UNIVERSITY OF WATERLOO

# Spatial Semantic Pointers (SSPs)

Chris Eliasmith
SYDE 556/750

CTN Waterloo

# Spatial Semantic Pointers

- Semantic pointers represent standard discrete structures (lists, trees, etc.)
- SSPs allow recurrent convolutions to have fractional powers

$$B^k = \underbrace{B \circledast B \circledast \ldots \circledast B}_{B \text{ appears } k \text{ times}}$$

- Compute fractional *k* in Fourier space

$$B^k = \mathcal{F}^{-1}\left\{\mathcal{F}\{B\}^k\right\}, \quad k \in \mathbb{R}$$

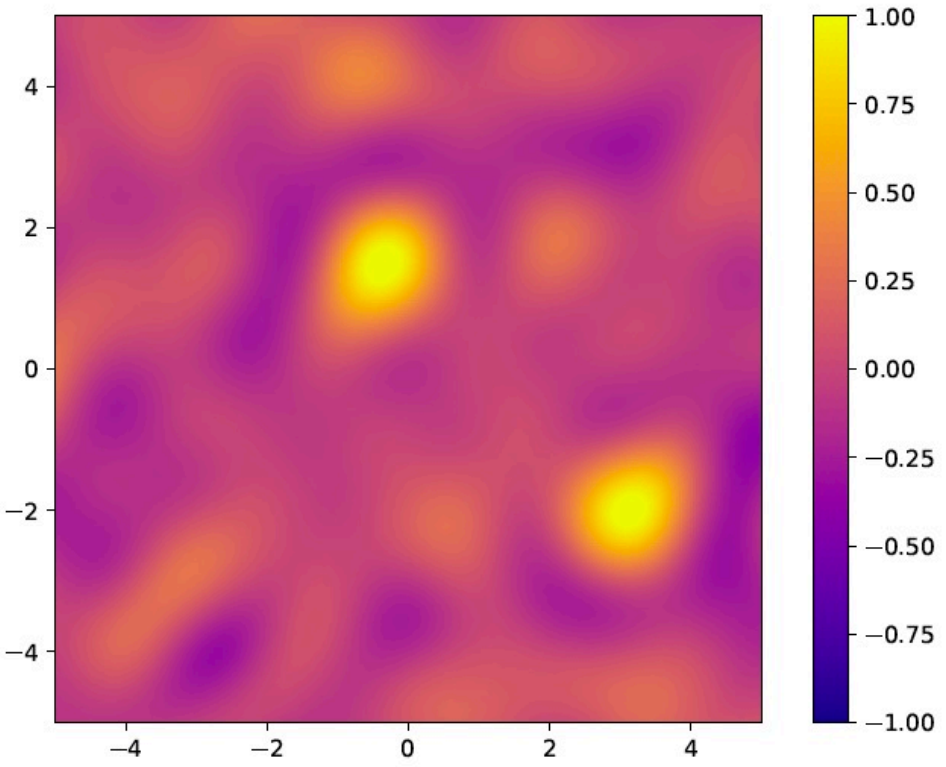
$$S(x,y) = X^x \circledast Y^y = \mathcal{F}^{-1}\{\mathcal{F}\{X\}^x \odot \mathcal{F}\{Y\}^y\}$$

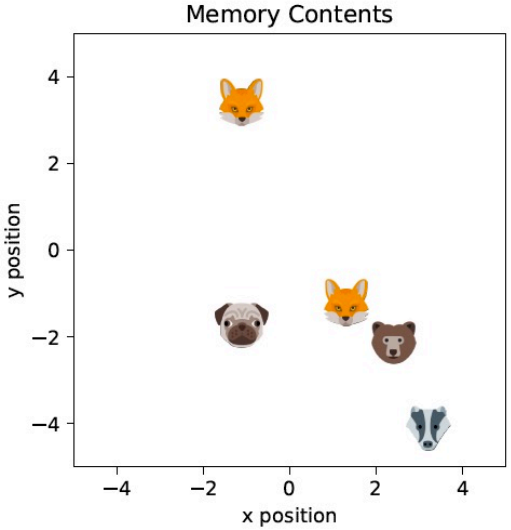
# Spatial Semantic Pointers

- Represent continuous space (Clifford torus)

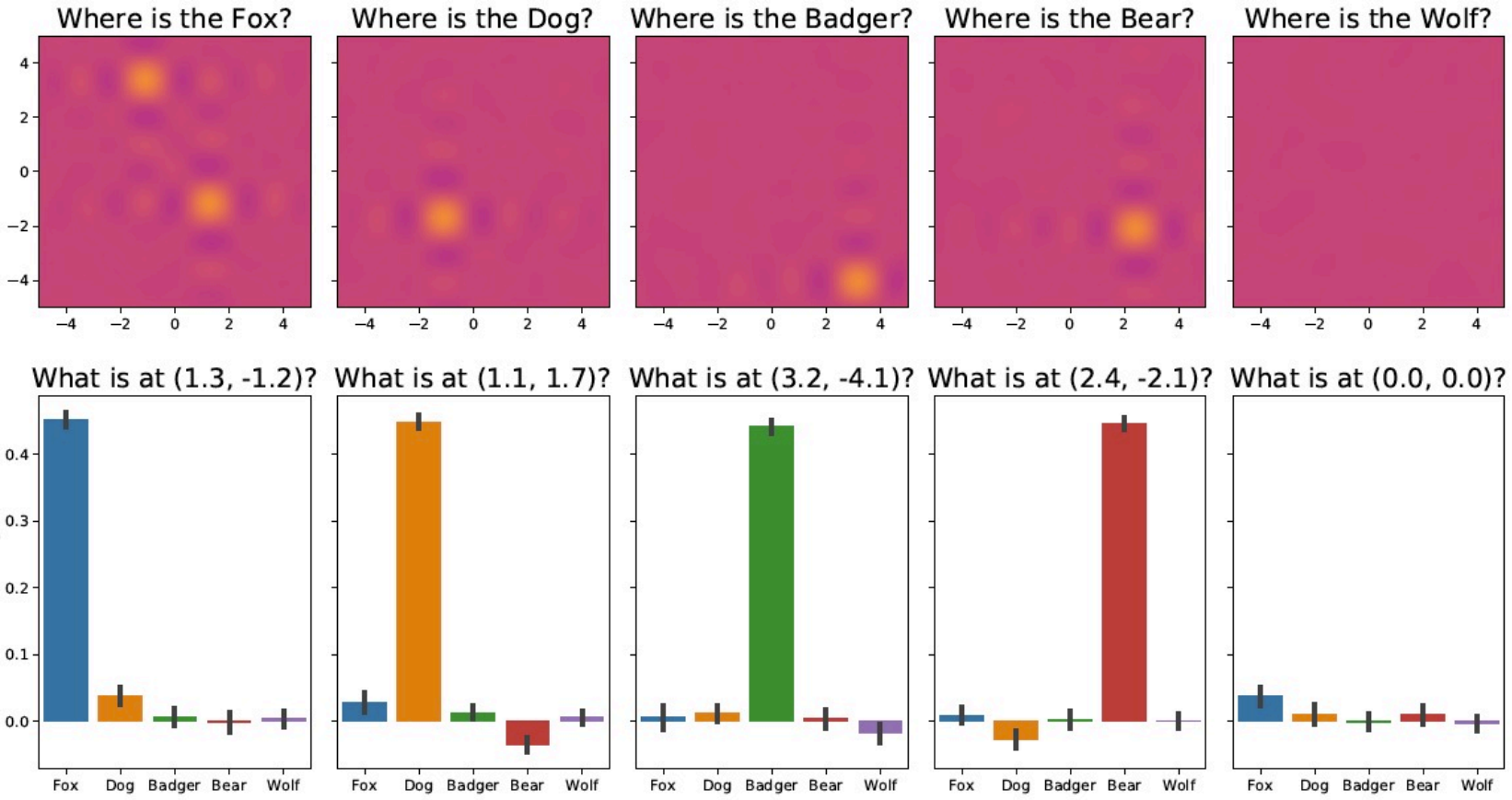
$$S(x,y) = X^x \circledast Y^y$$

- Heat map to visualize vector contents
- Dot product between SSP at every possible position and S






$$M = \sum_{i=1}^{m} OBJ_i \circledast S_i$$



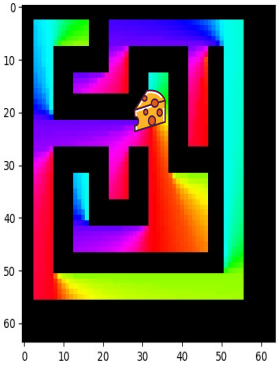
# Spatial Manipulations

| Desiderata | Accuracy | |
|---|---|---|
| | Non-Neural | Neural |
| Query single object | 99.1% | 95.7% |
| Query missing object | 99.4% | 96.7% |
| Query location | 97.3% | 94.7% |
| Query duplicate object | 97.4% | 95.3% |
| Query Region | 90.4% | 73.5% |
| Slide single object in group (all objects) | 75.7% | 67.3% |
| Slide single object in group (moved object) | 100.0% | 100.0% |
| Slide whole group | 97.8% | 96.7% |
| Readout x-y location from SSP | 95.7% | 94.1% |
| Construct SSP from x-y location | 100.0% | 99.0% |

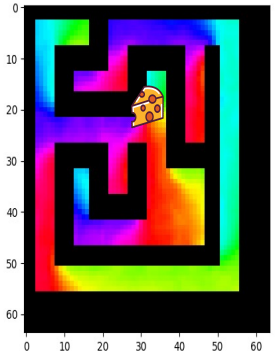
# Navigating to Goal

Single layer MLP, same for each encoding (tried many more).

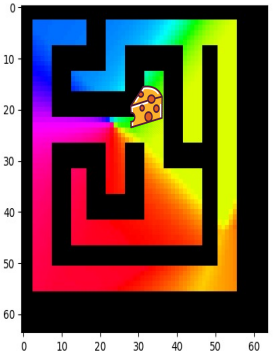
Ground Truth
RMSE: 0



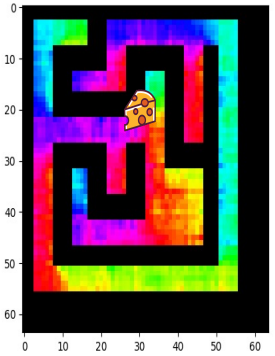
SSPs
RMSE: 0.0529



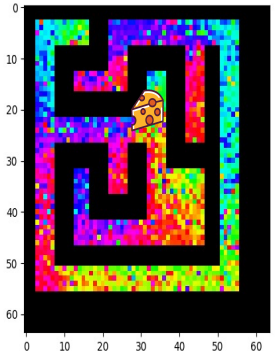
Random Projection
RMSE: 0.5351



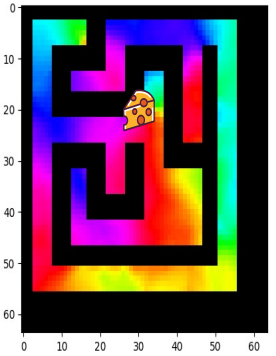
Trig Functions
RMSE: 0.1350



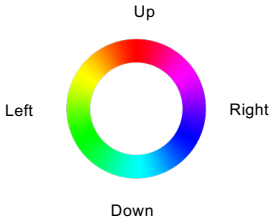
Random Mapping
RMSE: 0.2580



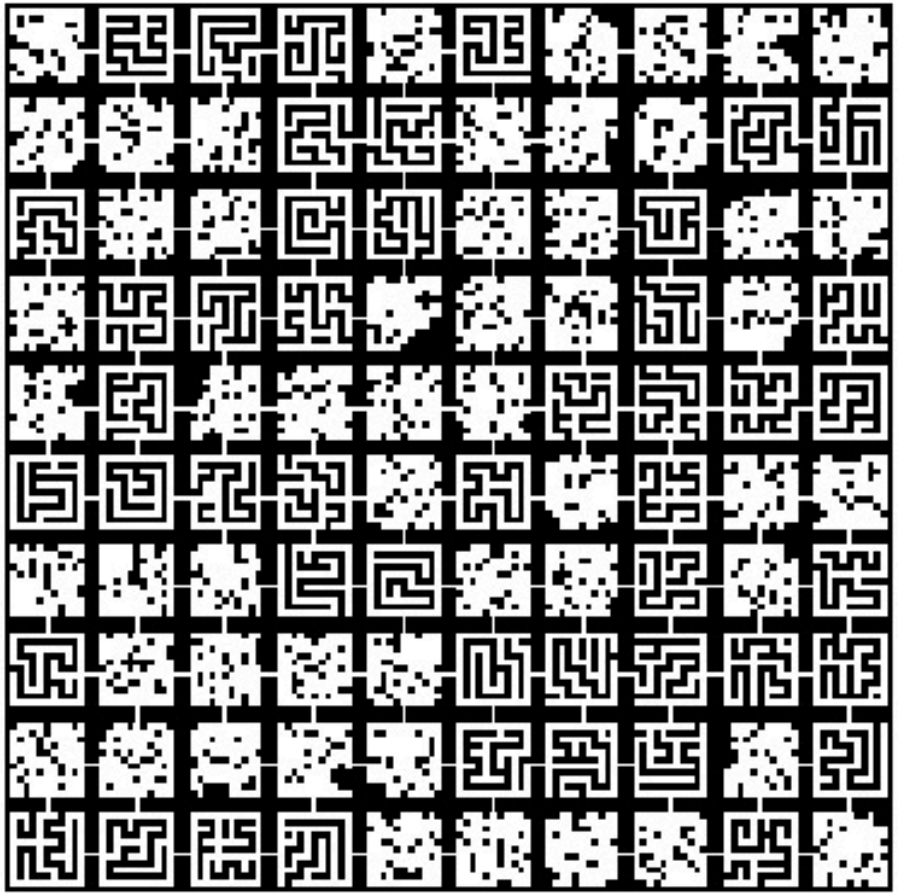
2D Coordinates
RMSE 0.1984

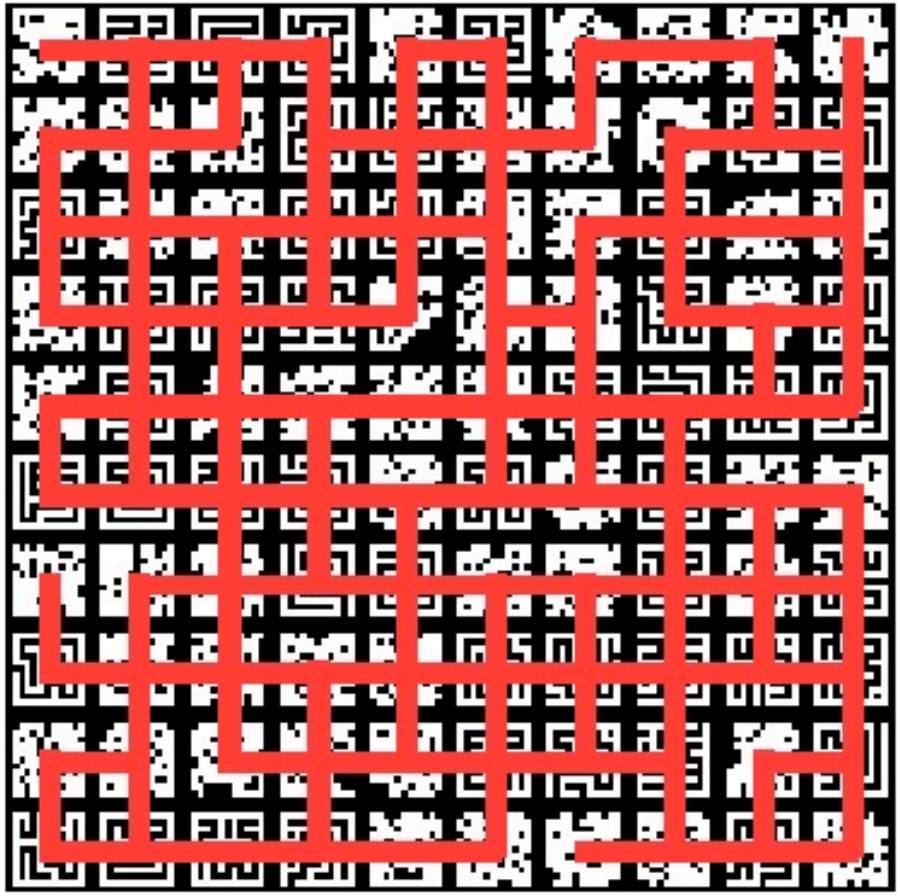


Legend



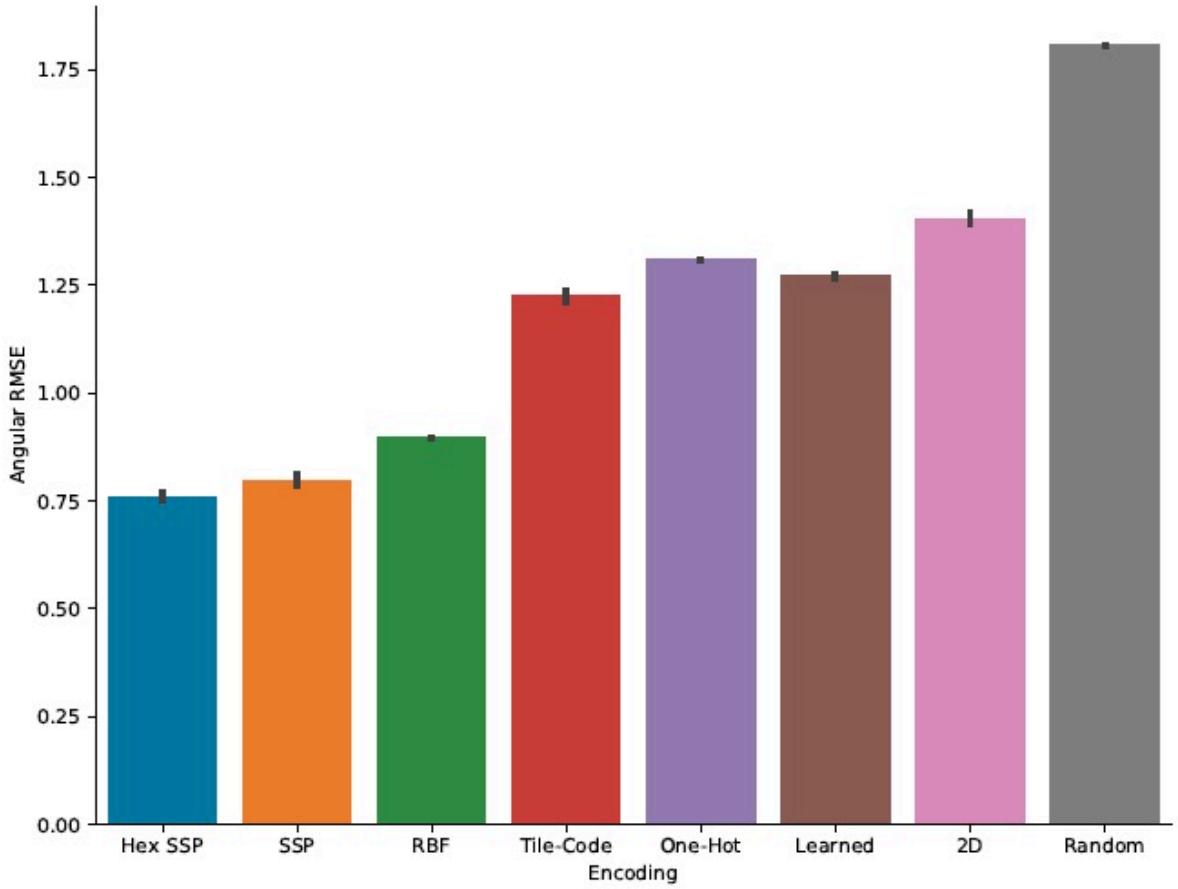
# Scaling





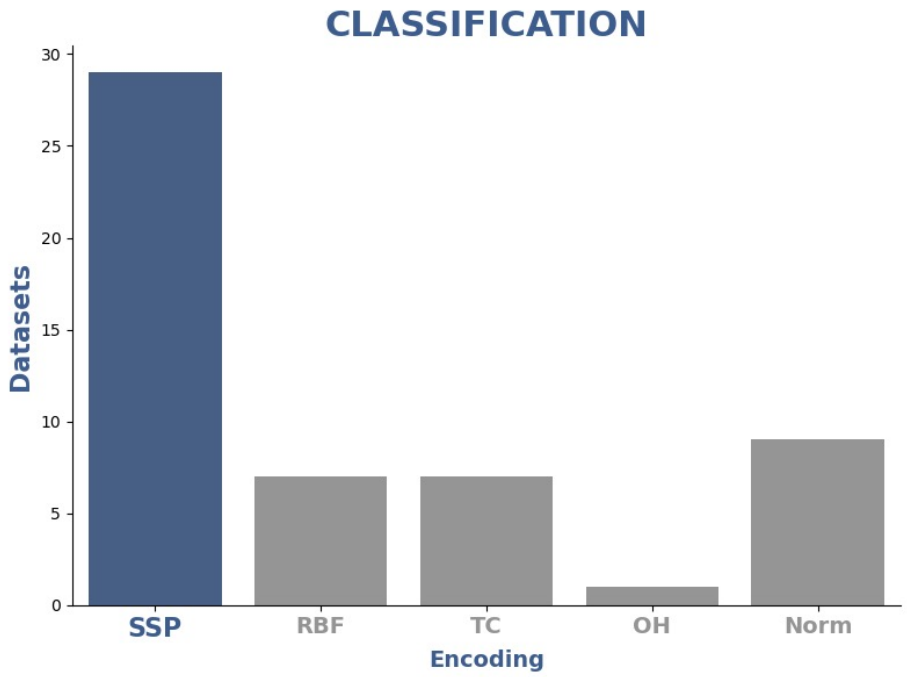
Example 10x10 joined, hierarchical environment

# Scaling




Performance of different encoders on 10x10 large maze from any point to any other

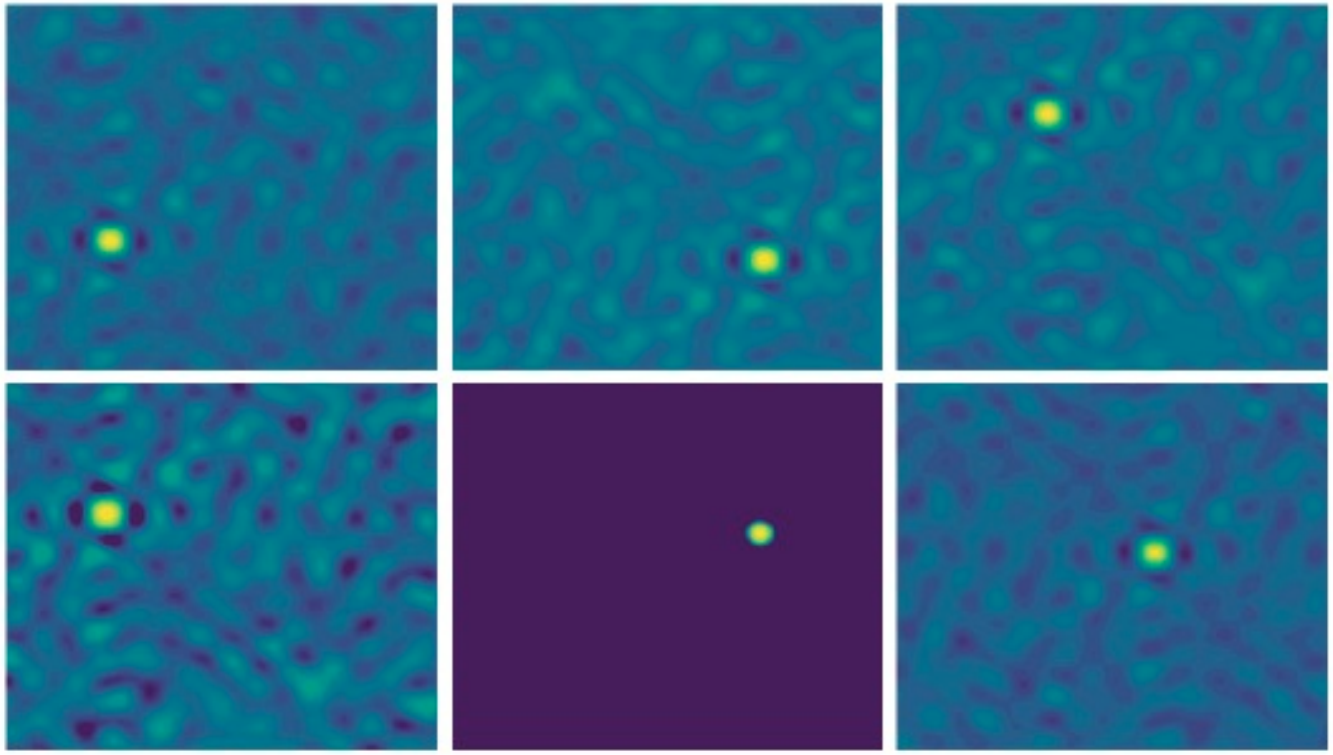
# Good for General ML


SSPs are more accurate on a large majority of 122 standard ML benchmarks.

# How to Choose Axis Vectors

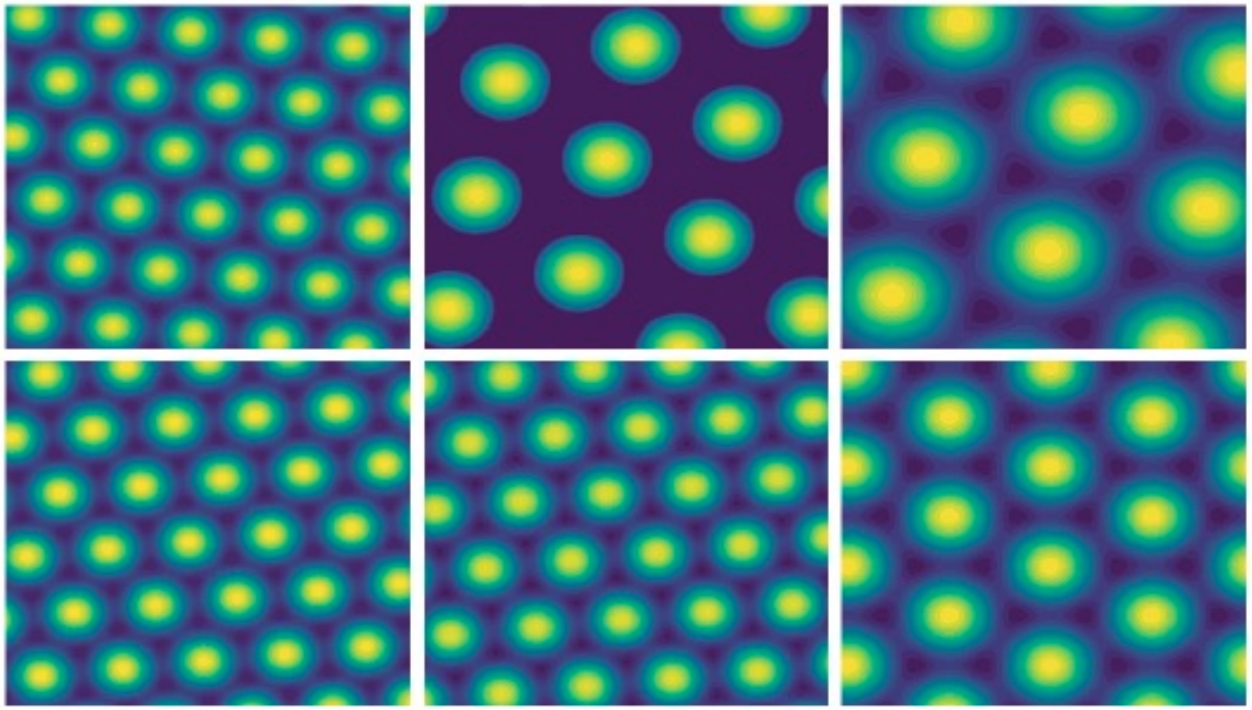
- Randomly (examples til now)



Tuning curves of neurons with random axis vectors and evenly tiled SSPs as encoders

# How to Choose Axis Vectors

- With plane wave structure

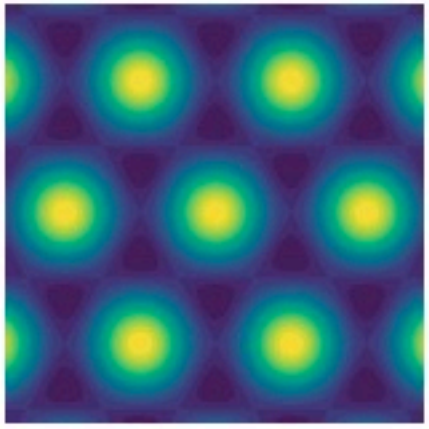
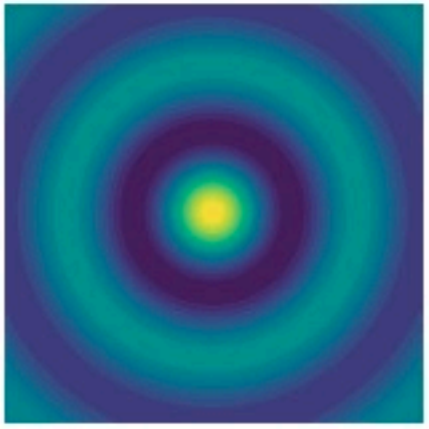
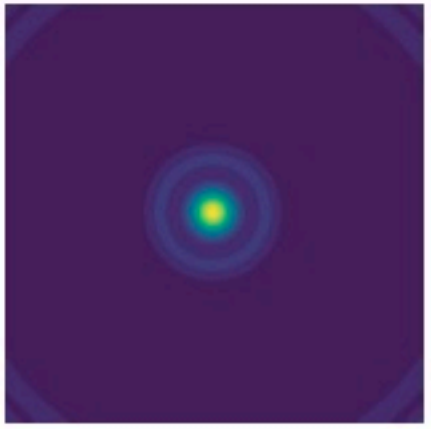


Tuning curves of neurons with structured axis vectors and encoders picking out plane waves

# How to Choose Axis Vectors

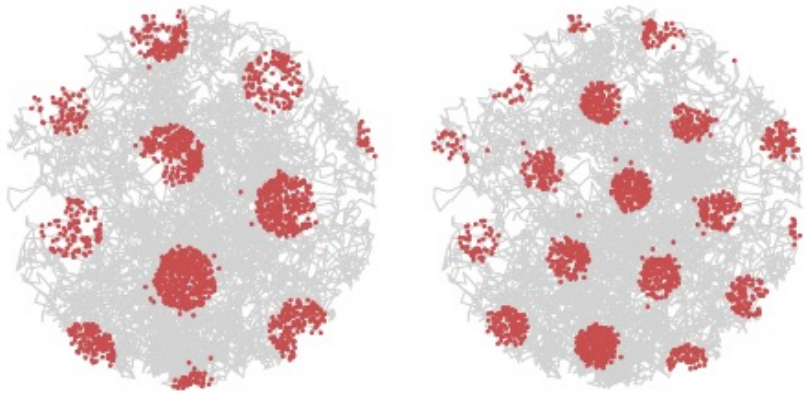
- Sums of planar waves



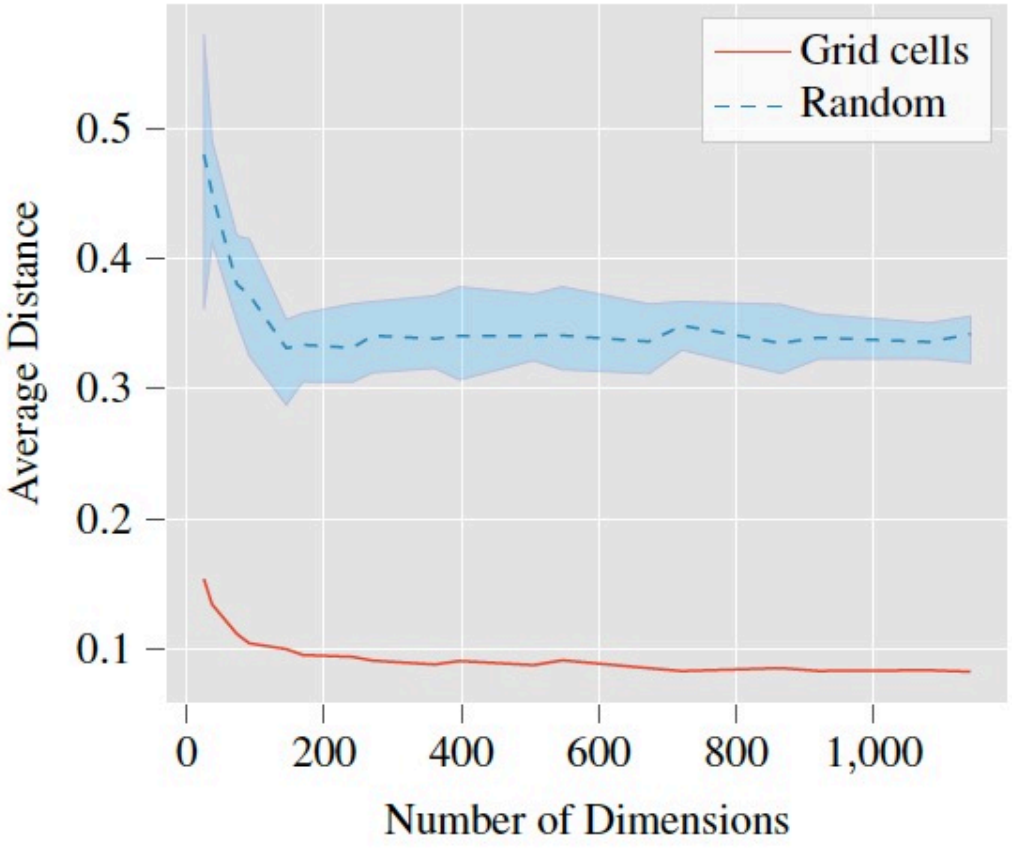
# Grid Cells

- With plane wave structure, spiking neurons give grid cell responses
- We can combine them to get place cells (with standard NEF decoders)



SSP Grid cells

# Place Cells


Grid cells
Random
Average Distance
0.5
0.4
0.3
0.2
0.1
0
200
400
600
800
1,000
Number of Dimensions

# Mathematical properties

- Euclidean space is preserved on a high-d torus

$$S(x_1, y_1) \circledast S(x_2, y_2) = S(x_1 + x_2, y_1 + y_2)$$

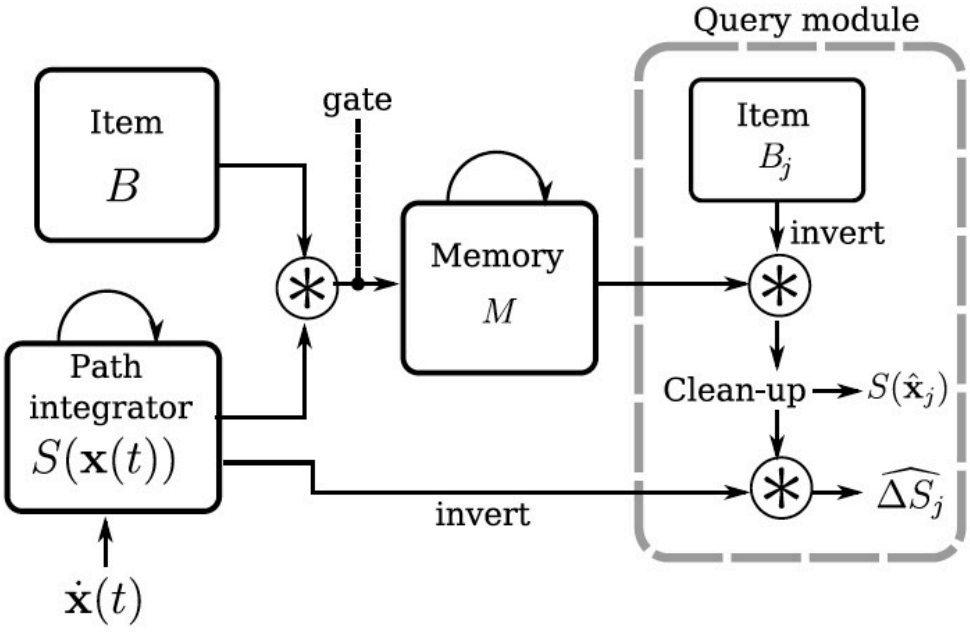
- E.g., $S(x, y) \circledast S(\Delta x, \Delta y) = X^{x+\Delta x} \circledast Y^{y+\Delta y}$

- We get the benefits of high-d representation, with accurate low-d Euclidean representation

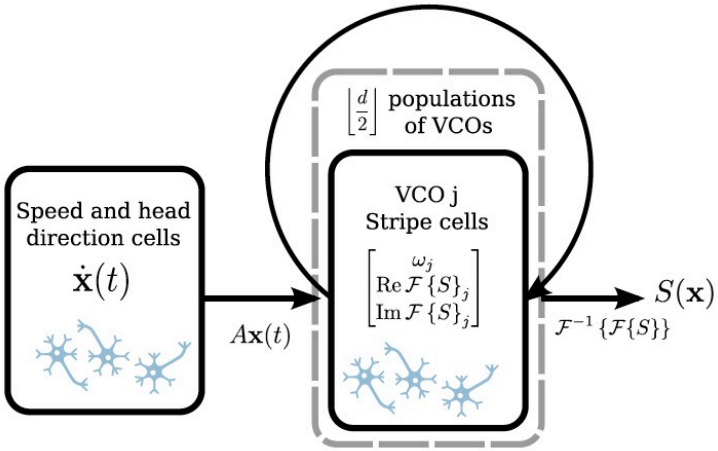
# Cognitive SLAM

- SLAM with complex features at certain spatial locations
- Start with no knowledge, bind vector descriptions to particular location in space
- Combines spatial and ‘symbol’ repn in neural network
- Semantic map as opposed to standard ‘image registration’ map

# Cognitive SLAM Model




Full Model

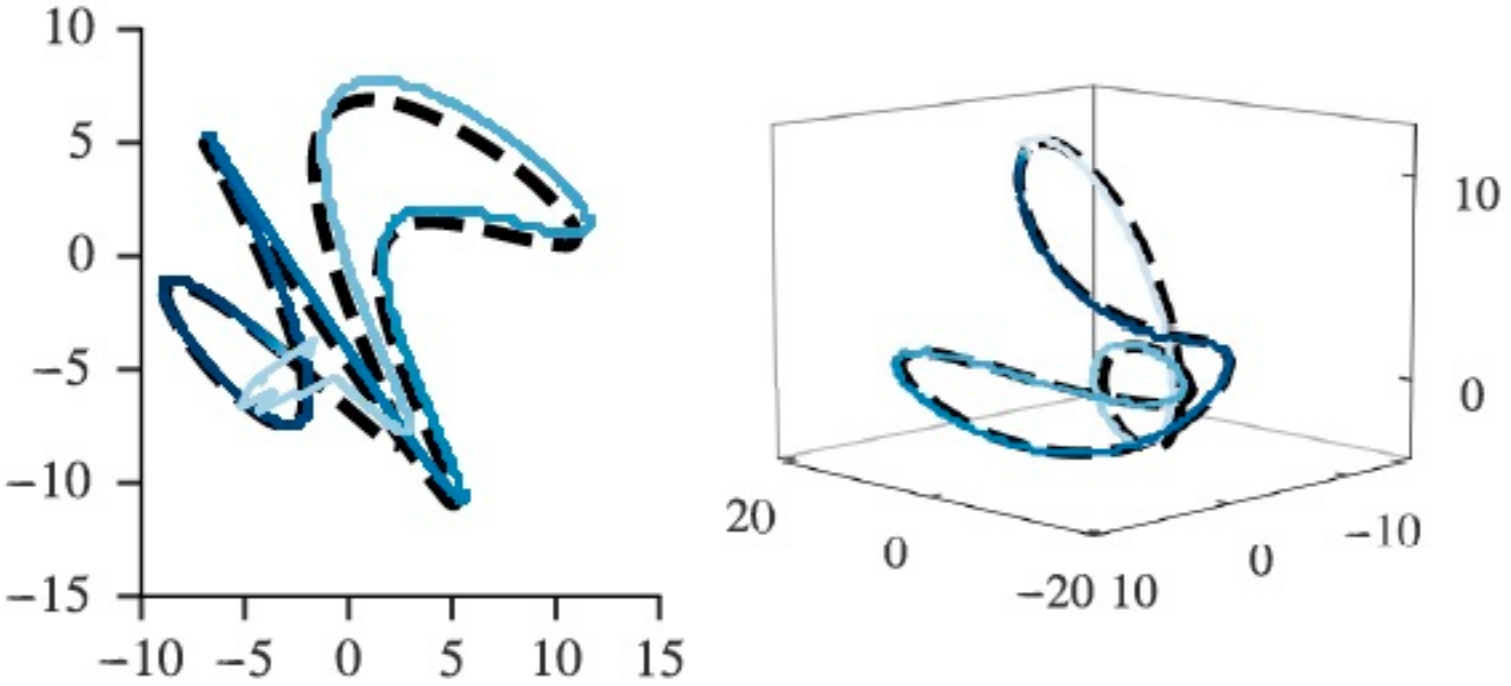



Path Integrator

- Path integrator tracks ego position from velocity
- SLAM model learns env map, outputs allo- and ego-centric position

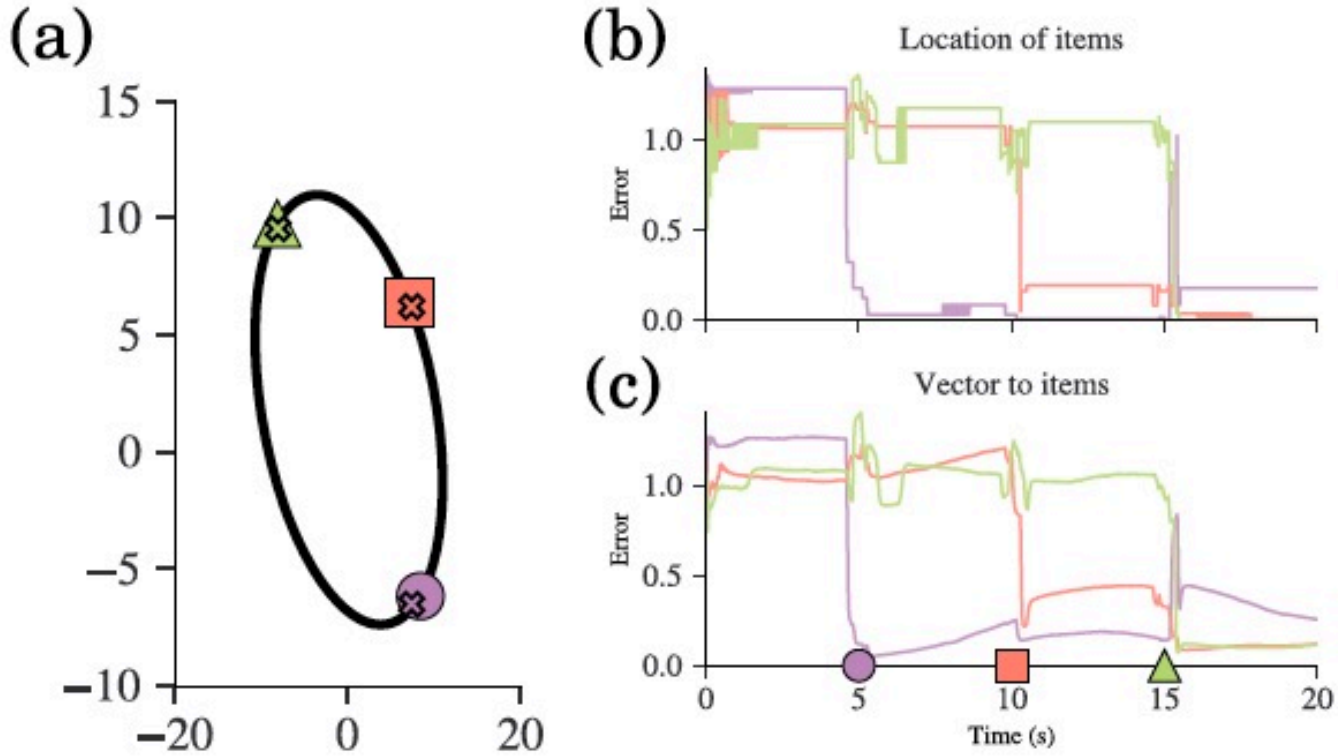
# Path Integration

- One-minute long paths in 2D and 3D, spiking network



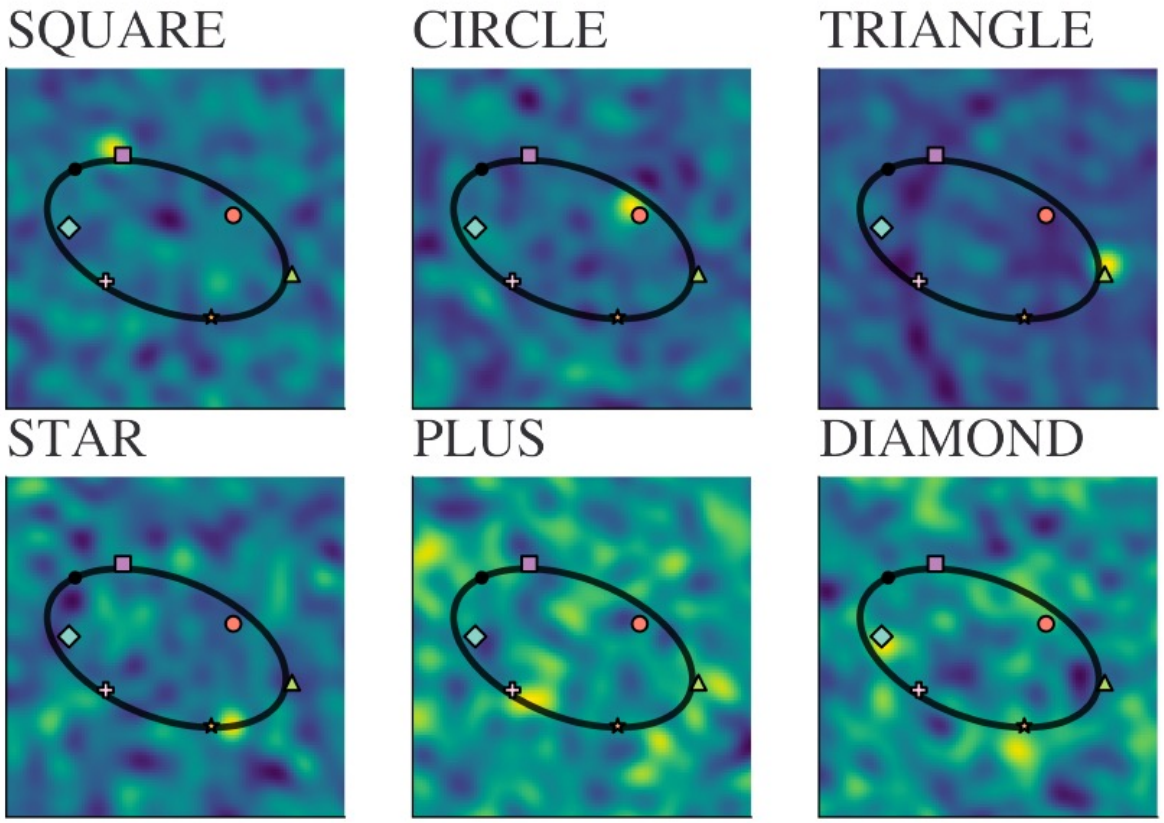
# Cognitive SLAM

- Learns env by end of 20s path, full spiking
- Scaling up; symbols ‘over’ space



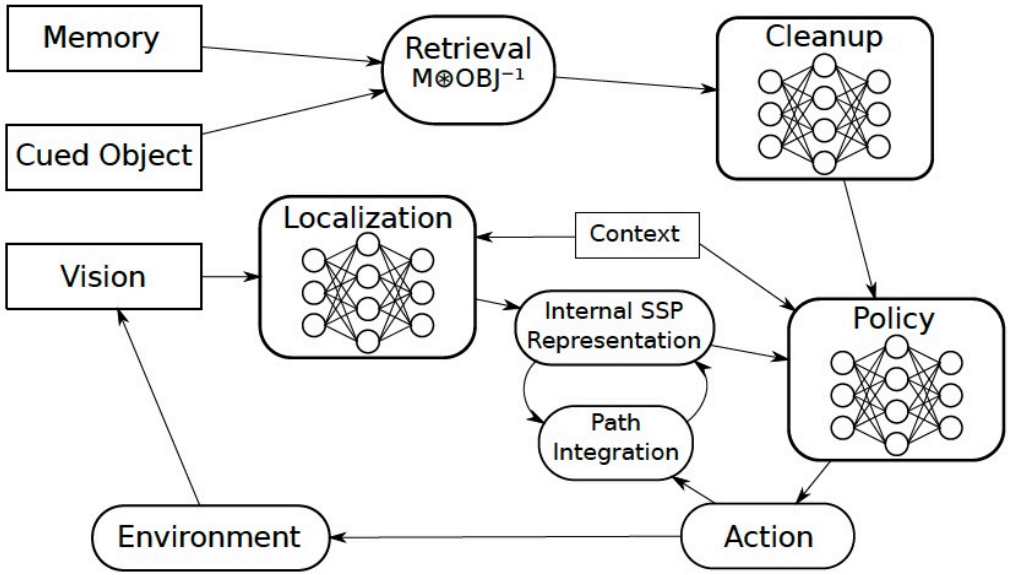
# Cognitive SLAM

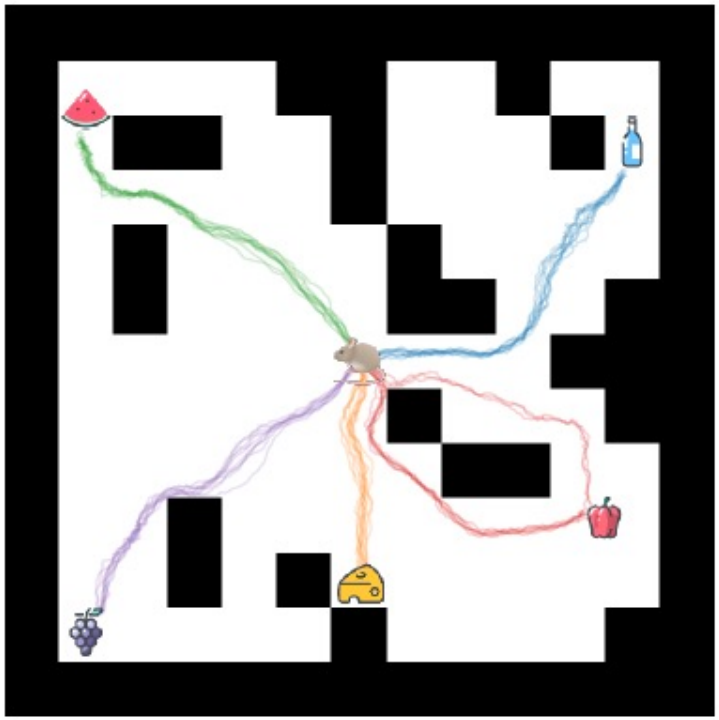
- Learns maps in LTM bound to position



# Navigation network

Combined all of the above into a network to recall location and navigate to arbitrary objects in a maze.



# SSPs as Probabilities

- SSPs can be a method for encoding and processing probabilities
- Method directly connects neural networks to probabilistic reasoning
- SSP based methods are efficient

# SSPs as Probabilities

*Background*

- Kernel Density Estimators

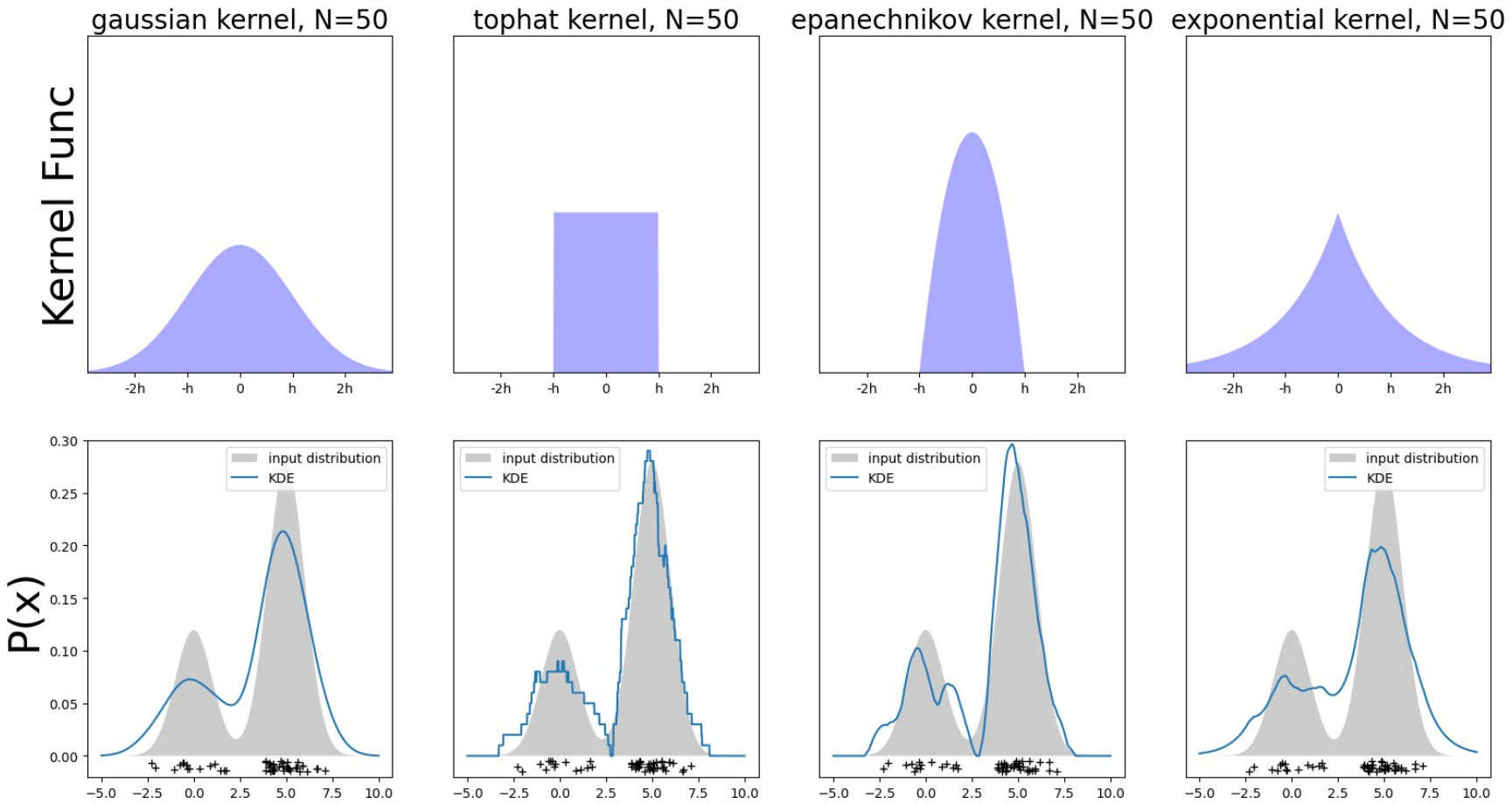
From a dataset $\mathcal{D} = (x_1, x_2, \ldots, x_n)$

With a kernel function $k_h(x, x') = k\left(\frac{\|x - x'\|}{h}\right)$

We can estimate the probability of x $P_{\mathcal{D}}(X = x) = \frac{1}{nh} \sum_{x_i \in \mathcal{D}} k_h(x, x_i)$

# SSPs as Probabilities

- Kernel Density Estimators Examples



# SSPs as Probabilities

*Problems*

- KDE memory grows linearly with the number of observations
- KDE time to compute a probability grow linearly with # of observations

*But…*

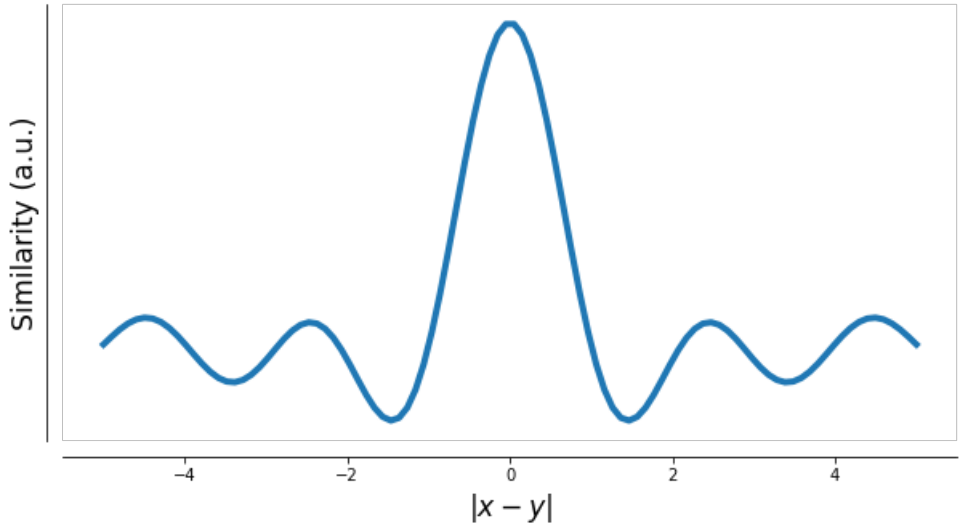
- Not if your kernel is a dot product

$$k_h(\mathbf{x}, \mathbf{x}') \approx \phi_h(\mathbf{x}) \cdot \phi_h(\mathbf{x}') \Rightarrow P(\mathrm{X} = \mathbf{x}) = \frac{1}{nh} \sum_{\mathbf{x}_i \in \mathcal{D}} \phi_h(\mathbf{x}) \cdot \phi_h(\mathbf{x}_i)$$

# SSPs as Probabilities

*SSPs induce a quasi-kernel*

- ‘quasi’ because there are negatives



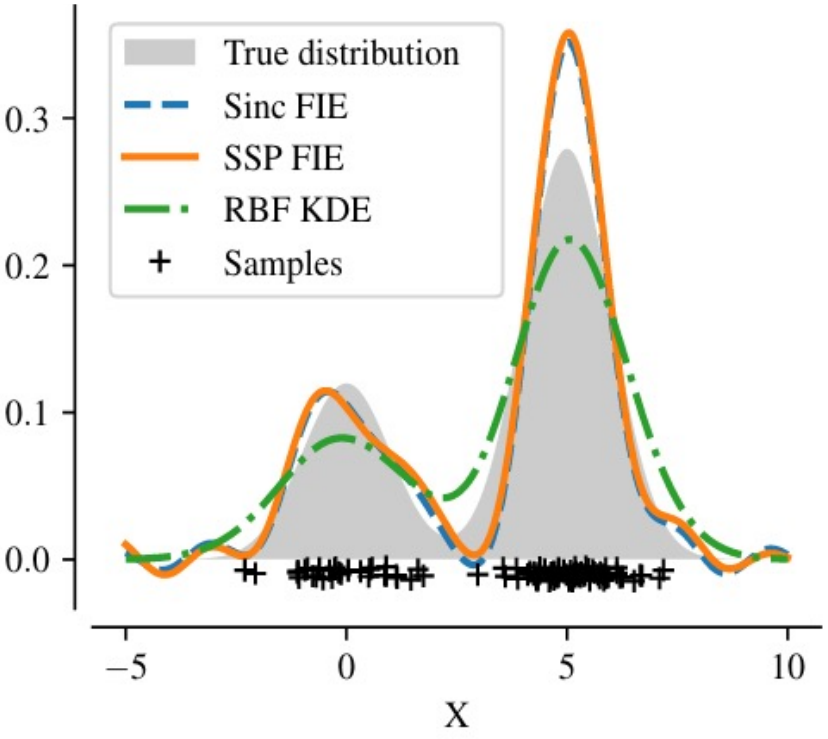

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) \approx \prod_{k=1}^{m} \operatorname{sinc}(|x_k - y_k|)$$

# SSPs as Probabilities

Can convert to probability estimator

$$P_{\mathcal{D}}(X = x) = \max\{0, \phi(x) \cdot M_{\mathcal{D},h} - \xi\}$$

Glad et al, 2003




$$\mathrm{ReLU}(\mathbf{w} \cdot \mathbf{z} + b)$$

# SSPs as Probabilities

So SSP memory is a latent probability distribution

- The distribution is stored in bundles of vector symbols.
- We can apply manipulations to bundles to produce probabilistic statements

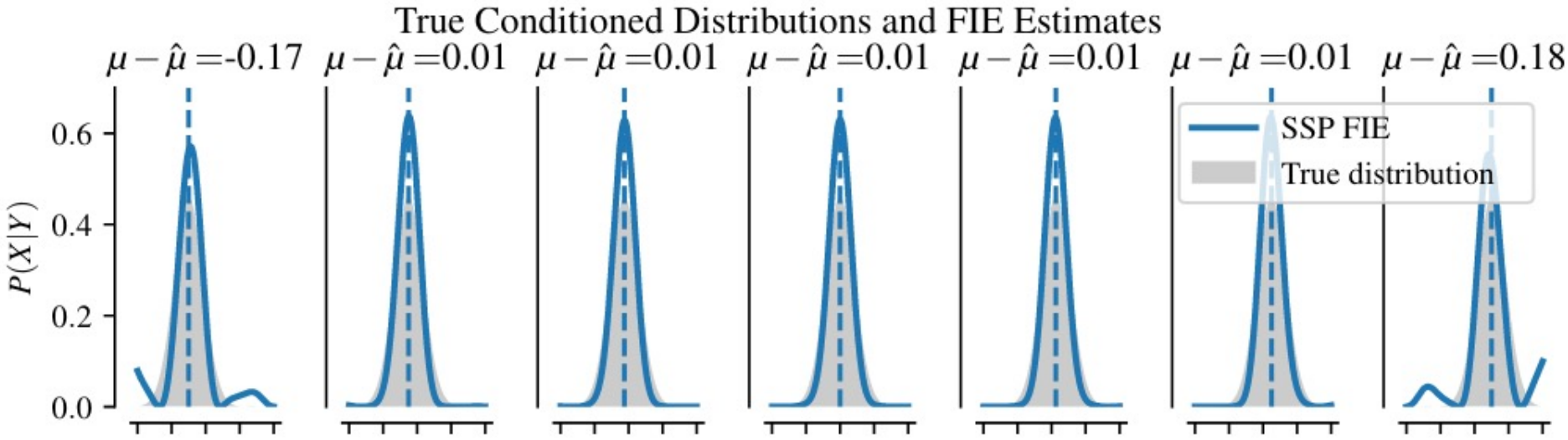
$$M_{\mathcal{D}} = \frac{1}{n} \sum_{\mathbf{x}_i \in \mathcal{D}} \phi(\mathbf{x}_i)$$

$$P(\mathrm{X} = \mathbf{x}) = \phi_h(\mathbf{x}) \cdot M_{\mathcal{D}}$$

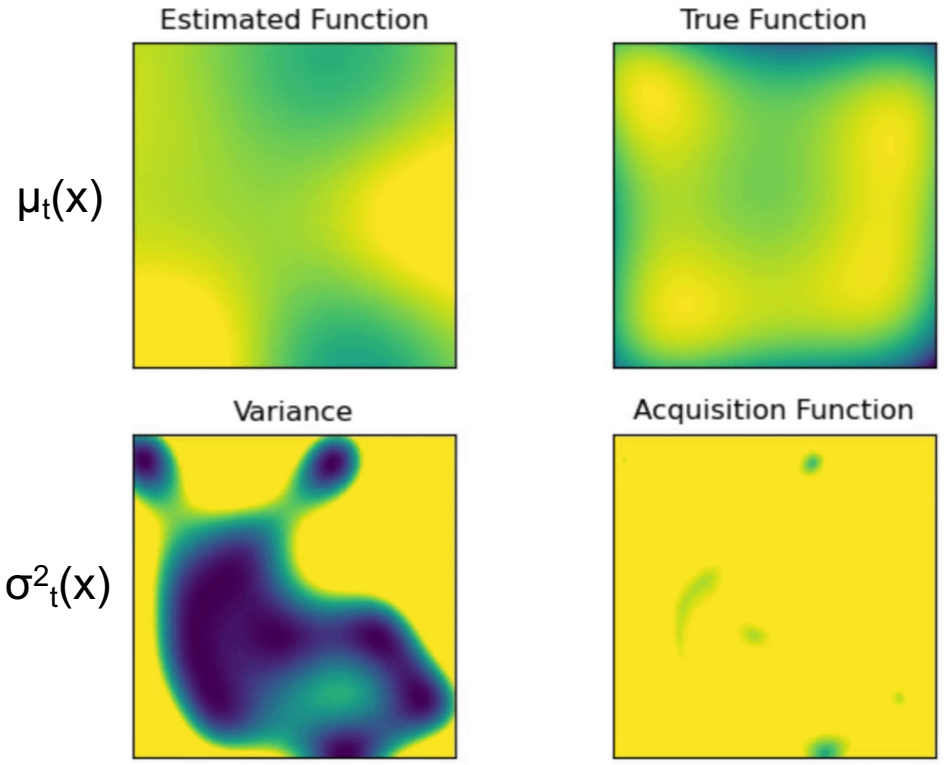
# SSPs as Probabilities

## Conditioning

$$P(\mathrm{X} = x \mid \mathrm{Y} = y) \approx \phi_{\mathrm{X}}(x) \cdot \left[M_{\mathcal{D}} \circledast \phi^{-1}(y)\right]$$



# SSPs as Probabilities

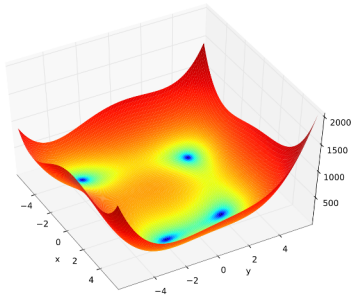


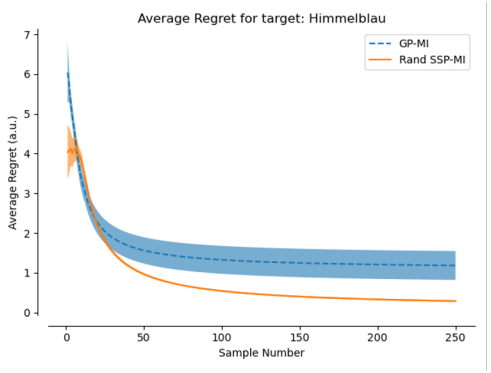

- Mutual Information (MI) is a common objective function used in exploration
- Gaussian Processes (GPs) are a convenient, but computationally intensive tool for computing MI
- We use Spatial Semantic Pointers and Bayesian linear regression to approximate a GP while improving in memory and time complexity

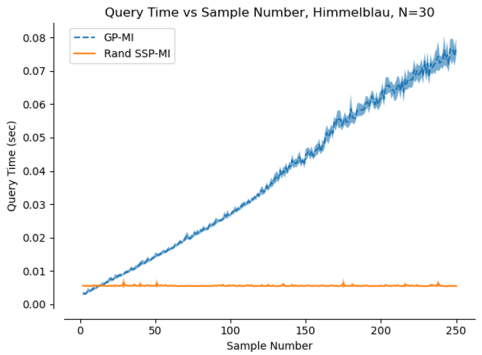
$$MI(X;Y) = H(Y) - H(Y|X) \text{ where } H(Y) = -\sum p(y)log(p(y))$$

# Results

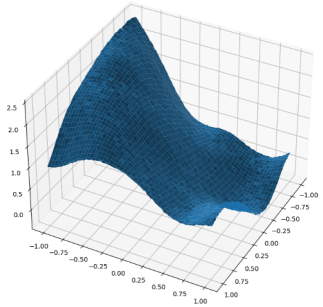
Himmelblau Function

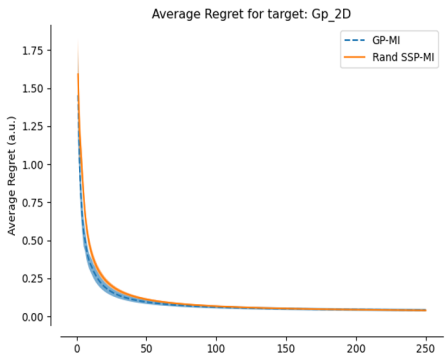


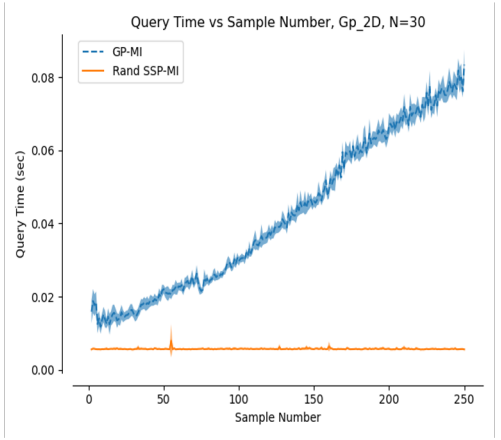


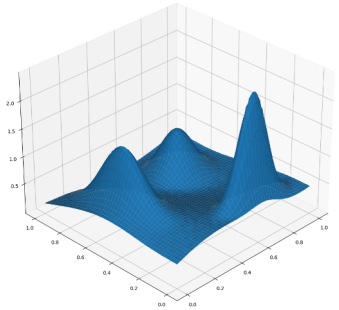

Sampled from GP with Matern Kernel + 1% noise

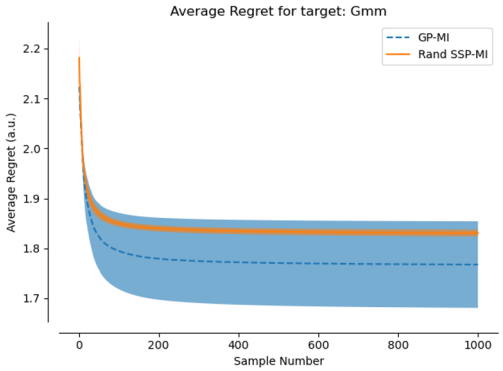


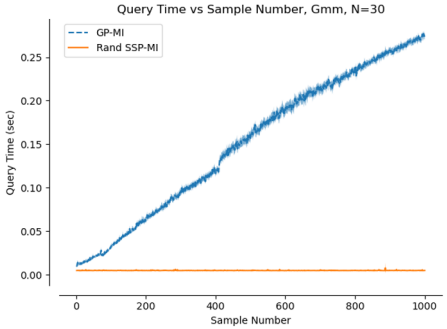



GMM with noise ~ (Matern Kernel + 1% noise)





# SSPs as Probabilities

Where we differ:

- Provide a general and abstract framework for modelling probabilities
- Draw a direct connection between cognitive models and probability statements
- Provide network architectures for conditioning, marginalization, entropy, and mutual information

# Conclusion

- SSPs support a variety of types of inference for cognitive models
  - Binding spatial and ‘symbolic’ representations
  - Representations of sampled data that can be used for probabilistic inference
- Improves
  - Interpretability
  - Efficiency (SWaP critical)
- Best engineered systems continue to learn from understanding biological solutions