Extended Abstract Track

A minimalistic representation model for head direction system

Editors: List of editors' names

Abstract

We present a minimalistic representation model for the head direction (HD) system, aiming to learn a high-dimensional representation of head direction that captures essential properties of HD cells. Our model is a representation of rotation group $U(1)$, and we study both the fully connected version and convolutional version. We demonstrate the emergence of Gaussian-like tuning profiles and a 2D circle geometry in both versions of the model. We also demonstrate that the learned model is capable of accurate path integration.

Keywords: U(1) rotation symmetry group, Group representation, Recurrent neural network, Path integration

1. Introduction

Spatial navigation is a fundamental cognitive function shared across many species, from insects to humans. A critical component of this navigational system is the perception of direction, which allows animals to maintain a consistent representation of their orientation in the environment. In mammals, this perception of direction is primarily mediated by the head direction (HD) system, a network of neurons that collectively encode the animal's current head orientation relative to its environment (Taube et al., 1990a).

HD cells, discovered in the rat's dorsal presubiculum (Rank, 1984; Taube et al., 1990b), exhibit a remarkable property: they fire maximally when the animal's head faces a specific direction in the horizontal plane, regardless of location or ongoing behavior. Each cell has a preferred direction, with firing rates decreasing as the head turns away, typically following a Gaussian-like tuning curve (Blair et al., 1997). Distributed across interconnected brain regions (Taube, 2007), these cells form a neural "compass" maintaining consistent directional representation (Cullen, 2019). Intriguingly, the HD system maintains direction representation even without external sensory cues – a phenomenon known as path integration (McNaughton et al., 2006). This suggests that the HD system functions as a neural integrator updating based on self-motion cues. Theoretical and computational models have proposed that the HD system functions as a continuous attractor network, where the collective activity of HD cells forms a stable "bump" of activity that can smoothly move to represent different head directions (Zhang, 1996; Skaggs et al., 1994). These models often represent the head direction on a ring, reflecting the circular nature of directional space.

In this paper, we propose a minimalist network model for the HD system to investigate the core component for learning the direction representation, while maintaining sufficient biological plausibility. Our approach is motivated by recent advancements in direction representation learning in high-dimensional spaces (Cueva et al., 2019; Mante et al., 2013; Yang et al., 2019; Maheswaranathan et al., 2019). We leverage the fact that head direction transformations form a representation of the rotation group $U(1)$, acting on a ring of

possible head direction representations. We present two versions of the model: a fully connected version and a convolutional version. Both models aim to learn a high-dimensional representation of head direction that exhibits key properties observed in biological HD systems. We demonstrate that our model can learn Gaussian-like tuning profiles for individual cells and produce a representation that exhibits a clear circle geometry when visualized with principal component analysis (PCA). The learned model is capable of accurate path integration. These emergent properties closely match the characteristics of biological HD systems, providing insights into the computational principles that may underlie their function.

2. Model and Learning

2.1. General Framework

We represent head direction $x \in [0, 2\pi)$ in a continuous d-dimensional vector $v(x) \in \mathbb{R}^d$, which is regarded as responses of putative HD cells and subjects to three constraints:

(1) Transformation rule: $v(x + dx) = F(v(x), dx)$, where F is a function describing changes in the representation $v(x)$ from a change dx in direction. The set of transformations $\{F(\cdot, dx), \forall dx\}$ and the set of representations $\{v(x), \forall x \in [0, 2\pi)\}$ together form a representation of the rotation symmetry group $U(1)$, so that $F(v(x), 0) = v(x)$, and $F(v(x), dx_1 + dx_2) = F(F(v(x), dx_1), dx_2)$. Here the addition in $x + dx$ is mod 2π.

(2) Nonnegativity constraint: $v(x) \geq 0$, reflecting neurons' nonnegative firing rates.

(3) Unit norm constraint: $|v(x)|^2 = \sum_{i=1}^{d} v_i(x)^2 = 1$ corresponds to a constant total activity of neurons regardless of direction x (to be one without loss of generality). This implies the direction x is only represented by spatial patterns of neuronal responses rather than summed responses, which has been widely used in neural coding (Pouget et al., 2003; Dayan and Abbott, 2005).

The transformation rule defines a recurrent neural network $v_t = F(v_{t-1}, dx_t)$ that enables path integration.

2.2. Model for local motion

For local motion dx, the first order Taylor expansion gives us

$$v(x + dx) = F(v(x), dx) = F(v(x), 0) + F'(v(x), 0)dx = v(x) + f(v(x))dx,$$

where $f(v(x)) = F'(v(x), 0)$ is the derivative of $F(v(x), dx)$ with respect to dx evaluated at $dx = 0$. This first-order Taylor expansion corresponds to the Lie algebra of the Lie group formed by the transformations $(F(v(x), dx), \forall dx)$. For larger motion dx, we can also use second-order Taylor expansion.

2.2.1. Fully Connected Version

In the fully connected version, we model local changes in direction as:

$$v(x + dx) = v(x) + Bv(x)dx$$

where $B \in \mathbb{R}^{d \times d}$ is a learnable matrix, and $dx \in [-b, b]$ for a small $b > 0$. This formulation allows for complex interactions between all dimensions of the representation, capturing potential long-range dependencies in the neural code.

2.2.2. Topographical Convolutional Version

In the topographical convolutional version, we place the neurons v_i on a ring, and we model local changes as $v(x + dx) = v(x) + B * v(x)dx$, where B is a learnable convolutional operator, $*$ denotes the 1D convolution operation with periodic boundary condition, and $dx \in [-b, b]$ for a small $b > 0$. The convolutional nature of B is expressed as: $(B * v(x))_i = \sum_{j=-k}^{k} B_j v_{(i+j) \bmod d}(x)$ where B_j are learnable weights of the convolutional kernel, and k is the kernel size.

2.3. Learning Method

Our model learns two sets of parameters:

(1) V: the representations $v(x)$ for all $x \in \{k\frac{2\pi}{n}, k = 0, ..., n-1\}$, where n is the number of grid points. We denote these $v(x)$ collectively as V. For a general continuous x, we express $v(x)$ as a linear interpolation between the two nearest grid points.

(2) B: the update matrix or kernel B.

We define a one-step loss function to train these parameters by minimizing the prediction error of local changes:

$$\mathcal{L}(V, B) = \mathbb{E}_{x,dx}\left[|v(x + dx) - F(v(x), dx)|^2\right]$$

This loss function focuses on the accuracy of single-step updates, eliminating the need for backpropagation through time, which significantly simplifies the learning process and reduces computational complexity.

The above loss function can be minimized by projected gradient descent, i.e., after a gradient descent step or a step of Adam optimizer (Kingma and Ba, 2014), we set all negative elements in each $v(x)$ to 0, and then normalize each $v(x)$ to have norm 1. Expectation $\mathbb{E}_{x,dx}$ can be approximated by uniformly sampling x from $[0, 2\pi)$ and dx from interval $[-b, b]$. The detailed training algorithm and details on linear interpolation can be found in Appendix A.

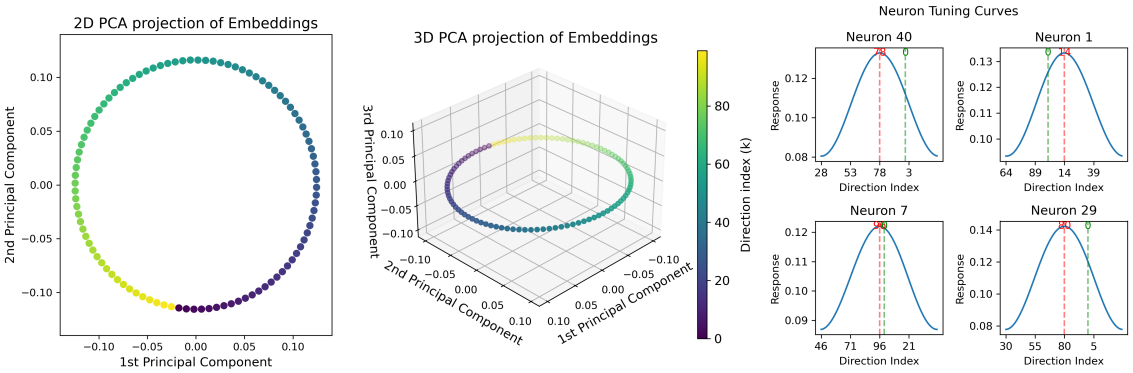
3. Experiments and Results

We conduct a series of experiments to evaluate the performance and properties of our model across various configurations. We explore dimensions $d \in \{10, 20, 50, 100\}$ and local range $b \in \{m\frac{2\pi}{n}, m = 2, 5, 10, 20\}$ for both the fully connected and convolutional versions of the model. Here we fix $n = 100$ in all experiments.

3.1. Ring Structure in PCA Plot

We apply Principal Component Analysis (PCA) to the learned representations $v(x)$ across all directions as in Figure 1(*a*). The PCA plot of the first two principal components reveal a clear ring structure. This emergent property demonstrates that our model has learned a continuous, circular representation of head direction, mirroring both the topology of the actual direction space and the attractor dynamics observed in biological HD systems.

The ring structure is consistent across both model versions, tested dimensions, and local ranges. This result suggests that our high-dimensional representation effectively captures the underlying one-dimensional nature of head direction while providing computational advantages, validating our model's ability to capture essential features of biological head direction systems despite its minimalistic design.



(*a*) PCA (*b*) Tuning curves

Figure 1: (a) 2D and 3D PCA visualization of learned head direction representations. Colors represent the discrete head direction indices from 0 to n, corresponding to angles from 0° to 360°. (b) Tuning curves of a random sample of neurons, which exhibit Gaussian-like tuning profiles. The x-axis represents the full 360° range of head directions, centered on each neuron's preferred direction (red dotted line) to illustrate the circular nature of the representation.The green dotted line marks direction index 0. Full tuning curves can be found in Appendix C.

3.2. Gaussian-like Tuning Profiles

After training, we observe that individual dimensions of the learned representation $v(x)$ exhibit Gaussian-like tuning profiles as in Figure 1(*b*). Each dimension (or "cell") in our model responds maximally to a particular head direction and shows a smooth decrease in activity for directions further from its preferred direction. This behavior closely resembles the tuning curves observed in biological HD cells (e.g., McNaughton et al. (2006)).

3.3. Path integration

We evaluate our model's capability for path integration, a crucial function of biological head direction systems. Path integration involves updating the direction estimate based on a sequence of incremental changes. Despite being trained with a one-step loss function, our model demonstrates remarkable accuracy in multi-step path integration tasks. We test the model's ability to accurately track directional changes over 50 steps and recover the final direction. Our experiments show that the model performs path integration with high accuracy. Detailed procedure and results are provided in Appendix B.

4. Conclusion

We present a minimalistic representation model for the head direction system that captures essential features of biological HD systems while maintaining computational efficiency. Our model demonstrates that key properties of HD cells, such as Gaussian-like tuning and a ring structure, can emerge from a simple learning framework based on representing and updating directions in a high-dimensional space.

References

Hugh T Blair, Brian W Lipscomb, and Patricia E Sharp. Anticipatory time intervals of head-direction cells in the anterior thalamus of the rat: implications for path integration in the head-direction circuit. *Journal of neurophysiology*, 78(1):145–159, 1997.

Christopher J Cueva, Peter Y Wang, Matthew Chin, and Xue-Xin Wei. Emergence of functional and structural properties of the head direction system by optimization of recurrent neural networks. *arXiv preprint arXiv:1912.10189*, 2019.

Kathleen E Cullen. Vestibular processing during natural self-motion: implications for perception and action. *Nature Reviews Neuroscience*, 20(6):346–363, 2019.

Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.

Ruiqi Gao, Jianwen Xie, Xue-Xin Wei, Song-Chun Zhu, and Ying Nian Wu. On path integration of grid cells: isotropic metric, conformal embedding and group representation. *arXiv preprint arXiv:2006.10259*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. *Advances in neural information processing systems*, 32, 2019.

Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474): 78–84, 2013.

Bruce L McNaughton, Francesco P Battaglia, Ole Jensen, Edvard I Moser, and May-Britt Moser. Path integration and the neural basis of the 'cognitive map'. *Nature Reviews Neuroscience*, 7(8):663–678, 2006.

Alexandre Pouget, Peter Dayan, and Richard S Zemel. Inference and computation with population codes. *Annual review of neuroscience*, 26(1):381–410, 2003.

JB Rank. Head-direction cells in the deep layers of dorsal presubiculum of freely moving rats. In *Soc. Neuroscience Abstr.*, volume 10, page 599, 1984.

William Skaggs, James Knierim, Hemant Kudrimoti, and Bruce McNaughton. A model of the neural basis of the rat's sense of direction. *Advances in neural information processing systems*, 7, 1994.

Jeffrey S Taube. The head direction signal: origins and sensory-motor integration. *Annual Review of Neuroscience*, 30(1):181–207, 2007.

Jeffrey S Taube, Robert U Muller, and James B Ranck. Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis. *Journal of Neuroscience*, 10(2):420–435, 1990a.

Jeffrey S Taube, Robert U Muller, and James B Ranck. Head-direction cells recorded from the postsubiculum in freely moving rats. II. Effects of environmental manipulations. *Journal of Neuroscience*, 10(2):436–447, 1990b.

Dehong Xu, Ruiqi Gao, Wen-Hao Zhang, Xue-Xin Wei, and Ying Nian Wu. Conformal isometry of lie group representation in recurrent network of grid cells. In Sophia Sanborn, Christian Shewmake, Simone Azeglio, Arianna Di Bernardo, and Nina Miolane, editors, *Proceedings of the 1st NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, volume 197 of *Proceedings of Machine Learning Research*, pages 370–387. PMLR, 2023.

Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

Kechen Zhang. Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory. *Journal of Neuroscience*, 16(6):2112–2126, 1996.

Appendix A. Training Details

A.1. Learning Algorithm

We use Adam optimizer(Kingma and Ba, 2014) to minimize the loss function. The algorithm proceeds as follows:

Algorithm 1: Learning Head Direction Representation

Input: Number of directions n, dimension d, learning rate η, number of iterations T
Output: Learned representations $v(x_k)$ and transition function F
Initialize $v(x_k)$ for $x_k = k\frac{2\pi}{n}, k = 0, 1, \ldots, n-1$;
Initialize B (matrix or convolutional kernel);
for $t \leftarrow 1$ **to** T **do**
  Sample a batch of (x, dx) pairs;
  Compute the loss $\mathcal{L}$ for the batch;
  Update $v(x_k)$ and B using gradients of $\mathcal{L}$ and learning rate η;
  for $k \leftarrow 0$ **to** $n-1$ **do**
    $v(x_k) \leftarrow \max(v(x_k), 0)$; `// Enforce non-negativity`
    $v(x_k) \leftarrow \text{project}(v(x_k))$; `// Project onto unit sphere`
  end
end
return $v(x)$ and B

A.2. Continuous Representation and Linear Interpolation

To achieve a continuous representation, we define $v(x)$ at discrete points $x_k = k\frac{2\pi}{n}$ for $k = 0, 1, \ldots, n-1$, and use linear interpolation for intermediate values:

$$v(x) = (1 - w)v(x_{\lfloor k \rfloor}) + wv(x_{\lceil k \rceil})$$

where $k = \frac{n}{2\pi}x$, $w = k - \lfloor k \rfloor$, and $\lfloor \cdot \rfloor$, $\lceil \cdot \rceil$ denote floor and ceiling functions respectively.

A.3. Second-Order Fully Connected Version

For larger local motion range b (specifically, $b = 20\frac{2\pi}{n}$ in our experiments), we employ a second-order model to capture higher-order dynamics:

$$v(x + dx) = v(x) + Bv(x)dx + Cv(x)d^2x$$

where $C \in \mathbb{R}^{d \times d}$ is another learnable matrix. This second-order term allows the model to better account for changes over larger directional steps.

A.4. Model parameters

In our training process, we use $n = 100$ discrete directions. The model was trained for 200,000 epochs with a batch size of 256, using an Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 4e-5. A learning rate scheduler (ReduceLROnPlateau) is employed with a factor of 0.8 and patience of 5000 epochs to adapt the learning rate during training. For the convolutional model, we use a kernel size of 3. An example training loss curve can be found in Figure 2.

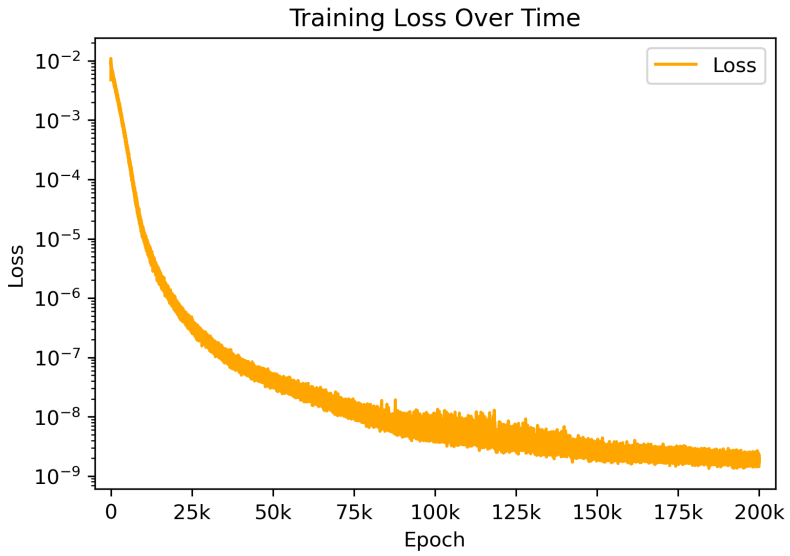


Figure 2: Training loss

Appendix B. Path Integration

Path integration is the process of updating a direction estimate based on a sequence of incremental changes. In the context of our direction representation model, we use path

integration to track changes in direction over time and recover the final direction (Gao et al., 2021; Xu et al., 2023).

B.1. Experiment procedure

Let $x_0 \in [0, 2\pi)$ be the initial direction, and $(dx_1, dx_2, \ldots, dx_n)$ be a sequence of directional shifts. Given our direction representation function $v(x) \in \mathbb{R}^d$ and the update function $F(v, dx)$, we track the changes in the direction representation as follows: $v_0 = v(x_0)$, $v_t = F(v_{t-1}, dx_t)$, $t = 1, ..., n$, where v_t represents the direction representation after t steps.

B.2. Recovering the Final Direction

After obtaining the final representation v_n, we recover the corresponding direction x_n by maximizing the inner product between v_n and $v(x)$ over all possible x:

$$x_n = \arg \max_{x \in [0,2\pi)} \langle v_n, v(x) \rangle$$

This maximization leverages the property that $v(x)$ should be most similar to v_n when x is close to the true final direction.

B.3. Evaluation Metrics

Path integration is evaluated using two local range scenarios: $b = \frac{2\pi}{n}$ radians, and $b = m\frac{2\pi}{n}$ radians, where m is the multiple of the basic angular step size used during training. In both cases, each step's motion dx is sampled uniformly from the range $[-b, b]$. The model estimates the direction after each step, with and without re-encoding. For path integration with re-encoding, we first decode $v \to \hat{x}$ to the 1D head direction angle via $\hat{x} = \arg\max_{x' \in [0,2\pi)} \langle v, v(x') \rangle$, and then encode $v \leftarrow v(\hat{x})$ back to the neuron space intermittently. Since our model is trained in a 1-step manner, this approach aids in rectifying the errors accumulated in the neural space throughout the transformation. Errors are calculated as the average angular difference between the true and estimated directions over all steps in a sequence, measured in radians. The "local range = $\frac{2\pi}{n}$ error" column represents results for the unit angular step range $\frac{2\pi}{n}$, while the "local range = $m\frac{2\pi}{n}$ error" column shows results using the larger range the model was trained with. The table presents these errors for different model architectures (varying in dimension d and training range multiple m), comparing performance with and without re-encoding for both scenarios.

Appendix C. Additional Results

In Figure 1(*b*), we sampled 4 neurons to show the tuning curves. Here we attach the full tuning curves with $d = 100$. We observe that all neurons representations in $v(x)$ exhibit Gaussian-like tuning profiles.

Table 1: Path integration results. m represents the multiple of the unit angular step size ($\frac{2\pi}{n}$) used for training and evaluation in the larger range scenario. Errors are reported as the average angular difference in radians over all steps in a sequence. Results are shown for two local motion range scenarios, $b = \frac{2\pi}{n}$ and $b = m\frac{2\pi}{n}$. In both settings, location motions dx are uniformly sampled from the range $[-b, b]$.

Architecture	d	m	local range $=\frac{2\pi}{n}$ error (radians)		local range $= m\frac{2\pi}{n}$ error (radians)	
			without re-encoding	with re-encoding	without re-encoding	with re-encoding
Fully-connected	100	2	0.000	0.000	0.069	0.000
	100	5	0.007	0.000	0.765	0.010
	100	10	0.044	0.000	1.069	0.232
	100	20	0.356	0.000	1.362	0.179
	50	2	0.000	0.000	0.028	0.000
	50	5	0.000	0.000	0.036	0.000
	50	10	0.008	0.000	0.872	0.167
	50	20	0.363	0.000	0.149	0.182
	20	2	0.000	0.000	0.008	0.000
	20	5	0.001	0.000	0.024	0.009
	20	10	0.079	0.000	0.188	0.153
	20	20	0.385	0.000	0.186	0.154
	10	2	0.196	0.000	0.044	0.000
	10	5	0.000	0.000	0.047	0.014
	10	10	0.000	0.000	1.287	0.563
	10	20	0.000	0.000	0.897	0.338
Convolutional	100	2	0.000	0.000	0.110	0.000
	100	5	0.002	0.000	0.832	0.000
	100	10	0.056	0.000	1.252	0.184
	100	20	0.035	0.000	1.322	0.107
	50	2	0.000	0.000	0.070	0.000
	50	5	0.001	0.000	0.293	0.000
	50	10	0.008	0.000	0.866	0.167
	50	20	0.360	0.000	1.239	0.180
	20	2	0.000	0.000	0.088	0.000
	20	5	0.000	0.000	0.239	0.000
	20	10	0.105	0.000	0.883	0.287
	20	20	0.035	0.000	0.144	0.180
	10	2	0.000	0.000	0.065	0.000
	10	5	0.002	0.000	0.125	0.014
	10	10	0.005	0.000	0.323	0.028
	10	20	0.003	0.000	0.225	0.096

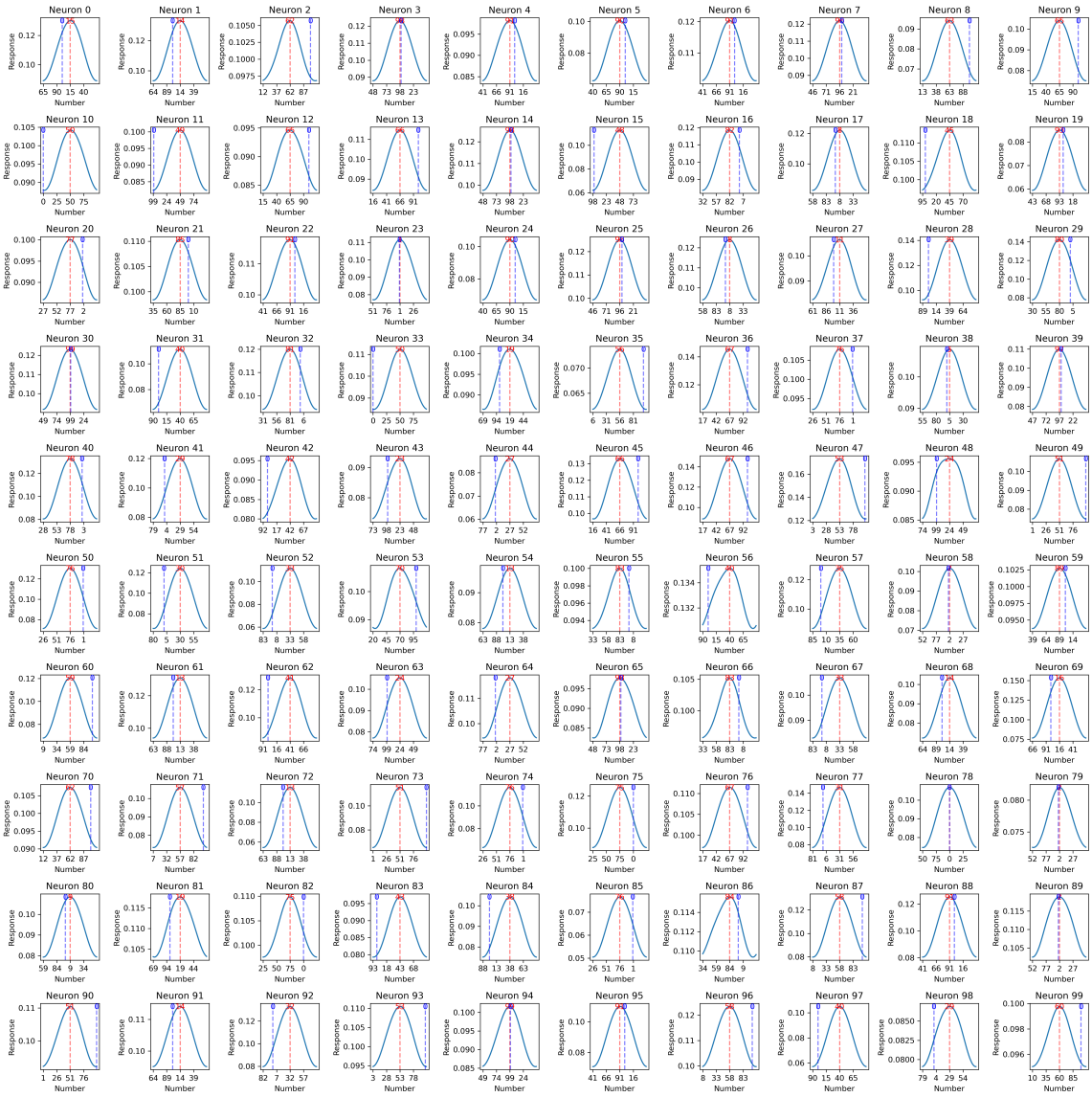


Figure 3: Full tuning curves with $d = 100$