Spiking Neuron Group Pipeline for Edge Orientation Classification

Surya Pandiaraju

April 14, 2021

Notebook Link: https://colab.research.google.com/drive/1_ZnJBwoyWczF_uwxJrfNAoQvpg3q5viJ

Abstract

Touch is one of five primary senses humans possess. It involves processes from mechanoreceptor stimulation to processing and perception in the somatosensory cortex. The computational modelling of touch at a neuronal level allows for a greater understanding of underlying biological mechanisms and can also be translated to the design of neural rehabilitation devices for individuals, such as those who rely on prosthetic arms. Neuromorphic approaches can be taken to represent an input and process it appropriately for a neurostimulation pipeline. In this study, images serving to emulate force outputs of palpated edges of various orientations (0^o to 90^o at 10^o intervals) over time are used as input to a spiking neuron group utilizing a generic Leaky Integrate and Fire neuron model and Izhikevich models. Specifically, three Izhikevich with different parameters and equations are used to emulate a static slowly-adapting, dynamic slowly-adapting, and rapidly-adapting mechanoreceptors. To assess the effectiveness of spike-train representation, k-nearest neighbours classification was used. After generating a neuron group's spike-train responses to a given input, principal-component analysis is performed for dimension reductionality and transformed data is used to train a k-nearest neighbours model to classify the palpated edge angle. Accuracy and confusion matrix metrics were used to evaluate the performance of different neuron model pipelines. The slowly-adapting dynamic neuron model outperformed the others, achieving a peak of 96.3% accuracy on 300 testing samples, followed by the slowly-adapting static (89.7%), and the Leaky Integrate and Fire (81.3%). Examining the confusion matrices, classifications were all within 10%. The rapidly-adapting neuron performed quite poorly in classification with an accuracy of ~10%.

1. Introduction

1.1 - Physiology of Tactile Sensation

The sense of touch is one of five primary senses human-beings possess. The pathway from stimulus contact to perception involves a series of biological processes that are able to provide the brain descriptive information that can be integrated with other sensory modalities.

Table 1: Mechanoreceptors and Functions

Mechanoreceptor	Adaptation	Function
Merkel's Disc	Slowly Adapting I	Sustained responses to stimulation/pressure. Small receptive field (RF)
Ruffini End-Organ	Slowly Adapting II	Respond to stretch. Large RF.
Meissner's Corpuscle	Rapidly Adapting I	Respond to flutter and slip. Small RF.
Pacinian Corpuscle	Rapidly Adapting II	Respond to vibration. Large RF.

Brief description of the four major mechanoreceptors and their functions.

Touch begins with contact between the skin and an object. This results in the activation of four major mechanoreceptors: Meissner's corpuscles, Merkel's discs, Ruffini end-organs, and Pacinian corpuscles. Table 1 provides a summary of these mechanoreceptors. Depending on the location of the body, the density of each of these mechanoreceptors differs such that areas of greater mechanoreceptor density are more sensitive. For instance, the human fingertip has an average density of mechanoreceptors of 240/cm^2 whereas the palms have an average density of 58/cm^2 [1]. Mechanoreceptors generate a number of low-level inputs that with subsequent processing reveal high-level characteristics of the object stimulus such as shape, size, and texture [2].

Response characteristics of mechanoreceptors can be described as slowly or rapidly adapting. Slowly adapting mechanoreceptors fire in response to the static indentation of the skin, such as in the event of sustained pressure [2]. Rapidly adapting mechanoreceptors on the other hand are more likely to fire on object boundaries where there is an onset or offset tactile sensation. The responses of the neurons innervated by these mechanoreceptors are integrated in the cuneate nucleus (CN), located in the medulla oblongata, where lateral inhibition integrates and filters neuronal firing [2]. Finally, these processed signals are relayed by the thalamus to the somatosensory cortex where the stimulus is identified and perceived. Unlike the relatively tactile afferents, these cortical neurons have much more complex receptive fields as a result of the intermediate processing.

1.2 - Ongoing Research in Robotic/Prosthetic Tactile Sensation

Reproducing tactile sensation is an ongoing research area in rehabilitation robotics as prosthetics lack the tactile capabilities of human hands [4]. Better understanding of the biocomputational principles underlying sensation would enable the design of better algorithms and systems that could restore the sensation of touch to upper-limb amputees [3][4]. This has motivated the use of neural code based systems, emphasizing the timing and sequence of response spikes, for the emulation of sensory inputs and exploiting them for neurostimulation. Applications of these principles have been studied to discriminate textures, explore indentation conditions (such as nodule depth under a surface), modulating hand grip, and edge and orientation detection [3][4].

1.3 - Project Summary and Relevance

This project explored a biologically motivated pipeline model of edge detection. Using simulated edge data of various orientations ranging from 0^o to 90^o at 10^0 intervals, different neuron models are used to convert the input into output spike trains. Specifically, a generic Leaky Integrate and Fire neuron along with three Izhikevich models are used. The Izhikevich models consist of different parameters and equations in order to emulate static slowly-adapting, dynamic slowly-adapting, and rapidly-adapting mechanoreceptor-coupled afferents. To emulate the filtering performed by the CN on these incoming inputs, principal component analysis of these inputs are used, thus reducing feature size. Finally, flattened PCA components are used in a

k nearest neighbour classifier to classify edge orientation of the input. Models are evaluated based on KNN accuracy and confusion matrix metrics. Simulation is performed using the Brian2 computational neuroscience library in Python. Results from this study can be used to motivate future studies with more elaborate pipelines, such as multiple neuron groups, and determine important considerations in the implementation of spiking neuron models in the classification of inputs.

2. Methods

2. 1 - Edge Simulation

In order to simulate edge detection, the palpation of an edge with a fingertip scanning vertically over a surface as seen in Figure 1.

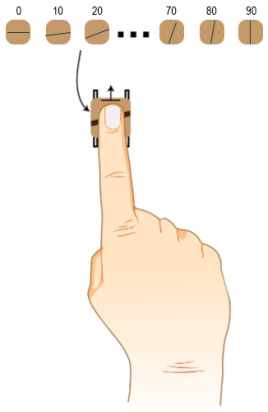

Figure 1: Palpation of finger across edge of particular orientation. []

To do so, 10 121x121 images containing linearly decaying grayscale gradients at different angles were used. Examples of 0^o, 40^0 and 90^o can be seen in Figure 2. For data augmentation, each image was copied 100 times and Gaussian noise was added. In total, 1000 edge orientation images were generated and used as inputs into the neuron model.

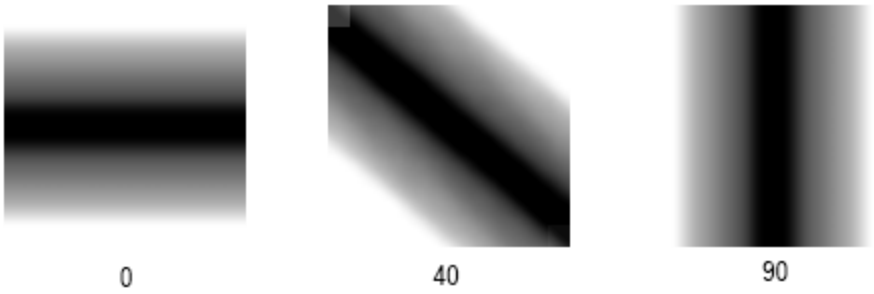

Figure 2: Examples of images used to emulate inputs prior to addition of Gaussian noise.

Rows of these images represent the force to be converted to current, assuming a linear conversion from force to current, for a given neuron index. Column indices represent the time points for which the edge is being palpated. Given the shape of the input, the neurons being stimulated by these edges would effectively be arranged in a horizontal bar sliding across the edge.

2.2 - Neuron Models

The neuron models contain parameters and equations that cause them to change voltage over time and generate a spike once a threshold is reached followed by a reset. In this project, neurons were input with a current from a given loaded image. Loaded images were converted to Brian2 TimedArray objects (dt = 1 ms) to enable an indexed input over time for each neuron in the group. All simulations were run for 120 ms. All of these simulations assume that every neuron receives current proportional to the force applied to the indented skin located immediately above it and that force decreases linearly from the edge.

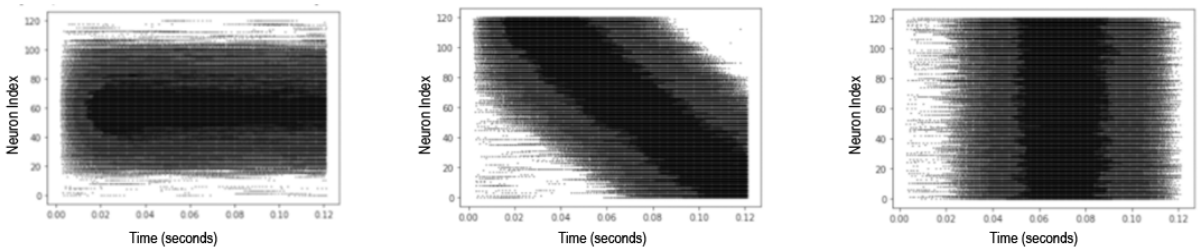

Figure 3: Examples of spiking patterns from neurons (slowly adapting dynamic) for 0^o, 50^o, and 90^o.

2.2.1 - Leaky Integrate and Fire

For the Leaky LIF neuron, initial testing revealed A = 1.5 to be a sufficient I_{in} modulator. τ is the time constant of this model [5].

$$\frac{dv}{dt} = \frac{A \cdot I_{in} - v}{\tau}$$

2.2.2 - Izhikevich Neurons

Izhikevich neurons using the following equations were used:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + k_{SA}I_{SA} + k_{RA}I_{RA}$$

$$\frac{du}{dt} = a(bv - u)$$

$$if\ v \geq 30mV.\ then \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases}$$

In this model, v represents the membrane potential and u is the adaptation variable. Their derivatives are parameterized by other constants, described in Table 2.

Table 2: Values of constants

Parameter	**Value**	**Parameter**	**Value**
a	0.02 s^{-1}	d	8 mV
b	0.2	k_{SA}	100 for slowly adapting, 0 for rapidly adapting
c	-65 mV	k_{RA}	0 for slowly adapting, 100 for rapidly adapting

Constant values used for neuron simulation.

Depending on the specific neuron subtype, different equations were used to define the input current.

2.2.2.1 - Slowly Adapting Static

The input current I_{in} which is linearly proportional to pixel values of the simulated edge images is modulated by a scalar factor k_I= 1 prior to being input to the neuron group [1].

$$I_{SA} = k_1 I_{in}$$

2.2.2.2 - Slowly Adapting Dynamic

The dynamic model has additional equations for the derivatives of I_{SA} and x, an auxiliary variable [1]. τ_{d_SA} and τ_{r_SA} respectively modulate the rise and decay time of the model. After initial adjustment and testing testing, k_1 was selected to be 50, $k_2 = 2$, and $k_3 = 3$ to optimize performance,

$$\tau_{d\_SA} \frac{dI_{SA}}{dt} = x - I_{SA}$$

$$\tau_{r\_SA} \frac{dx}{dt} = (k_2 \frac{dI_{in}}{dt} + k_1 I_{in} - x)$$

2.2.2.3 - Rapidly Adapting

For the rapidly adapting model, k_3 is 2 and τ_{RA} is the time constant. k_{RA} was adjusted and modified using values on orders of 10 but as seen in results, even the best value of k_{RA} was not great for classification [1].

$$\tau_{RA} \frac{dI_{RA}}{dt} = k_3 |\frac{dI_{in}}{dt}| - I_{RA}$$

2.3 - Principal Component Analysis

Principal component analysis (PCA) is used to reduce the dimensionality or size of the feature space. The size of data before PCA is quite large. The goal of this method is to summarize the information of the inputs by using the least number of time series and for this project, 1, 3, 5, and 10 PCA-fitted time series were used. To perform PCA, a covariance matrix is calculated and eigenvalue decomposition is performed. Eigenvectors are ordered to point in the direction in which there is the greatest to least variance in the dataset and can be used to project the original data. The resulting dataset gives a better representation of the overall dataset by filtering out the least significant components of signals and leaving ones containing the most important spiking responses.

2.4 - K Nearest-Neighbours Classification

K Nearest-Neighbours (kNN) classification analysis was used to recognize the orientation of the angle at which the input currents were situated. KNN was performed after PCA analysis to classify using the most important components of the signals generated by the 121 neurons. This leads to a more accurate classifier and smaller run-time. A kNN classifier is a non-parametric algorithm that uses the chosen number of nearest neighbours at a parameter. A variety of k values were tested for the model to determine the one that resulted in the highest accuracy. K = 5 was tested as it was the value implemented in the paper [1]. Additionally, values above and below k = 5 (from 1 to 10) were all considered to determine the most effective k number. As a result, a k = 10 was determined to produce the highest accuracy score for the kNN classifier. The input to the kNN model was the flattened spike train generated from the 5 important signals from PCA producing a collective spike train of 550 points for a sample. The output is the angle classification of edge orientation. A set of 70% of the data (flattened pca data and corresponding labels) was used to train the model, while the remaining 30% was to test. The process of the kNN classifier includes a distance calculation, determination of closest neighbours and a vote for the corresponding label. The distance between the input signal and the other points are calculated using the Minkowski distance metric as shown below, where p, the order, is equal to 1.

$$D(X,Y) = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^{p} \right)^{\frac{1}{p}}$$

Next the number k=10, in this scenario, neighbours are determined by the smallest distance between the input point and the other objects. Each object then votes for the class of the input points and the class with the most votes is taken as the prediction of the angle orientation (10°, 20°, …, 90°).

2.4.1 - Metrics

The metrics used to analyze the tested dataset were a corresponding accuracy score and a confusion matrix. The accuracy score compared the set of labels predicted for the data against the true set of labels and produced a number that shows the percentage of correctly classified samples. A confusion matrix was also utilized to evaluate the performance of the model by displaying the number correctly and incorrectly classified labels and to identify which class the model is incorrectly classifying the sample as.

3. Results

3.1 - Neuron Model 1, Leaky Integrate and Fire

Table 3 - Leaky Integrate and Fire Simulation Results

Number of PCA Components	**Number of Neighbours**	**Accuracy**
1	1	96
	2	98.6
	3	96.6
	4	98.3
	5	97.3
	6	99
	7	97.6
	8	98.6
	9	97.3
	10	98.3

3	1	83.3
	2	85
	3	81
	4	86
	5	82
	6	85.3
	7	82.6
	8	85.3
	9	83.3
	10	85
5	1	75.3
	2	80.6
	3	79.6
	4	77.3
	5	79.6
	6	77.6
	7	80
	8	78.3
	9	77
	10	77.3
10	1	71.6
	2	71.6
	3	77.3
	4	72.3
	5	74

	6	72
	7	71.6
	8	73
	9	72.3
	10	73.3

Results using 1, 3, 5, and 10 PCA components and 1-10 Number of Neighbours

Table 4 - Best Confusion Matrix for Leaky Integrate and Fire Neuron

	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
0	35	0	0	0	0	0	0	0	0	0
10	0	20	0	0	0	0	0	0	0	0
20	0	0	36	1	0	0	0	0	0	0
30	0	0	0	32	2	0	0	0	0	0
40	0	0	0	0	24	0	0	0	0	0
50	0	0	0	0	0	33	0	0	0	0
60	0	0	0	0	0	0	30	0	0	0
70	0	0	0	0	0	0	0	23	0	0
80	0	0	0	0	0	0	0	0	29	0
90	0	0	0	0	0	0	0	0	0	35

Confusion matrix for best result using 1 PCA component and 6 nearest neighbours for classification.

3.2 - Neuron Model 2, Slowly Adapting Static

Table 5 - Slowly Adapting Static Simulation Results

Number of PCA Components	**KNN**	**Accuracy**
1	1	94.3
	2	97.7

	3	94.3
	4	97.3
	5	94.3
	6	95.6
	7	94.33
	8	96.3
	9	95
	10	96
3	1	87
	2	91
	3	88.6
	4	91.66
	5	89.33
	6	91
	7	90.33
	8	91.33
	9	90
	10	90.6
5	1	81.3
	2	83
	3	81
	4	85.3
	5	83.3
	6	83.6
	7	83.6

	8	84.3
	9	84
	10	86.3
10	1	80.6
	2	80.6
	3	79.6
	4	83.6
	5	80
	6	83.6
	7	80.3
	8	82.3
	9	81.3
	10	81.6

Results using 1, 3, 5, and 10 PCA components and 1-10 Number of Neighbours

Table 6 - Best Confusion Matrix for Slowly Adapting Static Neuron

	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
0	30	5	0	0	0	0	0	0	0	0
10	2	18	0	0	0	0	0	0	0	0
20	0	0	37	0	0	0	0	0	0	0
30	0	0	0	34	0	0	0	0	0	0
40	0	0	0	0	24	0	0	0	0	0
50	0	0	0	0	0	33	0	0	0	0
60	0	0	0	0	0	0	30	0	0	0
70	0	0	0	0	0	0	0	23	0	0
80	0	0	0	0	0	0	0	0	29	0

90	0	0	0	0	0	0	0	0	0	35

Confusion matrix for best result using 1 PCA component and 2 nearest neighbours for classification.

3.3 - Neuron Model 3, Slowly Adapting Dynamic

Table 6 - Slowly Adapting Dynamic Simulation Results

Number of PCA Components	**Number of Neighbours**	**Accuracy**
1	1	99.7
	2	100
	3	100
	4	100
	5	100
	6	98.7
	7	98.7
	8	98.7
	9	98.7
	10	98.7
3	1	1
	2	99.3
	3	99.7
	4	98.3
	5	98.7
	6	97.7
	7	97.3
	8	97.3
	9	97.3

	10	97.0
5	1	98.3
	2	97.3
	3	97.0
	4	96.7
	5	96.7
	6	96.7
	7	96.3
	8	96.3
	9	95.7
	10	96.7
10	1	96.3
	2	95.3
	3	95.0
	4	94.7
	5	95.3
	6	95.3
	7	94.0
	8	94.0
	9	93.3
	10	94.3

Results using 1, 3, 5, and 10 PCA components and 1-10 Number of Neighbours

Table 7 - Best Confusion Matrix for Slowly Adapting Dynamic Neuron

	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
0	35	0	0	0	0	0	0	0	0	0

10	0	20	0	0	0	0	0	0	0	0
20	0	0	36	1	0	0	0	0	0	0
30	0	0	0	32	2	0	0	0	0	0
40	0	0	0	0	24	0	0	0	0	0
50	0	0	0	0	0	33	0	0	0	0
60	0	0	0	0	0	0	30	0	0	0
70	0	0	0	0	0	0	0	23	0	0
80	0	0	0	0	0	0	0	0	29	0
90	0	0	0	0	0	0	0	0	0	35

Confusion matrix for best result using 1 PCA component and 2 nearest neighbours for classification.

3.4 - Neuron Model 4, Rapidly Adapting

Table 8 - Rapidly Adapting Simulation Results

Number of PCA Components	**Number of Neighbours**	**Accuracy**
1	1	0.0766
	2	0.1066
	3	0.0966
	4	0.1133
	5	0.1066
	6	0.12
	7	0.1133
	8	0.11
	9	0.1233
	10	0.12
3	1	0.12

	2	0.14
	3	0.13
	4	0.13
	5	0.14
	6	0.13
	7	0.11
	8	0.12
	9	0.09
	10	0.1
5	1	0.08
	2	0.09
	3	0.1
	4	0.1
	5	0.08
	6	0.09
	7	0.11
	8	0.1
	9	0.1
	10	0.11
10	1	0.09
	2	0.1
	3	0.08
	4	0.1
	5	0.1
	6	0.11

	7	0.1
	8	0.09
	9	0.1
	10	0.09

Results using 1, 3, 5, and 10 PCA components and 1-10 Number of Neighbours

Table 9 - Best Confusion Matrix for Rapidly Adapting Neuron

	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
0	11	9	2	2	1	5	4	1	0	0
10	7	1	1	3	3	0	3	1	1	0
20	7	10	5	3	2	2	2	1	5	0
30	6	10	1	9	1	0	2	3	2	0
40	3	4	4	3	4	2	1	1	1	1
50	2	12	3	5	1	1	4	3	0	2
60	6	6	4	4	1	1	2	1	5	0
70	1	2	2	5	4	3	0	4	1	1
80	7	9	3	2	1	1	2	2	1	1
90	9	6	5	2	2	2	0	2	4	3

Confusion matrix for best result using 3 PCA components and 5 nearest neighbours for classification.

4. Discussion

In this project, a pipeline using emulated tactile sensory data of different edge orientation angles was converted into input currents for a neuron layer of 121 neurons using one of four different neuron models and with the spike trains, PCA was performed to reduce the size of data for kNN classification.

In the first simulation, a leaky integrate-and-fire model was used [5]. This model performed quite well. While the number of nearest neighbours seemed to marginally improve accuracy, more impactful was the number of PCA components used in kNN classification. The PCA fitted data transforms the input data and provides signals projected in the direction of the original data where there is greatest variation. Based on the results it can be concluded that for this given pipeline, a fewer number of PCA components provides the best result, as seen in the over 10% drop in accuracy going from ~97% to ~83% for 1 to 3 components and subsequent ~5% decrease going from 3-5 and 5-10 components. This model performs quite well.

In the second simulation, a slowly adapting static Izhikevich model was used. This model performed marginally worse than the LIF model, but not by much at all, with a peak accuracy of 97.7%. Interestingly, while classification accuracy decreased using a greater number of components, the average drop in accuracy when going from 1 to 3 components was only 5%. Decrease in accuracy from 3-5 components was ~5% and then ~3%, similar to the LIF. In addition, number of neighbours did not have a particularly significant role in impacting classification accuracy

In the third simulation, a slowly adapting dynamic Izhikevich model was employed. This model performed better than both previous models, with a peak classification accuracy of nearly 100%. When using a greater number of components, the accuracy dropped from 1-3, 3-5, and 5-10 components was all in the range of ~1-2%, indicating that the model was a very strong fit and even though the additional components may have made classification less accurate, it was not by much.

The fourth and final simulation leveraged a rapidly adapting model. As seen in the results, this pipeline performs quite poorly, leaving most classifications to chance with a classification accuracy of ~10%. This may in part be due to poor parameter optimization given that ideal values for this neuron were not found unlike the others for which they were and saw drastic improvements in classification accuracy.

Based on these results, it is evident that the slowly-adapting dynamic neuron provides the best classification results for edge-detection (nearly 100%). However, both the LIF and slowly adapting static models are not too far behind with best performances of 98.6% and 97.7%. Impressively, any misclassifications involve confusion with only the adjacent angle orientations for each of these models. The rapidly adapting neuron model performed very poorly across conditions, evident in its classification accuracies and confusion matrices, which may be in part due to poor parameter optimization in scaling input current I_{in} appropriately as well as I_{RA} as it affects the change in voltage over time.

In terms of the most optimal number of components, there is generally a decrease in accuracy in the models as the number of PCA components increases. This provides evidence that there is a lot of unnecessary data for the processing of spike trains and emphasizes the importance of data integration and filtering in physiological systems, such as the cuneate nucleus, to save computational time and resources.

The number of neighbours used for kNN classification resulted in marginal improvements for classification accuracy for this project.

It is also important to note that experimenting with various parameter values for constants of these experiments was important to generating optimal spike trains that could be classified. These parameters are quintessential to ensuring that both sufficient spikes are generated in the appropriate conditions without the addition of too much noise. In the context of neuromodulatory systems, were a spiking neural network or neuron group approach to be adopted, the appropriate amplification of a current would be an important consideration

This project also has a number of limitations. Neuron groups are assumed to be a horizontal bar but in physical systems, would involve a 2-dimensional grid-like input instead. Moreover, the mapping from the physical position of an indentation to the neurons activated would be much more complicated than simply innervating an afferent immediately adjacent to a stimulated mechanoreceptor that solely stimulates it (biological afferents are innervated by a number of different mechanoreceptors [3]).In addition, this model simplifies mechanoreceptors

heavily and does not involve an integrated approach of various mechanoreceptors to produce a current as each tested pipeline relies solely on a type of single neuron model.

This study provides groundwork for much more future work. As mentioned as a limitation, inputs to a grid instead of a horizontal bar of afferents would be better to properly emulate human tactile sensing. To emulate the sensors that would be used on a prosthetic arm, the physical spacing between neurons can be considered when creating the current inputs, reducing the resolution at which input currents are generated as it is possible that they were "too ideal" in this study. Actual sensor data can be used, Angle orientations at smaller intervals, such as 5^{o}, could also be used and the effect on an optimal number of nearest neighbours as that was not found to be particularly significant in this project. The pipeline currently uses PCA for data filtering as a means of modelling processing at the cuneate nucleus. Instead, studies have shown that these neurons can be modelled using an additional neuron group that is then followed by PCA to emulate the physiological process with greater biological fidelity [1]. Furthermore, as angles are not fixed at discrete 10^{o} intervals, a kNN regression could be used instead to make classification a continuous problem with root-mean squared error as a metric.

References

[1] A. Parvizi-Fard, M. Amiri, D. Kumar, M. M. Iskarous, and N. V. Thakor, "A functional spiking neuronal network for tactile sensing pathway to process edge orientation," *Scientific Reports*, vol. 11, no. 1, 2021.

[2] A. S. French and P. H. Torkkeli, "Mechanoreceptors," *Neuroscience and Biobehavioral Psychology*, pp. 689–695, 2009.

[3] D. Kumar, R. Ghosh, A. Nakagawa-Silva, A. B. Soares, and N. V. Thakor, "Neuromorphic approach to tactile edge orientation estimation using spatiotemporal similarity," *Neurocomputing*, vol. 407, pp. 246–258, Sep. 2020.

[4] M. Ortiz-Catalan, E. Mastinu, P. Sassu, O. Aszmann, and R. Brånemark, "Self-Contained Neuromusculoskeletal Arm Prostheses," *New England Journal of Medicine*, vol. 382, no. 18, pp. 1732–1738, 2020.

[5] B. Anderson, "Brian 2¶," *Brian2FinalSubmit*. [Online]. Available: https://brittlab.uwaterloo.ca/assets/rawHtmlPgs/Brian2FinalSubmit.html. [Accessed: 10-Apr-2021].