Clusterless decoding of mouse choices based on Point-wise Neural Network

Bole (James) Pan[1]

Supervised by: Liam Paninski [2], Yizi Zhang [2]

Abstract

Neural decoding is an important task in systems neuroscience for correlating brain activity and behavior. Traditional decoding methods, which require spike-sorting, can prove computationally expensive with the ever-increasing dataset size. Recent advances in clusterless decoding have achieved convincing results but may lack in capturing nonlinear features in the datasets. Here we introduce a new method for clusterless decoding by modifying deep learning networks PointNet and PointNet++. By treating spikes sequences as point clouds, we proposed a permutation-invariant network that also takes into account the temporal feature through the method of time-binning. The method is able to achieve performances on par with the state-of-the-art decoding methods, opening up possibilities and providing insights for further research in neural-network-based clusterless decoding.

1. Introduction

Neural decoding refers to the extraction of semantically meaningful information from brain activity data. It is central to understanding the relationship between neural activities and behaviors in the outside world (1, 2). Most neural decoding techniques require input data, typically electrophysiological recording, to go through "spike-sorting" – a process in which each neuronal spike is assigned to a particular neuron based on waveform features.

Despite recent research developments in spike-sorting that have led to increased performances in neural decoding (3-5), researchers are still actively looking for ways of perfecting the process. Often in spike-sorting, spikes that are too small or too difficult to assign to a specific neuron are discarded, resulting in a net loss of information (6). Moreover, as the capability of neural recording increases exponentially, performing spike-sorting on every new dataset can prove computationally expensive (7). These issues have led to efforts in exploring methods in the family of "clusterless decoding": can we improve decoding by retaining uncertainty about spike assignment?

Previous research in this direction has proved the feasibility of the clusterless approach. The results from Trautmann et al. (7) showed that multichannel thresholding, the method of assigning each spike to a channel instead of a particular neuron, achieved similar performances with spike-sorting. Zhang et al. proposed a density-based neural decoding approach for Neuralpixels

[1] Columbia College, Columbia University; New York, NY, 10027

[2] Department of Statistics and Neuroscience, Center for Theoretical Neuroscience, Grossman Center for the Statistics of Mind, Zuckerman Institute, Columbia University; New York, NY, 10027

recordings that models the distribution of spike features as a mixture of Gaussians. Each spike is assigned to a particular Gaussian distribution with a probability density, thus retaining uncertainty about spike assignment. This was shown to achieve superior performances compared to decoding on spike-sorted data (6). These works all provide the theoretical justification for continued research on decoding behaviors from unsorted Neuralpixels data.

Advanced in neural data processing pipeline has allowed the extraction of an ever-increasing richer set of spike features from multi-channel probes recording. In addition to spike time and waveform features, extractable spike features now also include spike location (*x, y, z*), provided by 2D and 3D spike localization techniques developed from recent research (8, 9). Hence, Neuralpixels recording data can be now processed into the form of spike trains with each spike carrying multiple features, showing a resemblance to point-cloud data.

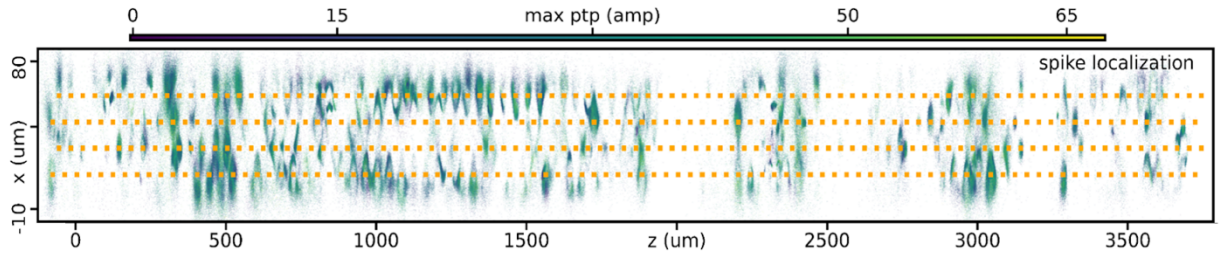

Figure 1. **Spike features *(x, z, max_ptp)* of one trial plotted on a graph**. The horizontal axis is the *z* coordinate of the spike (unit in micrometers), the vertical axis is the *x* coordinate of the spike (unit in micrometers), and the color indicates the maximum peak-to-peak amplitude of the spike (the lighter the color, the larger the amplitude, the stronger the spike). Data presented in this way makes it easy to understand the motivation behind adopting neural networks for point cloud processing on the dataset. Figure from Zhang et al. (2022) (6).

Herein, we explore the application of established supervised learning frameworks for point-cloud processing on spike train data. An underlying hypothesis is that, given labels, a supervised learning neural network is the most suitable paradigm for generating a low-dimensional embedding of electrophysiology recording data that correlates with behaviors, capturing non-linearities in the process. Building upon previous work, we make the following key contributions in this paper:

1. We examine the effects of directly applying established deep learning networks for point clouds, PointNet and PointNet++, on neural data
2. We modify the current PointNet architecture to propose a permutation-invariant network that takes into account the temporal feature and demonstrates its utility in decoding mouse choices
3. We propose a hierarchical way of applying this architecture to capture spatial local structures in addition to temporal features

2. Methods

2.1 PointNet

Point-cloud object classification has been an active area of computer vision, owing to its applications in particularly self-driving (10). PointNet, proposed by Qi et al. in 2016, is the first unified architecture for directly consuming point-cloud data for deep learning (11). PointNet takes into account two key features of the point set: permutation-invariance and transformation-invariance. This means that changing the order of inputting the points and rotating the points clouds should both not affect the result of classification. To make sure the network is permutation invariant, PointNet uses a symmetric function, a max pooling layer. A symmetric function of n variables has the following property:

$$f(x_1, x_2, \ldots, x_n) = f(x_2, x_1, \ldots, x_n) = f(x_3, x_1, \ldots, x_n, x_{n-1}), \ etc.$$

And the symmetry function is represented as following:

$$f(x_1, x_2, \ldots, x_n) \approx g(h(x_1), h(x_2), \ldots, h(x_n))$$

PointNet approximates h by a multi-layer perceptron (MLP) network and g by a max pooling function. By processing each point using the MLP, the network increases the number of features for each spike, effectively reducing potential information loss in the subsequent max pooling step. During experiments, we explore how different choices of function g (like SUM, AVG) affect decoding performances and discuss the possible reasons for variation.

The part of PointNet making the network transformation invariant is not implemented in this project due to the fact that Neuralpixel recording data doesn't exhibit such a property.

2.2 PointNet with Time-binning

PointNet is originally developed for object classification and segmentation on static point clouds, not taking into account the dimension of time. In Neuralpixel recording, spike times are of great significance, as different neurons firing in different orders could suggest completely different outward behaviors of a subject. To address this, we modify the architecture on top of PointNet Vanilla (Figure 2): after the MLP operation, instead of using one symmetric function to generate one global feature from all spikes of (1*embedding_dimension), we first split the entire interval of one trial (t seconds in duration) into k time bins, each t/k seconds long. The k time bins, therefore, would be $(0, t/k]$, $(t/k, 2t/k]$, …, $((k\text{-}1)t/k, t]$. Each spike would be assigned to a time bin based on its own spike time, and then the symmetric function would be performed on all spikes from each time bin.

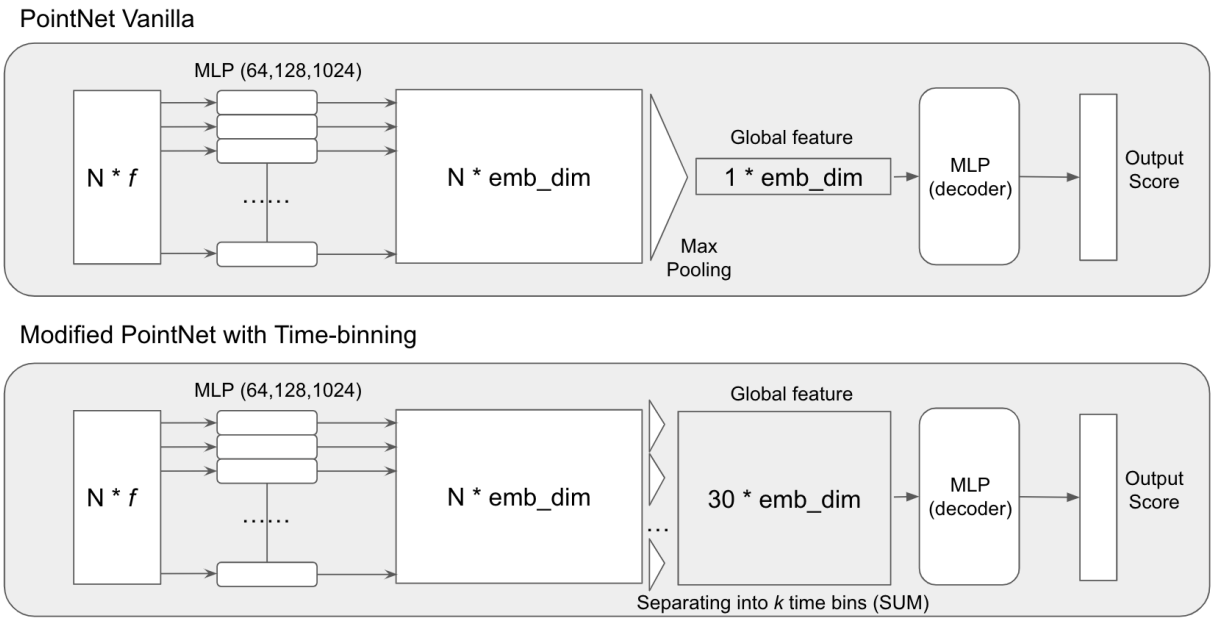

Figure 2. **Architectures of PointNet Vanilla and Modified PointNet with Time-binning.** MLP stands for multi-layer perceptron, and the numbers are the size of each layer. PointNet Vanilla takes N spikes (dimension is 1**f*, where *f* is the number of features) as inputs, applies multi-layer perceptron to transform each input point into a higher dimensional vector, and applies the symmetric function of max pooling. Modified PointNet with Time-binning takes N spikes, applies the same transformation, assigns each spike to a time bin, and applies the symmetric function of summing on each time bin. A second multi-layer perceptron network serves as a decoder to process the global features generated. The outputs are classification scores for *k* classes in both architectures. Batch normalization is applied for all layers, and activation functions are ReLU for all layers.

2.3 PointNet++

One major issue of PointNet is that local spatial structures are not captured. Building upon their original work, Qi et al. introduced PointNet++, a hierarchical neural network that uses the method of applying PointNet recursively on a nested partitioning of the input point set (12). By gradually capturing features on different scales, the network is proved to be able to capture spatial local structures of the data. Figure 3 shows the pipeline for applying PointNet++ directly on Neuralpixel recording data. PointNet++, though capturing spatial local structures, still fails to provide resolution on the scale of time. Here we design an architecture combining time-binning and hierarchical featuring, as shown in Figure 4.

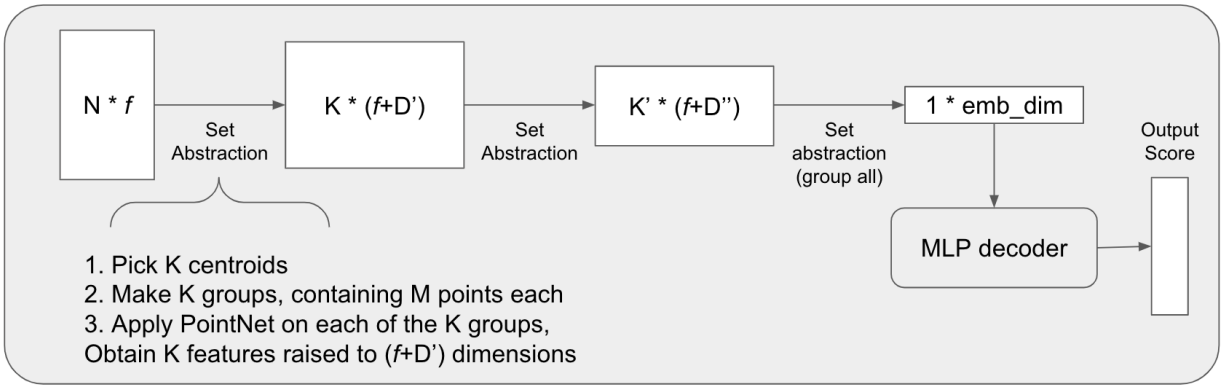

Figure 3. **Illustration of hierarchical feature learning architecture and its application on neural data.** The network takes N spikes as inputs (each spike's dimension is 1**f*, where *f* is the number of features). The inputs go through two Set Abstraction layers, which include the following: a sampling layer first selects K centroids using the furthest point sampling method; a grouping layer makes K groups, which each include one centroid and M points that are closest to this centroid; and a PointNet layer: for each of these groups, a PointNet is applied to each of them to obtain K local features, each raised to (1+D') dimensions. A final PointNet applies the symmetric function to aggregate all local features obtained from the two set abstractions to get one global feature (1*embedding_dimension), which is then decoded through a multi-layer perceptron network to produce the output of classification score.

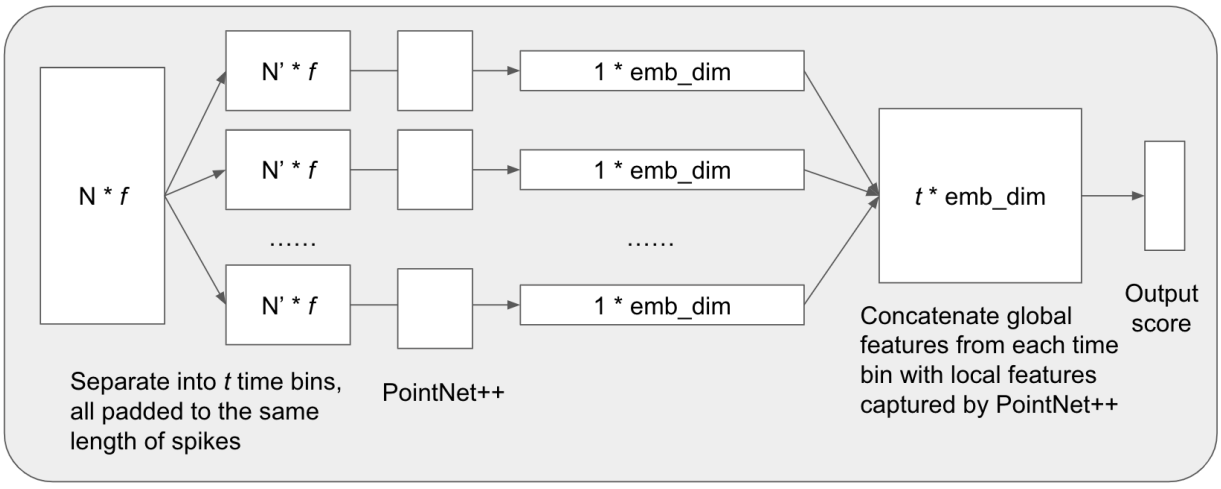

Figure 4. **Illustration of time-binning + hierarchical feature learning architecture.** Inputs are the same as before: *n* spikes, each spike's dimension being 1**f*, where *f* is the number of features. Before any processing, each spike is assigned to one of the *t* time bins based on its spike time, and every time bin is padded to the same length to ensure they all contain the same number of spikes. Each time bin, which can be considered as the representation of the subject's brain state during a short time interval, undergoes PointNet++ processing separately to generate a global feature of that period. This global feature is already capturing the local spatial features. Concatenating *t* global features from the *t* time bins, we obtain a global feature matrix (*t* * embedding_dimension) which we can decode using the multi-layer perceptron network to obtain classification scores.

3. Results

3.1 Experimental data

Electrophysiological and behavior data from subject mouse DY-016 is used for testing the established and the newly-proposed point cloud processing architectures. One session selected from DY-016 included 286 trials, with each trial's duration being 1.5 seconds. During one trial, a visual stimulus is presented to the mouse, and the mouse has to make a choice between left or right. A Neuralpixel recording probe recorded the mouse's electrophysiology data during the entire session. All datasets used in this paper are publicly available on the International Brain Laboratory (IBL) website.

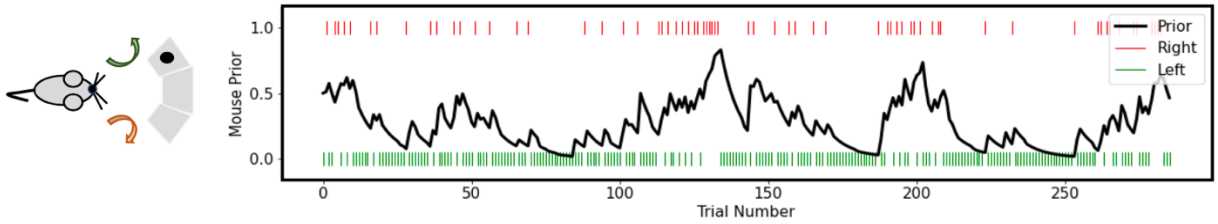

Figure 5. (Left) **The setup of the experiment during which data is collected.** (Right) **The distribution of mouse choices of left and right over the entire session.** Figure from Zhang et al. (6).

3.2 Comparison of performances of different point cloud processing architectures

Decoding performance was evaluated using the unsorted spike train data from the mouse choice task, and results are shown in Figure. Serving as benchmarks for comparison are (1) linear decoding based on Kilosorted data and (2) the clusterless multi-channel thresholding approach using unsorted data. From the graph, it is evident that directly applying PointNet and PointNet++ to neural data resulted in poor performance. Modified PointNet with Time-binning, however, achieved satisfactory performances, coming close to the accuracy and AUC of the benchmarks. This is convincingly showing that, unlike static point clouds from LiDAR scans, time is a crucial dimension in neuronal spike point cloud datasets. Any attempt of decoding behaviors from electrophysiology recording must find a way of efficiently encoding time when generating the feature embedding.

We can also compare two ways of accounting for the dimension of time: directly inputting it (making spike time a feature of each spike), or using time bins (assigning each spike to a time bin and extracting global features from each). While the two approaches achieve similar accuracy scores, the AUC of the time bin approach is significantly higher than the former. Time binning is effectively reducing the temporal resolution. This could mean that too high of a temporal resolution is bringing too much noise into the data – what helps to decode performances might not be the information of which neuron fired at exactly what time, but the patterns of neuron activities in the mouse brain as a whole during different time segments.

It's also worthwhile to examine the training curves. During experiments, when directly applying PointNet to neural data, while the training AUC curve reached around 0.95 after 40 epochs, the validation AUC curve didn't show signs of increasing – this is a sign of overfitting, originating often from not enough training data or that training data doesn't accurately represent all possible input recording data. One possible explanation of this is, while the classification network of PointNet is trained on 12,311 CAD models (each was downsampled to containing 1024 points), our network is only trained on 286 trials from a single session, each trial containing from 2134 – 3578 spikes. This could also be an explanation for the observation that PointNet and PointNet++ didn't show significant differences in performance. If both models are overfitting to training data, it would be difficult to tell which would actually fit the data better in the event of feeding the models enough training data.

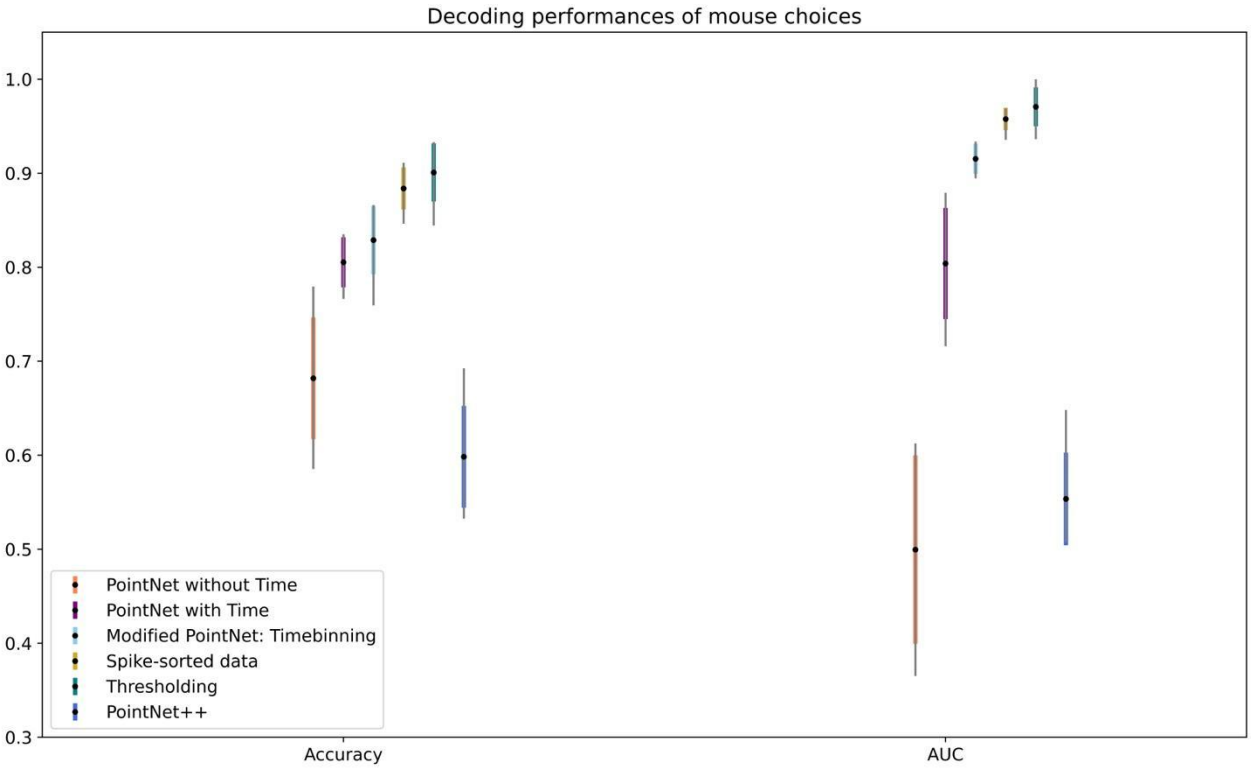

Figure 6. **Comparison of decoding performance on mouse choices using different architectures**. Error bars indicate the minimum, maximum, mean, and standard deviation of accuracy and AUC from 5-fold cross-validation.

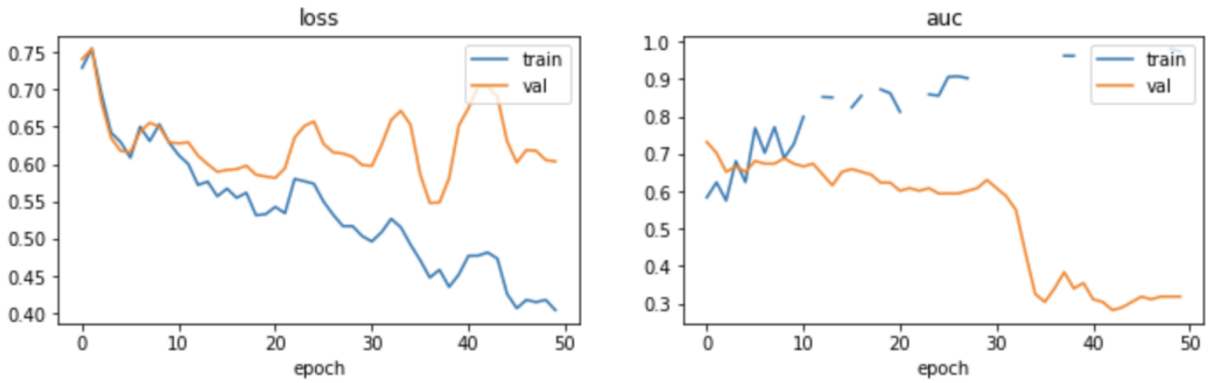

Figure 7. **Loss and AUC curve for training and validation when directly applying PointNet Vanilla on Neuralpixel recording data.** The x-axis is the number of epochs; the y-axis is the Cross-entropy loss (left) and the AUC score (right) of each epoch.

3.3 Comparison of performance of Modified PointNet under different parameters

Knowing that Modified PointNet with Time-binning yields the best decoding performances, we examined whether using different parameters for the method would alter the results. We first explored changing the set of features inputted into the model. Interestingly, no matter whether the set of features is (*x, z, ptp*), (*x, y, z*), (*x, y, z, ptp*), etc., there appear to be no significant differences in the decoding performances (*ptp* is the maximum peak-to-peak amplitude of the spike waveform). Accuracy is around 0.83 and AUC is around 0.91. With features *x*, *y,* and *ptp* removed individually, the model all produced similar performances. The removal of feature *z*, however, seems to impact performance the most, indicating the *z* coordinate of the spike to be the main feature the model relies on for classification. It remains to be investigated as to why adding an additional feature and switching the *x* and *y* coordinate features to the waveform feature have little impact on performance. This could hint at flaws in model design, that the model not fully utilizing all available features given.

Changing the symmetric function from max pooling, as originally proposed in the PointNet paper, to summing, does result in a significant increase in performance. A possible explanation is that, when the symmetric function is applied to reduce dimension in the step of time binning, more information is preserved under the summation function than the max pooling function. If more neurons that are close together actively fired in a time bin compared to another time bin, summation can more accurately reflect that; max pooling, instead, could just simply focus on the spike with the strongest firing pattern in a time bin and ignores the population information.

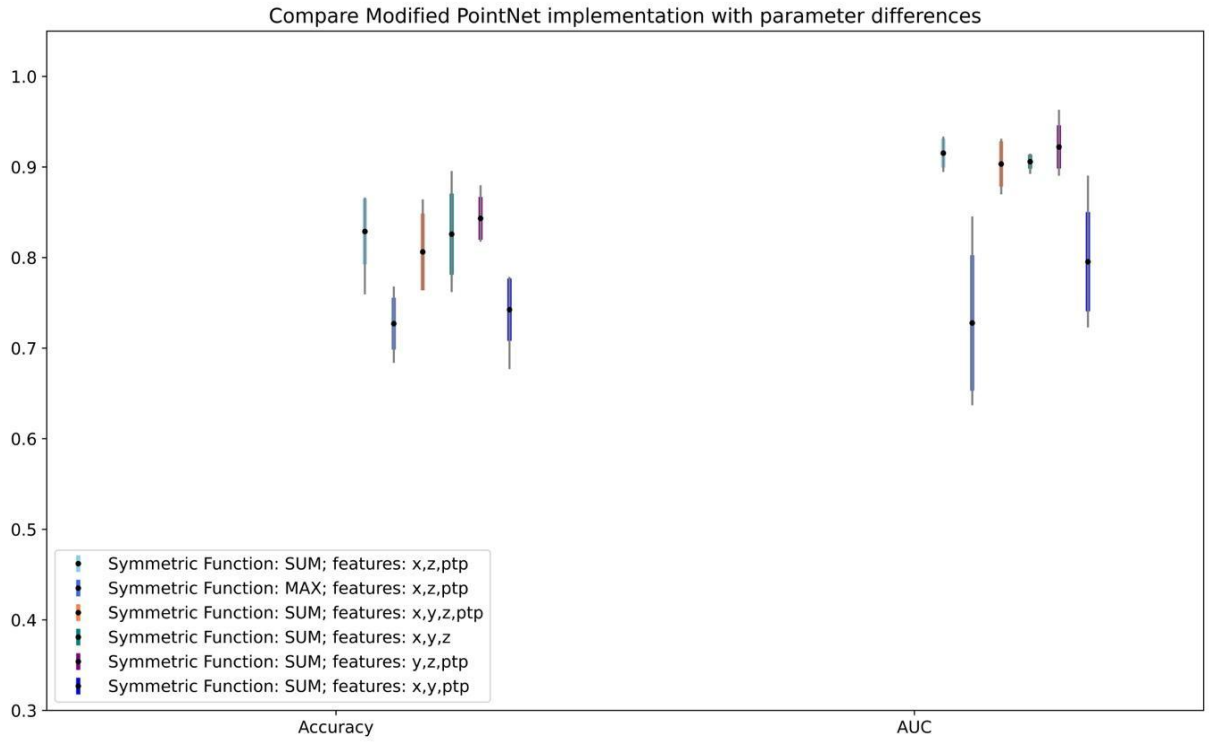

Figure 8. **Comparison of decoding performance on mouse choices of the Modified PointNet with different features and symmetric functions**. As above, error bars indicate the minimum, maximum, mean, and standard deviation of accuracy and AUC from 5-fold cross-validation.

3.4 Future Work

This work covers the preliminary findings in experimenting with the application of point cloud processing deep learning architectures on Neuralpixel data. Aside from trying to gain insights into the experimental results from the above, a key direction for further research is finding a way of extracting local features from the dataset. Initial attempts of directly applying the PointNet++ paradigm have shown poor performances. This raises doubt about whether extracting spatial local features through sampling and grouping is indeed a valid approach. Could the local structures that were captured just be noise around a centroid? As suggested by Dr. Cole Hurwitz, we plan to first test the feasibility of hierarchical feature extraction using clusters approximated by a Gaussian Mixture Model (GMM) (6). The first step of the plan is to perform hard assignments for each spike with the probability scores obtained by the GMM, take the centroid of each cluster, and train a linear classifier on the centroids that are weighted by their relative density. Further work from other angles on repurposing point cloud processing architecture for neural data would also be made, with the focus on figuring out how to best incorporate time information and coping with a smaller training set size.

4. Conclusion

In summary, we examined the feasibility of using point-wise neural networks for clusterless decoding of mouse choices. We directly applied established deep learning networks for point clouds, PointNet and PointNet++, on neural data. We modified the current PointNet architecture to propose a permutation-invariant network that takes into account the temporal feature. Results demonstrate the importance of accounting for the temporal feature in the decoding model, while also suggesting too high of a temporal resolution could adversely impact decoding performances. Modified PointNet with Time-binning helped achieve accuracy on par with the state-of-the-art decoding methods, proving the concept of treating such data as point clouds. This opens the door for a computationally-inexpensive approach to decoding that could capture nonlinearity in Neuralpixel recording data. Future works would center on identifying the best approach for capturing spatial local structures to further increase clusterless decoding performances.

Acknowledgments

The author thanks the supervision and guidance of Yizi Zhang and Prof. Liam Paninski. The author thanks Dr. Cole Hurwitz for providing support and advice on the project. The author also appreciates the presentation feedbacks from fellow classmates of STAT8201.

References

1. M. A. van Gerven, K. Seeliger, U. Güçlü, Y. Güçlütürk, Current advances in neural decoding. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 379-394 (2019).
2. J. I. Glaser *et al.*, Machine learning for neural decoding. *Eneuro* **7** (2020).
3. M. Pachitariu, N. Steinmetz, S. Kadir, M. Carandini, Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *BioRxiv*, 061481 (2016).
4. J. H. Lee *et al.*, YASS: yet another spike sorter. *Advances in neural information processing systems* **30** (2017).
5. A. P. Buccino *et al.*, SpikeInterface, a unified framework for spike sorting. *Elife* **9**, e61834 (2020).
6. Y. Zhang *et al.*, Density-based Neural Decoding using Spike Localization for Neuropixels Recordings. *Not Published* (2022).
7. E. M. Trautmann *et al.*, Accurate estimation of neural population dynamics without spike sorting. *Neuron* **103**, 292-308. e294 (2019).
8. J. Boussard, E. Varol, H. D. Lee, N. Dethe, L. Paninski, Three-dimensional spike localization and improved motion correction for Neuropixels recordings. *Advances in Neural Information Processing Systems* **34**, 22095-22105 (2021).

9. C. Hurwitz, K. Xu, A. Srivastava, A. Buccino, M. Hennig, Scalable spike source localization in extracellular recordings using amortized variational inference. *Advances in Neural Information Processing Systems* **32** (2019).
10. D. Fernandes *et al.*, Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy. *Information Fusion* **68**, 161-191 (2021).
11. C. R. Qi, H. Su, K. Mo, L. J. Guibas (2017) Pointnet: Deep learning on point sets for 3d classification and segmentation. in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 652-660.
12. C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* **30** (2017).