, Models and Acute Injury Mechanisms". In: *Frontiers in Neurology* 10 (Mar. 2019), p. 282. ISSN: 1664-2295. DOI: 10.3389/fneur.2019.00282.

[5] URL: https://www.christopherreeve.org/todays-care/paralysis-help-overview/stats-about-paralysis/.

[6] Ahmet Höke and Thomas Brushart. "Challenges and Opportunities for Regeneration in the Peripheral Nervous System". In: *Experimental neurology* 223.1 (May 2010), pp. 1–4. ISSN: 0014-4886. DOI: 10.1016/j.expneurol.2009.12.001.

[7] Aleksandra Kawala-Sterniuk, Natalia Browarska, Amir Al-Bakri, Mariusz Pelc, Jaroslaw Zygarlicki, Michaela Sidikova, Radek Martinek, and Edward Jacek Gorzelanczyk. "Summary of over Fifty Years with Brain-Computer Interfaces—A Review". In: *Brain Sciences* 11.1 (Jan. 2021), p. 43. ISSN: 2076-3425. DOI: 10.3390/brainsci11010043.

[8] Fabien B. Wagner et al. "Targeted neurotechnology restores walking in humans with spinal cord injury". en. In: *Nature* 563.7729 (Nov. 2018), pp. 65–71. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0649-2.

[9] Elon Musk and Neuralink. "An integrated brain-machine interface platform with thousands of channels". en. In: (July 2019), p. 703801. DOI: 10.1101/703801. URL: https://www.biorxiv.org/content/10.1101/703801v2.

[10] Chaery Lee, Yeonjun Kim, and Bong-Kiun Kaang. "The Primary Motor Cortex: The Hub of Motor Learning in Rodents". In: *Neuroscience* 485 (Mar. 2022), pp. 163–170. ISSN: 0306-4522. DOI: 10.1016/j.neuroscience.2022.01.009.

[11] Sara Fabbri, Alfonso Caramazza, and Angelika Lingnau. "Tuning Curves for Movement Direction in the Human Visuomotor System". In: *The Journal of Neuroscience* 30.40 (Oct. 2010), pp. 13488–13498. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.2571-10.2010.

[12] Felix Pei et al. "Neural Latents Benchmark '21: Evaluating latent variable models of neural population activity". en. In: *arXiv.org* (Sept. 2021). URL: https://arxiv.org/abs/2109.04463v4.

[13] Apostolos P. Georgopoulos, Andrew B. Schwartz, and Ronald E. Kettner. "Neuronal Population Coding of Movement Direction". In: *Science* 233.4771 (Sept. 1986), pp. 1416–1419. DOI: 10.1126/science.3749885.

[14] Kechen Zhang, Iris Ginzburg, Bruce L. McNaughton, and Terrence J. Sejnowski. "Interpreting Neuronal Population Activity by Reconstruction: Unified Framework With Application to Hippocampal Place Cells". In: *Journal of Neurophysiology* 79.2 (Feb. 1998), pp. 1017–1044. ISSN: 0022-3077. DOI: 10.1152/jn.1998.79.2.1017.

[15] 2019. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html.

[16] Uri T. Eden and Mark A. Kramer. "Drawing inferences from Fano factor calculations". In: *Journal of Neuroscience Methods* 190.1 (June 2010), pp. 149–152. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2010.04.012.

[17] Joshua I. Glaser, Ari S. Benjamin, Raeed H. Chowdhury, Matthew G. Perich, Lee E. Miller, and Konrad P. Kording. *Machine learning for neural decoding*. en. Aug. 2017. URL: https://arxiv.org/abs/1708.00909v4.

[18] Douglas G Altman and J Martin Bland. "Standard deviations and standard errors". In: *BMJ: British Medical Journal* 331.7521 (Oct. 2005), p. 903. ISSN: 0959-8138.

[19] Sean M. Perkins, John P. Cunningham, Qi Wang, and Mark M. Churchland. "Simple decoding of behavior from a complicated neural manifold". en. In: (Apr. 2023), p. 2023.04.05.535396. DOI: 10.1101/2023.04.05.535396. URL: https://www.biorxiv.org/content/10.1101/2023.04.05.535396v2.

Appendix 1. Supplementary figures

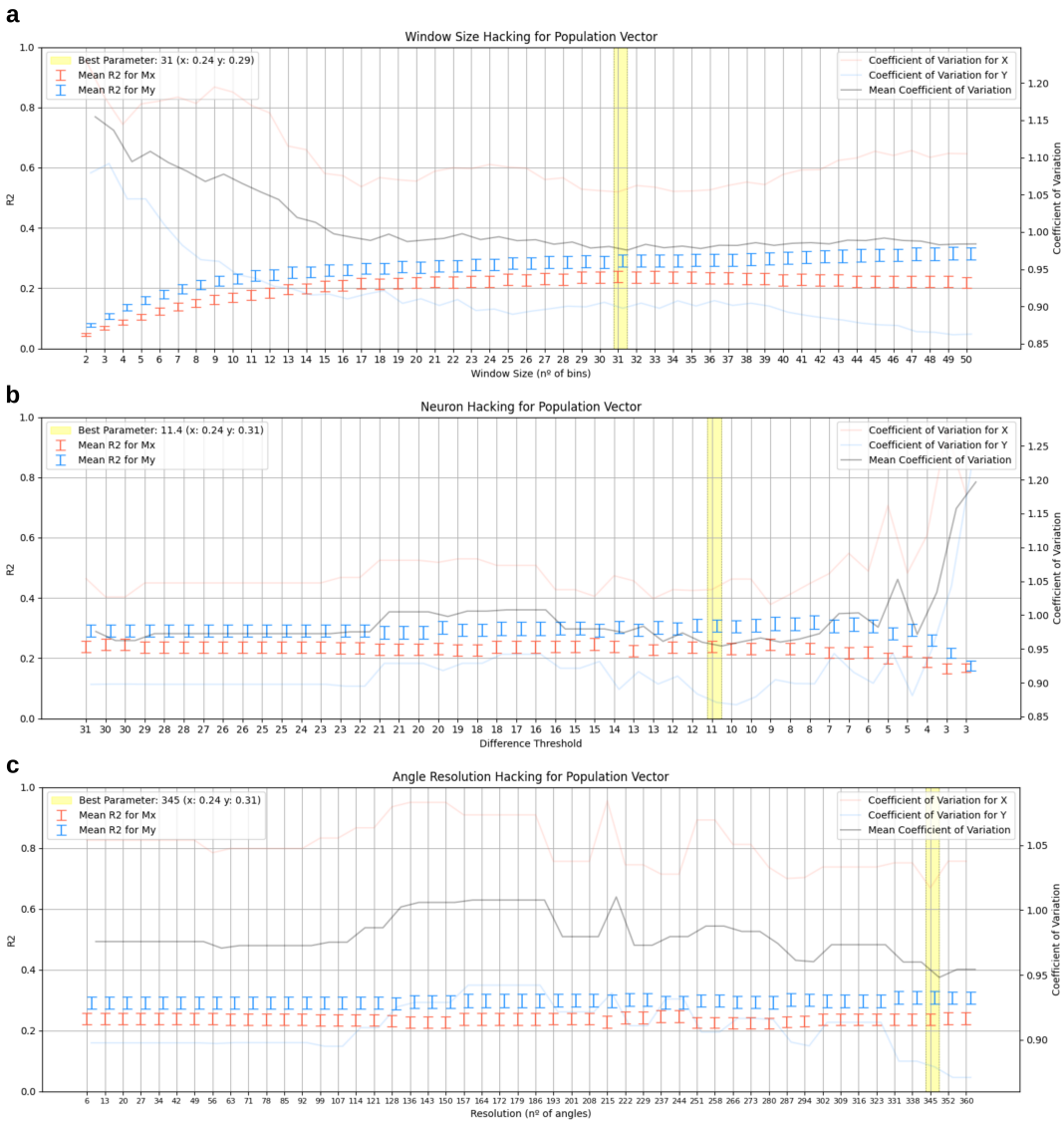

Figure 8. **(a)** Hyperparametrization hacking for the Time Resolution Hyperparameter. The minimal Coefficient of Variance of the R^2 evaluation parameter was minimal when the model ran with a window of size of 31 bins or 155 ms **(b)** Hyperparametrization hacking for the Neuron Removal Threshold Hyperparameter. The minimal Coefficient of Variance of the R^2 evaluation parameter was minimal when the model ran with only neurons that had fitted curves with a Euclidean distance smaller than 11.4 **(c)** Hyperparametrization hacking for the Tuning Resolution Hyperparameter. The minimal Coefficient of Variance of the R^2 evaluation parameter was minimal when the model had the tuning curves built with 345 bins or an angle step of 0.0182 rads or 1.0435º

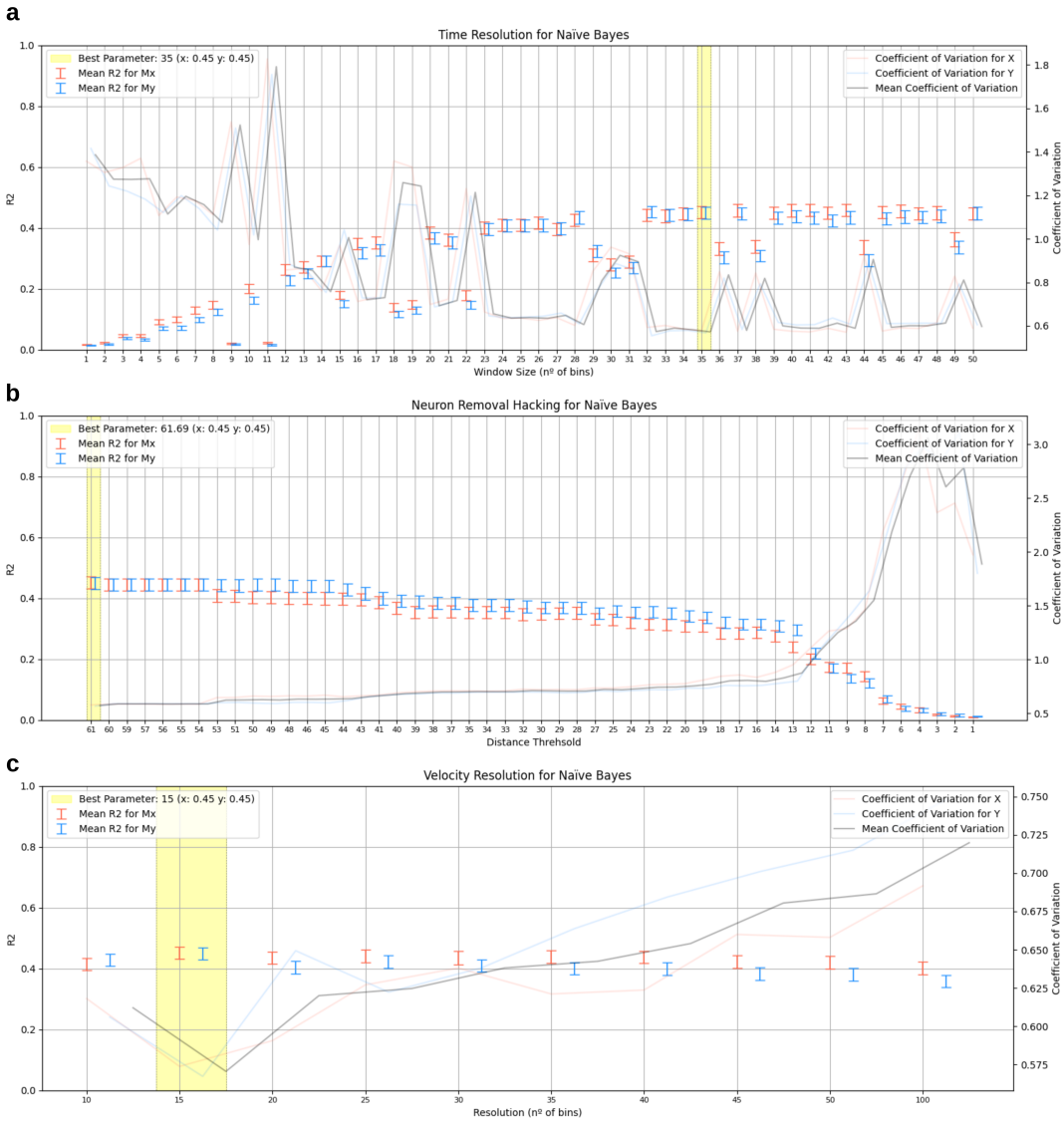

Figure 9. **(a)** Hyperparametrization hacking for the Time Resolution Hyperparameter. The minimal Coefficient of Variance of the R^2 evaluation parameter was minimal when the model ran with a window of size of 35 bins or 175 ms **(b)** Hyperparametrization hacking for the Neuron Removal Threshold Hyperparameter. The minimal Coefficient of Variance of the R^2 evaluation parameter was minimal when the model ran with all neurons **(c)** Hyperparametrization hacking for the Tuning Resolution Hyperparameter. The minimal Coefficient of Variance of the R^2 evaluation parameter was minimal when the model had the tuning surfaces built with 15 bins or a velocity step of 145.06 px/s

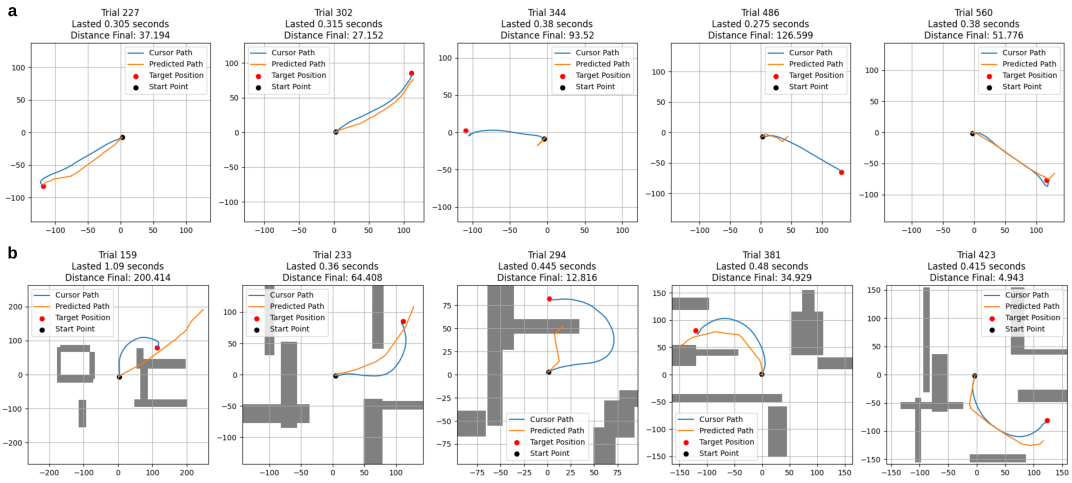

Figure 10. **(a)** Example of predicted trajectories by the optimised Naïve Bayes Model on direct trials **(b)** Example of predicted trajectories by the optimised Naïve Bayes Model on indirect trials

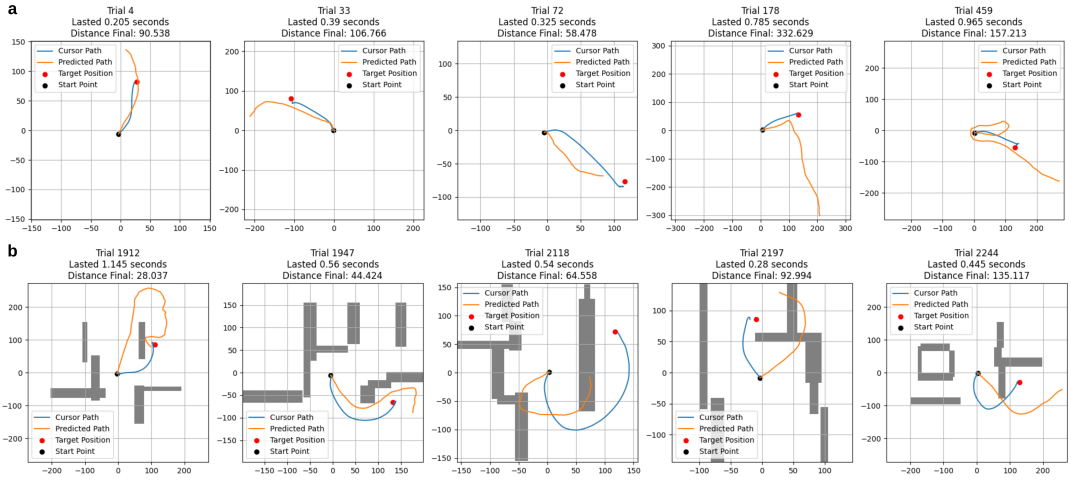

Figure 11. **(a)** Example of predicted trajectories by the optimised Population Vector Model on direct trials **(b)** Example of predicted trajectories by the optimised Population Vector Model on indirect trials

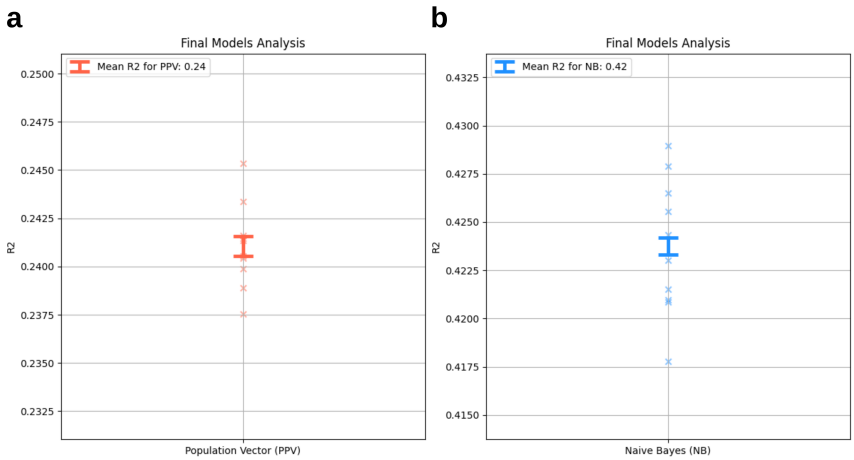

Figure 12. **(a)** Population Vector final model performance. It produced a mean R^2 of about 0.24. The greyed-out crosses represent the 10-fold runs, while the error bars represent the standard error **(b)** Naïve Bayes final model performance. It produced a mean R^2 of about 0.42. The greyed-out crosses represent the 10-fold runs, while the error bars represent the standard error

Table 1. Neuron naming reference

Neuron	1011	1021	1022	1031	1032	1033	1034	1041	1051	1061	1062	1063	1071	1072
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Neuron	1091	1092	1101	1102	1111	1112	1121	1122	1123	1141	1151	1161	1162	1171
Index	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Neuron	1182	1183	1191	1192	1193	1211	1231	1232	1233	1241	1251	1252	1261	1271
Index	28	29	30	31	32	33	34	35	36	37	38	39	40	41
Neuron	1291	1292	1341	1361	1362	1372	1381	1401	1411	1421	1431	1441	1442	1461
Index	42	43	44	45	46	47	48	49	50	51	52	53	54	55
Neuron	1471	1481	1501	1502	1511	1512	1541	1561	1562	1581	1582	1601	1661	1662
Index	56	57	58	59	60	61	62	63	64	65	66	67	68	69
Neuron	1663	1691	1701	1702	1711	1721	1741	1751	1752	1761	1781	1791	1801	1812
Index	70	71	72	73	74	75	76	77	78	79	80	81	82	83
Neuron	1831	1841	1851	1861	1881	1891	1901	1902	2051	2121	2131	2132	2151	2161
Index	84	85	86	87	88	89	90	91	92	93	94	95	96	97
Neuron	2181	2191	2192	2231	2232	2241	2251	2252	2271	2272	2273	2281	2301	2302
Index	98	99	100	101	102	103	104	105	106	107	108	109	110	111
Neuron	2311	2321	2331	2341	2351	2352	2361	2371	2381	2391	2392	2401	2412	2413
Index	112	113	114	115	116	117	118	119	120	121	122	123	124	125
Neuron	2421	2422	2423	2431	2441	2451	2452	2453	2461	2471	2472	2481	2491	2492
Index	126	127	128	129	130	131	132	133	134	135	136	137	138	139
Neuron	2501	2541	2542	2561	2571	2581	2591	2601	2602	2611	2612	2621	2631	2641
Index	140	141	142	143	144	145	146	147	148	149	150	151	152	153
Neuron	2651	2691	2721	2731	2732	2741	2742	2751	2761	2762	2771	2791	2792	2801
Index	154	155	156	157	158	159	160	161	162	163	164	165	166	167
Neuron	2811	2821	2841	2842	2861	2862	2871	2881	2882	2911	2931	2941	2951	2961
Index	168	169	170	171	172	173	174	175	176	177	178	179	180	181