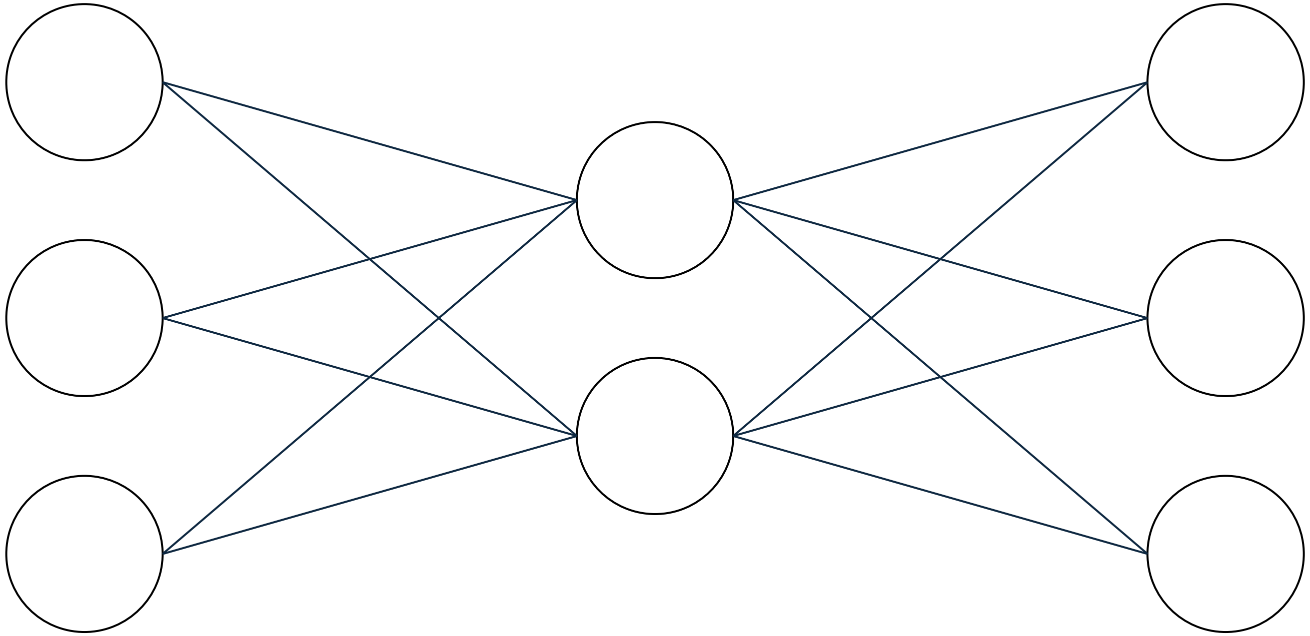
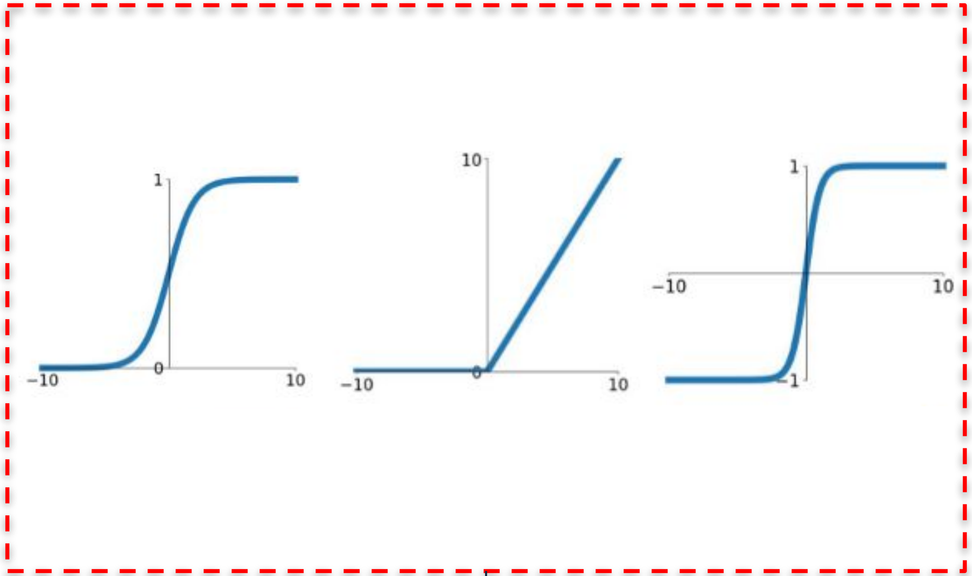
A Characterisation of Convolutional Spiking Autoencoders

Luke Alderson - 02523332

The Foundations

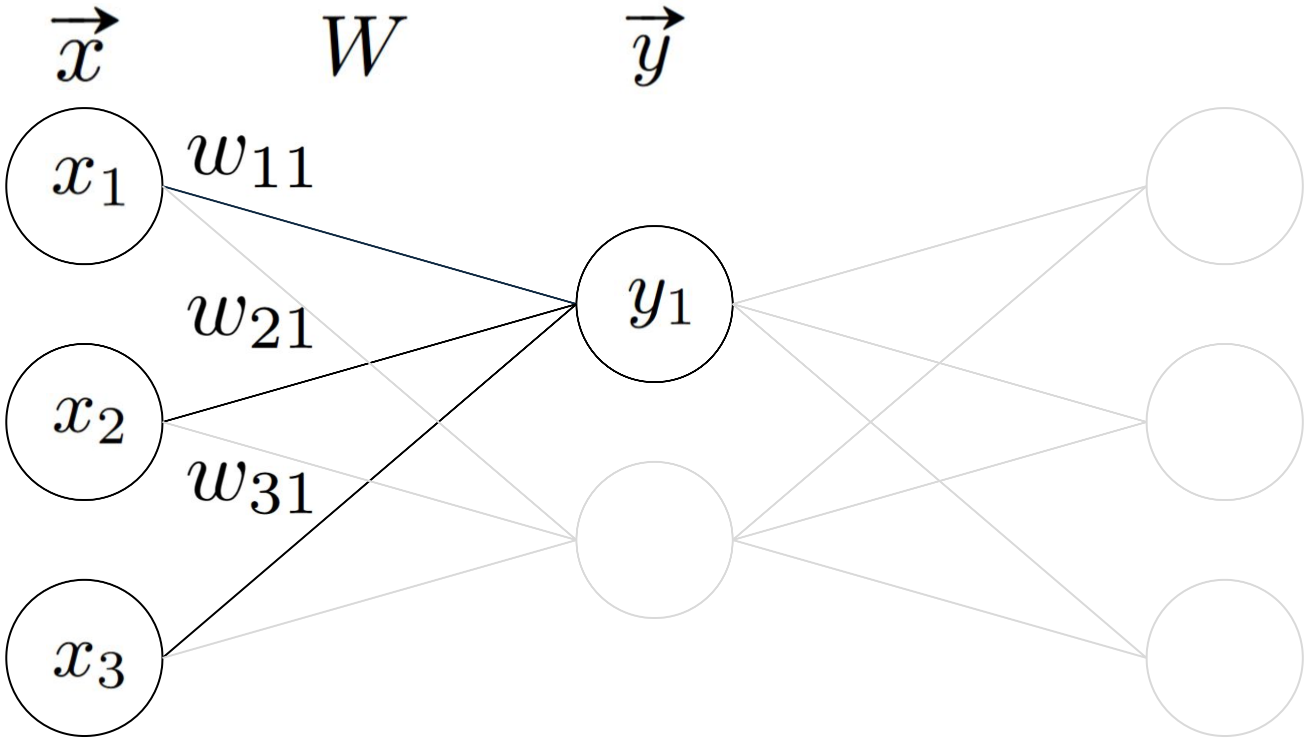


Artificial Neural Network (ANN)

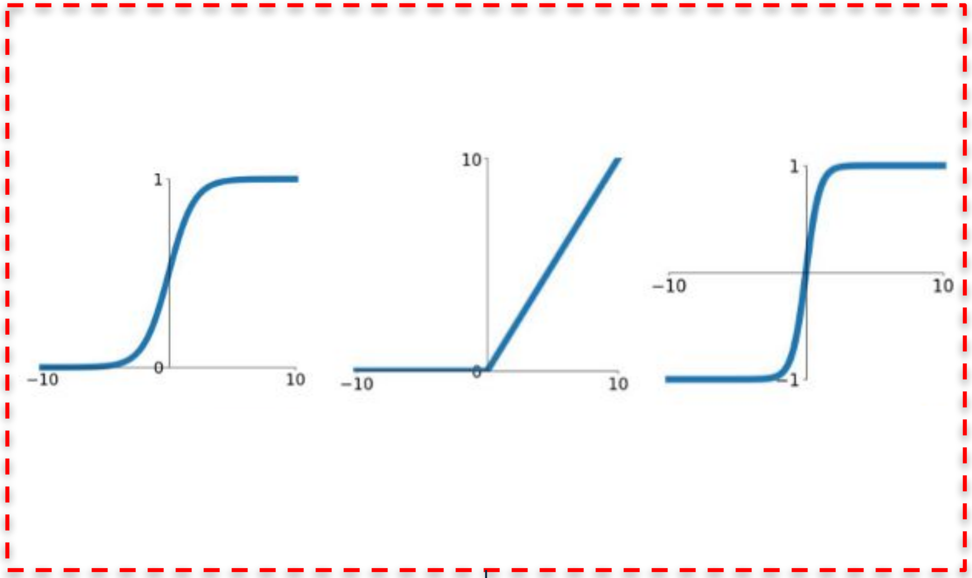


$$\vec{y} = \sigma(W\vec{x} + \vec{b})$$

The Foundations

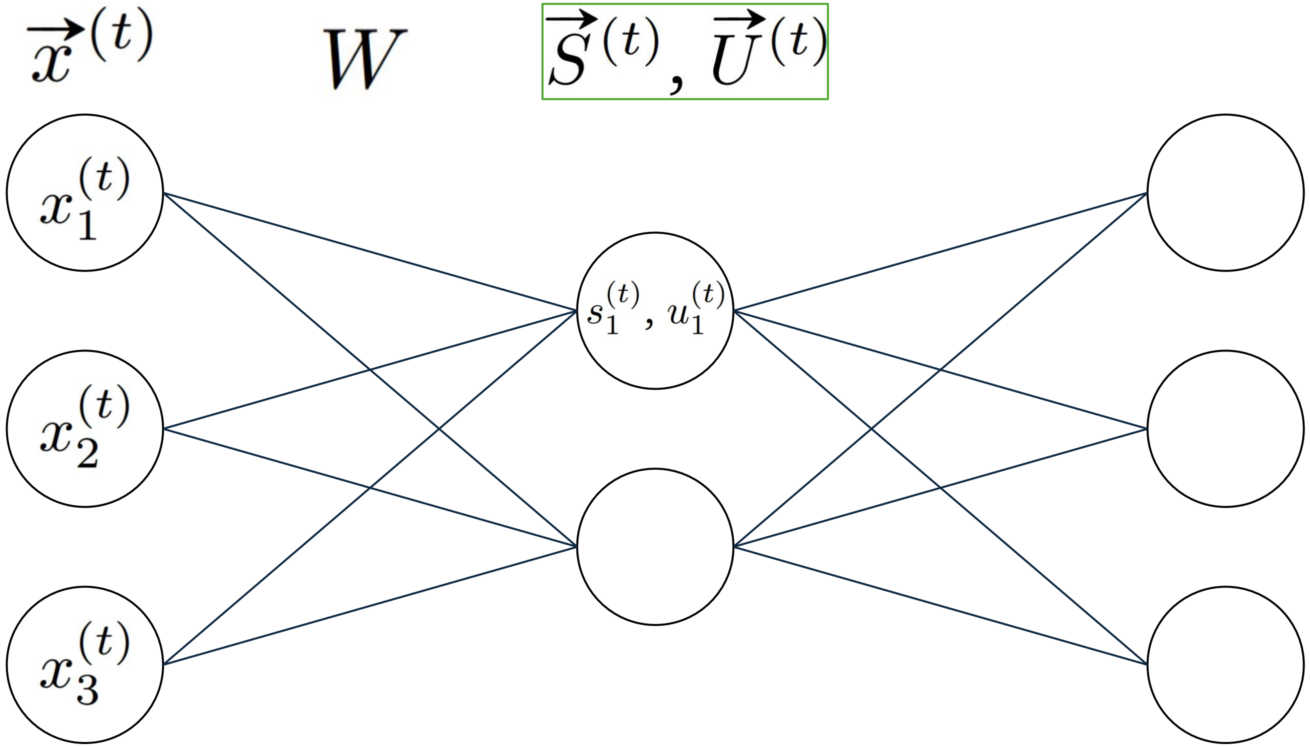


Artificial Neural Network (ANN)



$$\vec{y} = \sigma(W\vec{x} + \vec{b})$$

Spikes and Surrogate Gradients



Spiking Neural Network (SNN)

$$\tau_m \frac{dU(t)}{dt} = -[U(t) - U_{rest}] + R_m I_{ext}(t)$$

$$\boxed{U[t] = \beta U[t-1] + (1-\beta) I_{ext}}$$

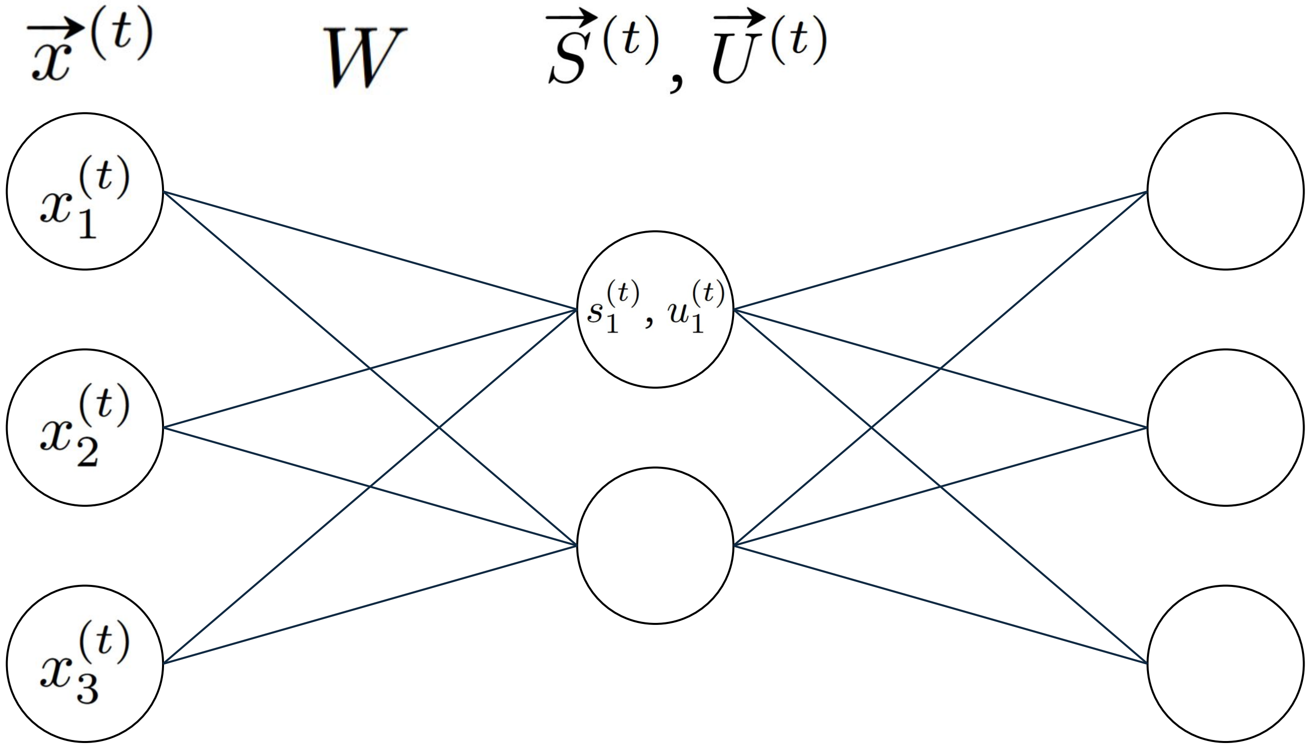
$$\boxed{S_{out}(t) = \begin{cases} 1, & U(t) \geq U_{th} \\ 0, & \text{otherwise} \end{cases}}$$

$$\text{if } U(t) \geq U_{th}, \text{ then } U(t) \rightarrow U_{reset}$$

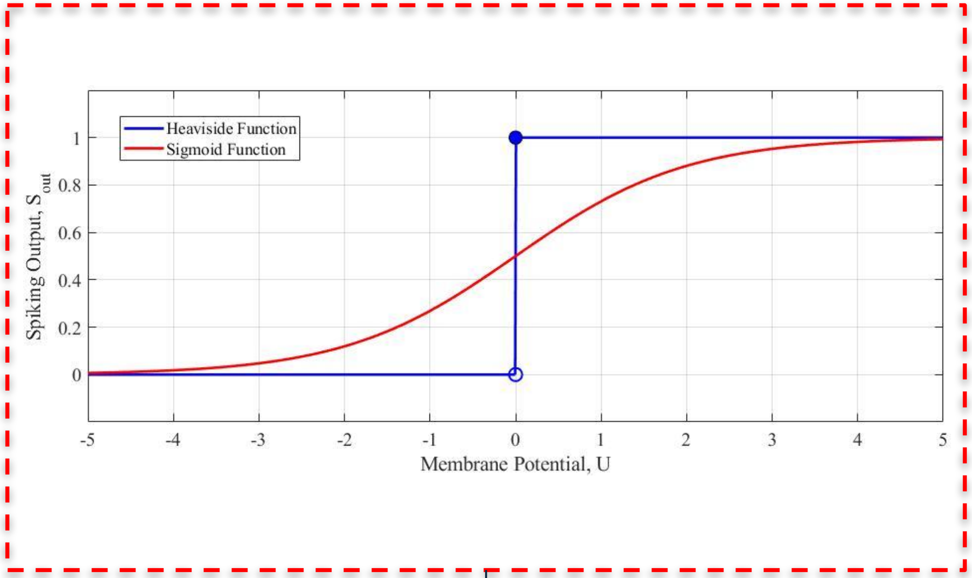
$$\vec{S}^{(t)} = \sigma(\vec{U}^{(t)})$$

(Burkitt, 2006)

Spikes and Surrogate Gradients



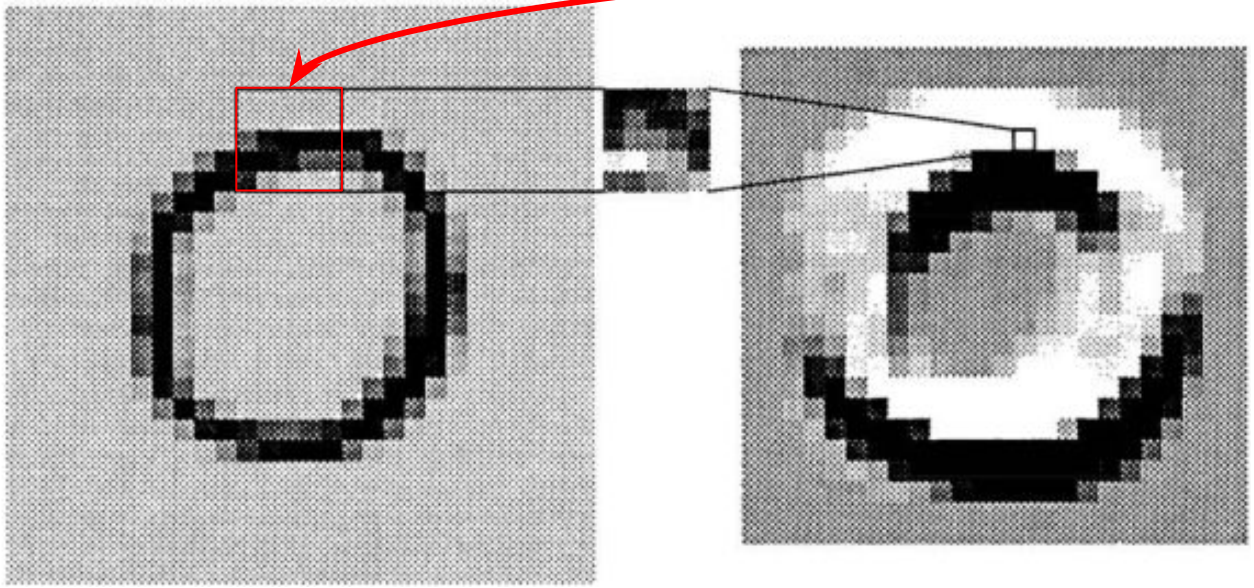
Spiking Neural Network (SNN)



$$\vec{S}^{(t)} = \sigma(\vec{U}^{(t)})$$

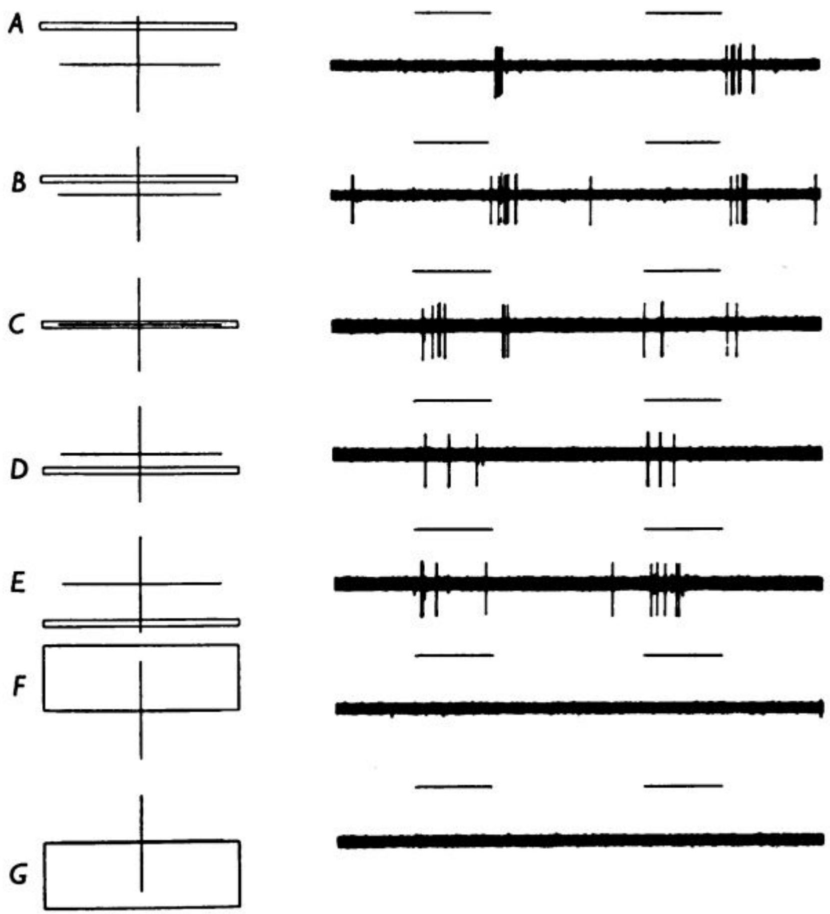
Artificial and Natural Receptive Fields

Convolutional Neural Network



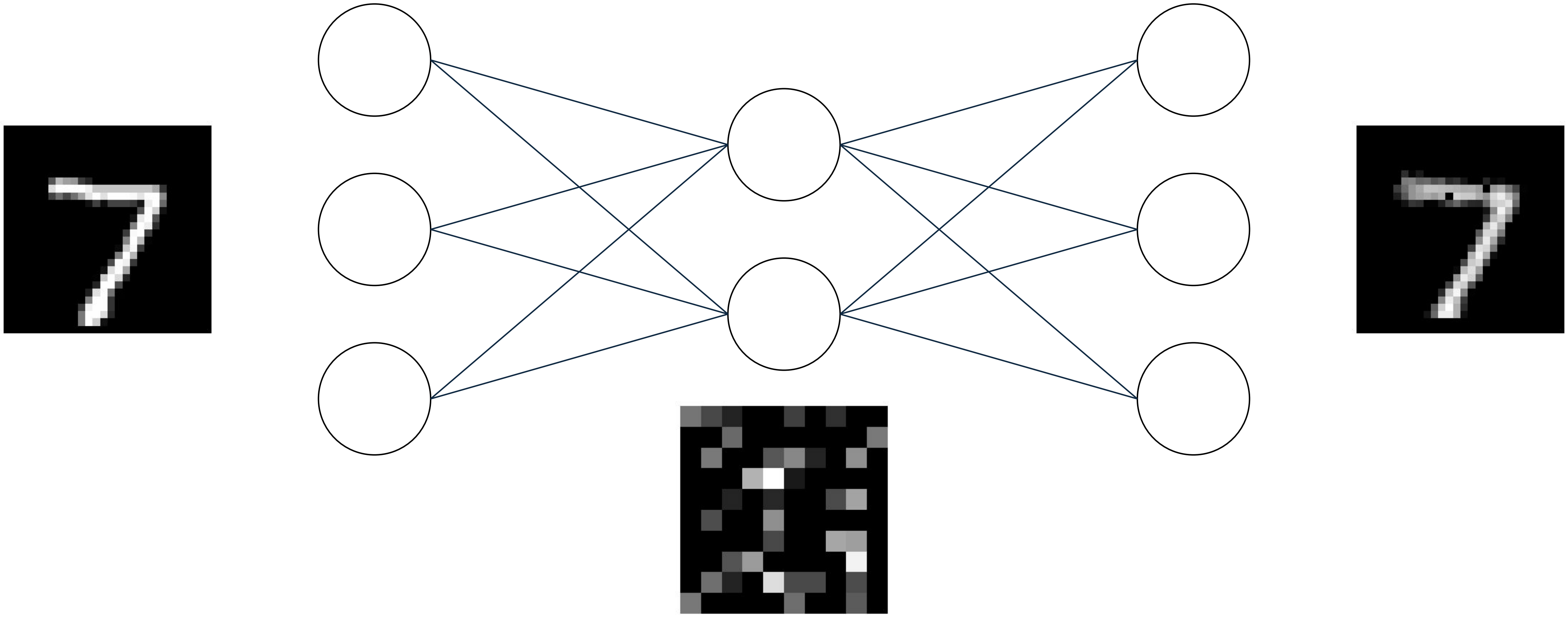
Note: *Darker pixels indicate higher activations.*

Cat Visual Cortex

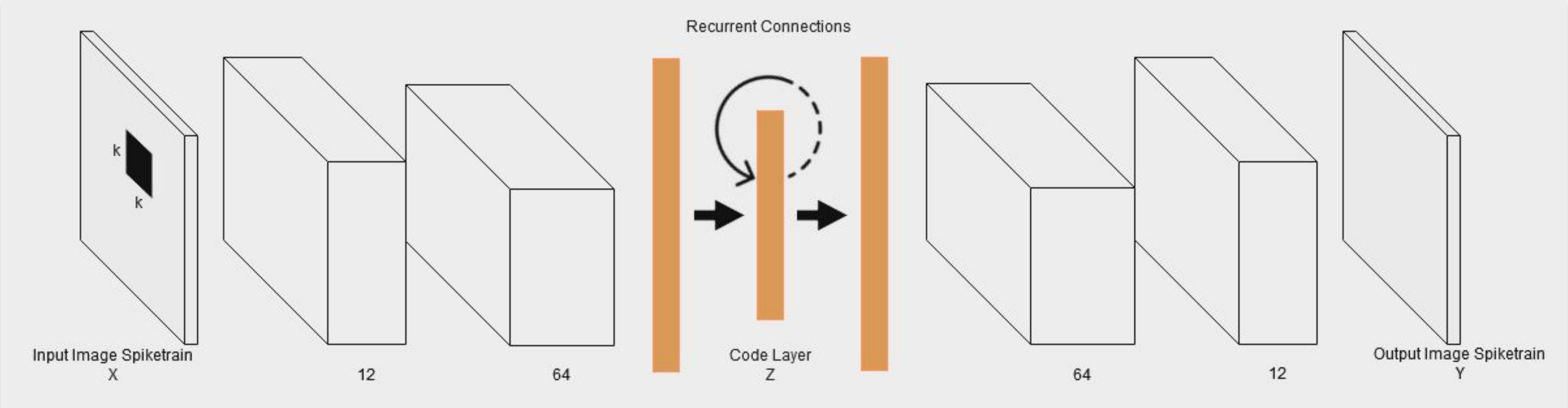


(Le Cun *et al.*, 1989 & Hubel and Wiesel, 1962)

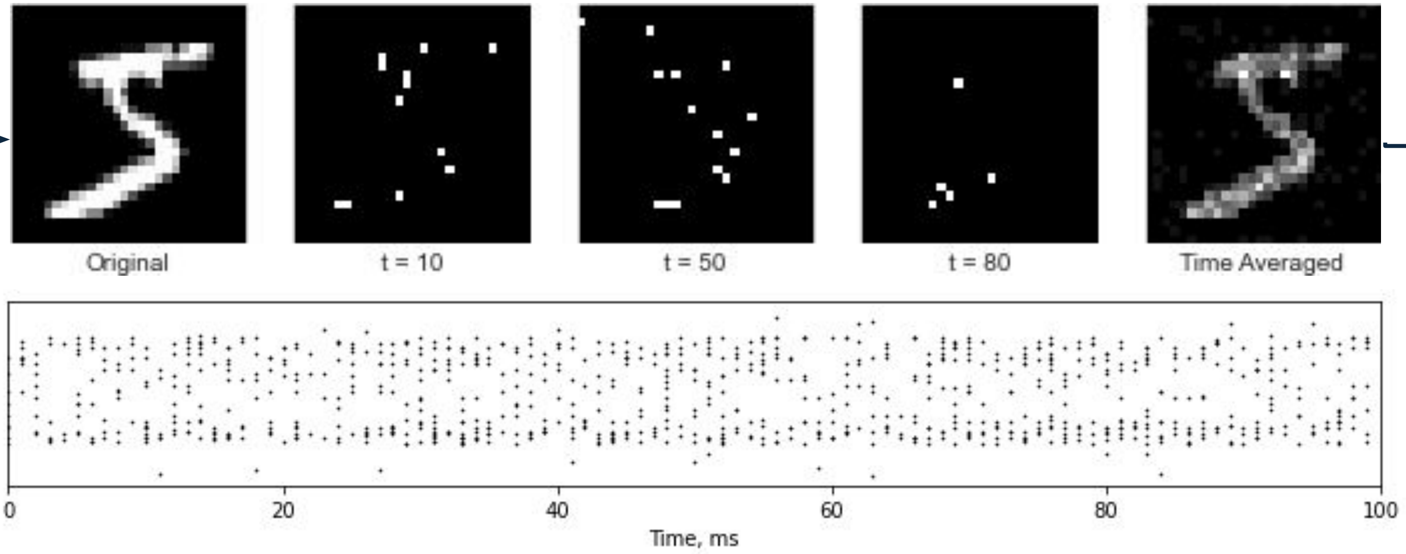
Autoencoders

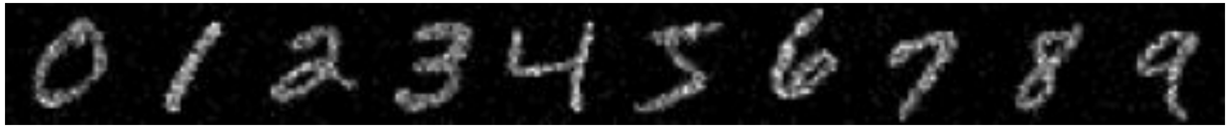


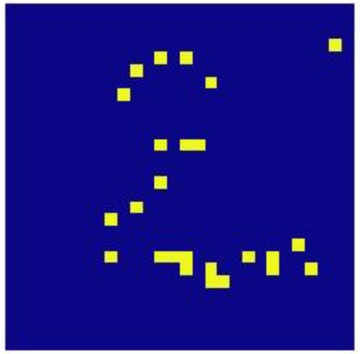
Convolutional, Spiking Autoencoder



MNIST - From Floats to Spikes



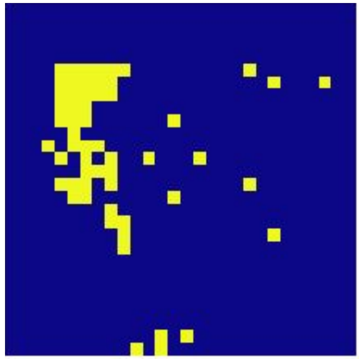


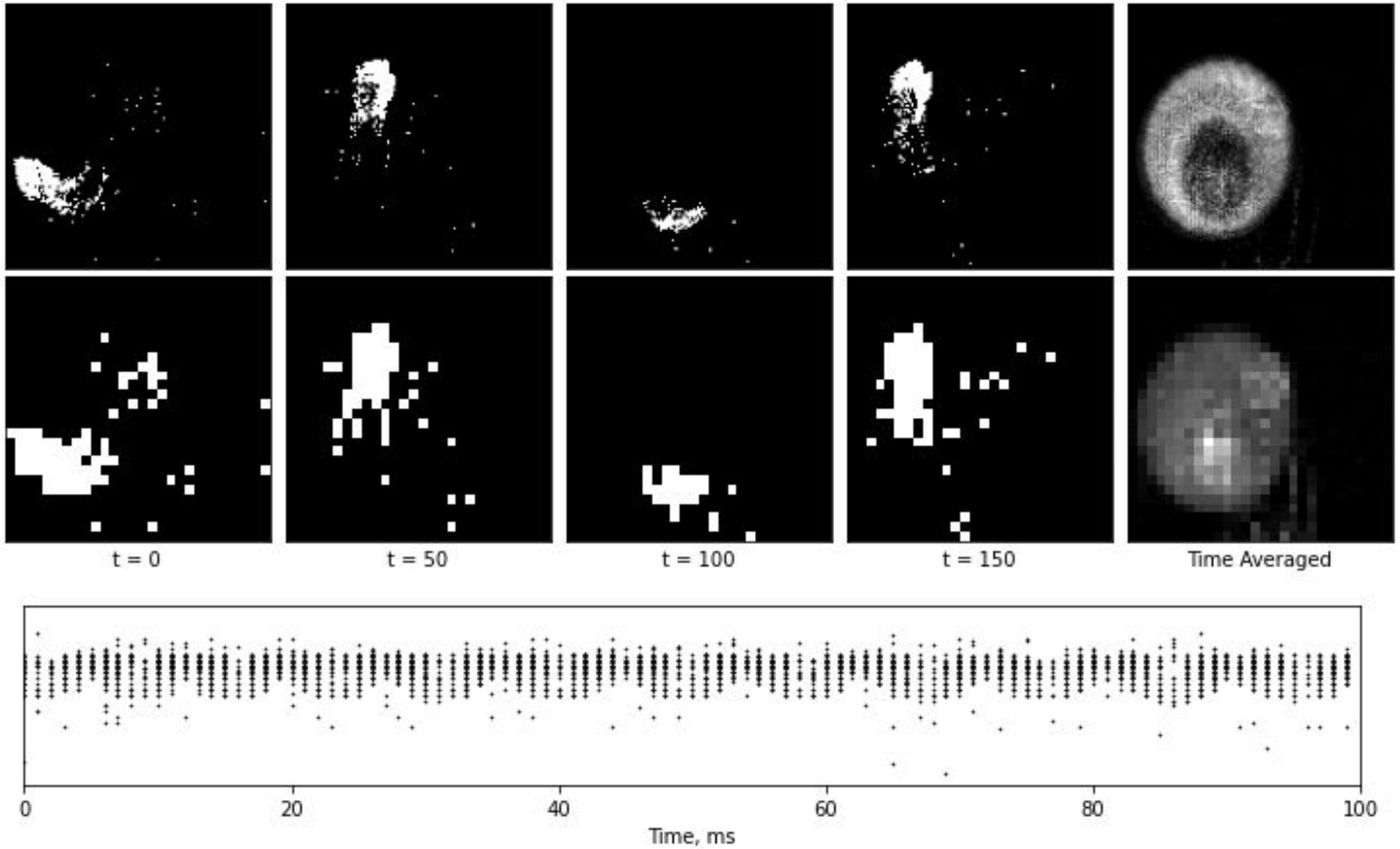


DVS Gesture - Natively Neuromorphic

Native:
- 128 x 128
- Microsecond

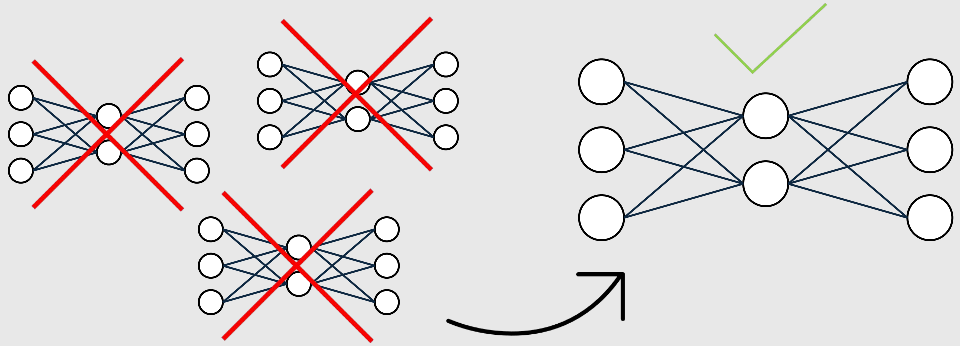
Downsampled:
- 28 x 28
- Millisecond



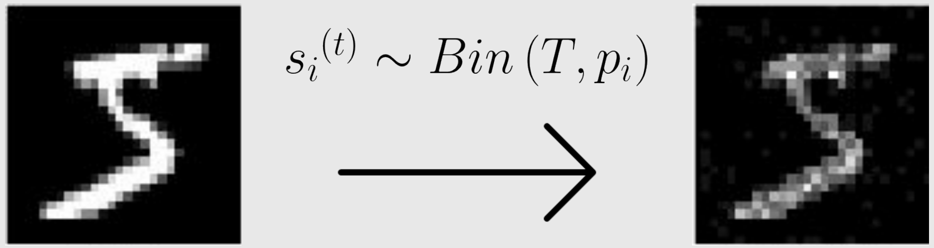


Experiments

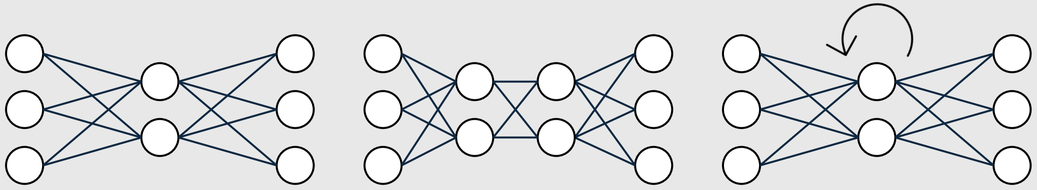
Optimal Hyperparameters in SAE and AE



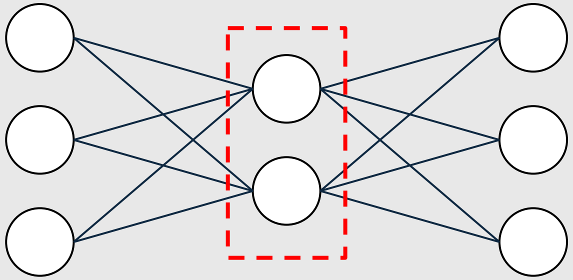
Encoding Frequency



Recurrent Connections

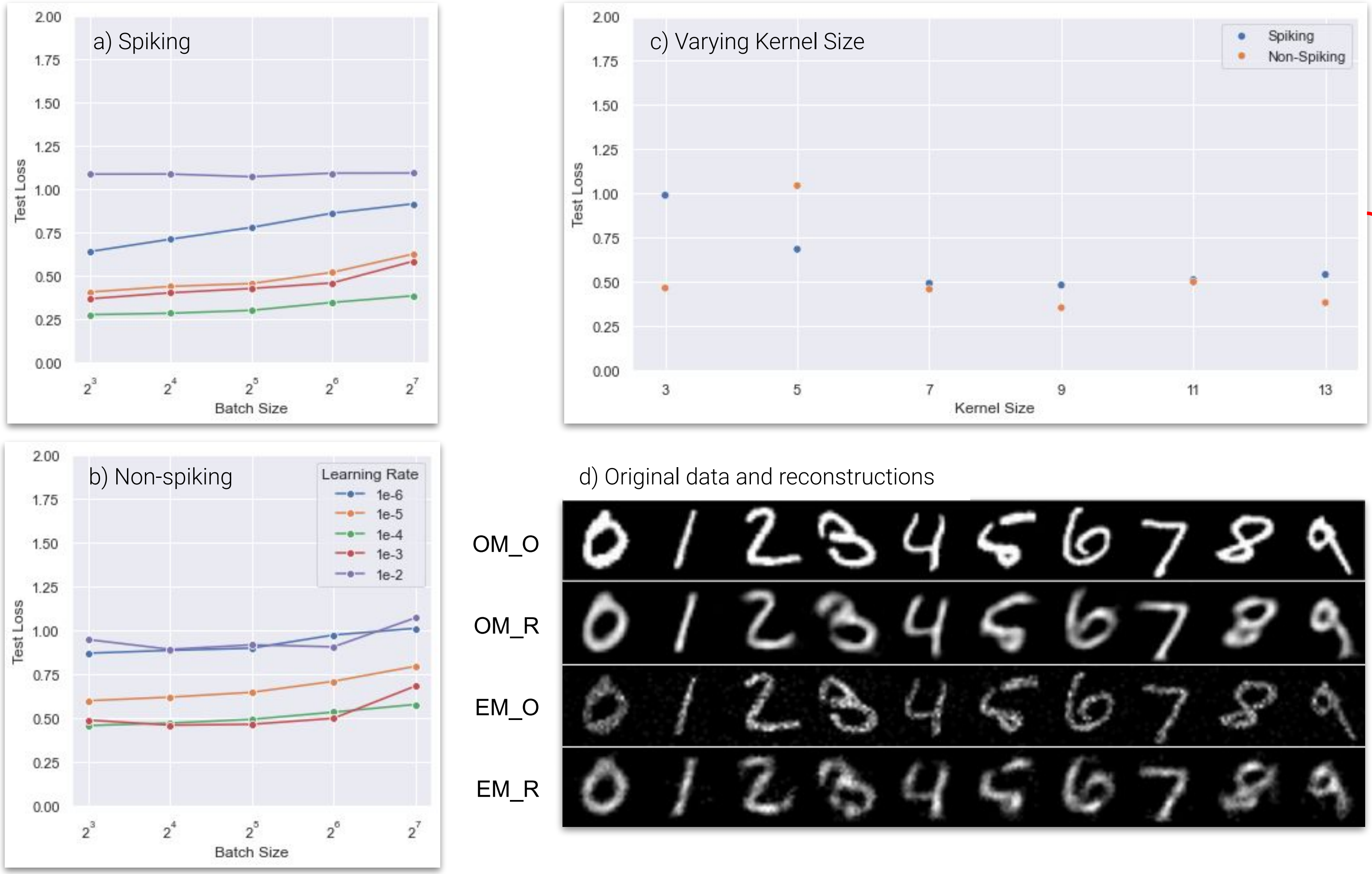


Embedding Size



$\{x \in \mathbb{N} \,|\, 2 \leq x \leq 1000\}$

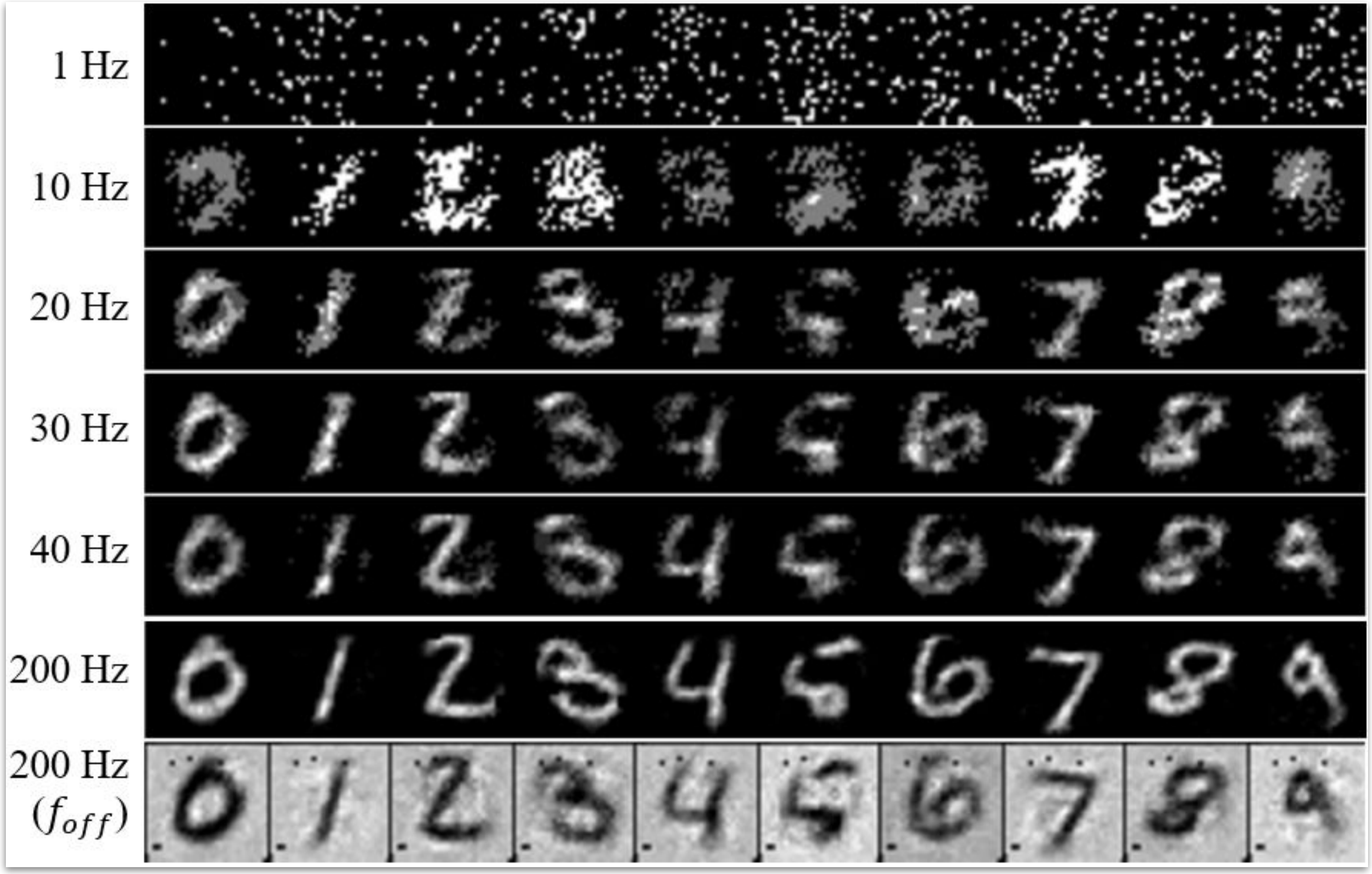
Increasing kernel size reduces loss



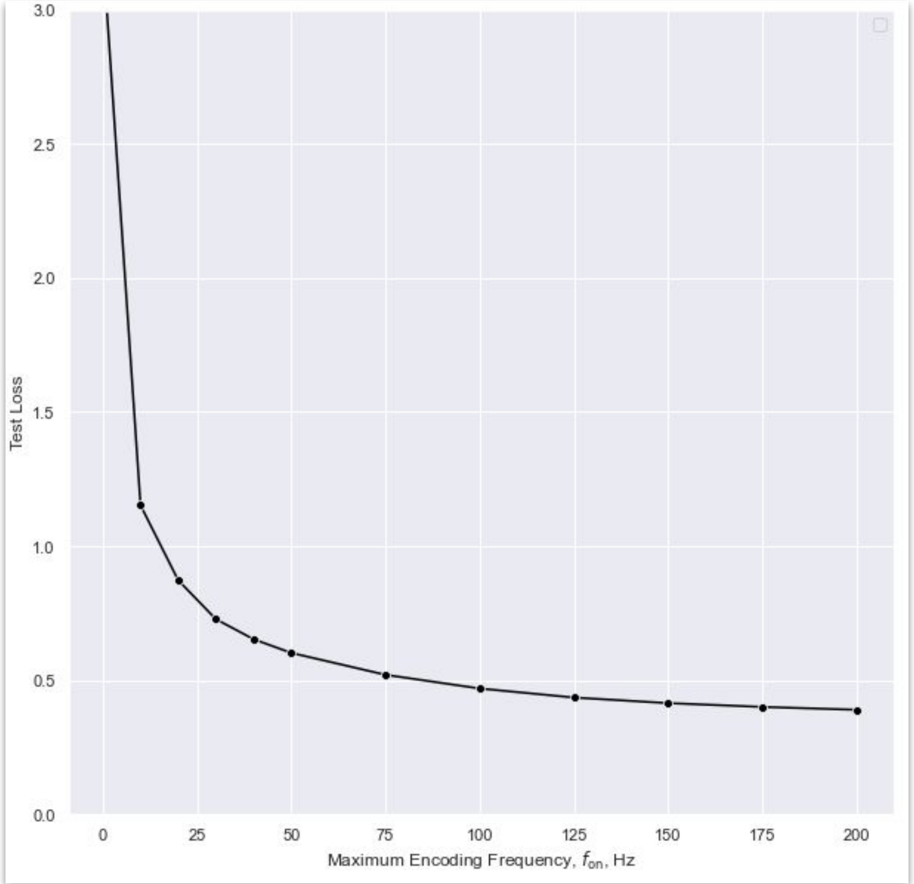
Optimal Parameters	
Batch Size	64
Learning Rate	1e-4
Kernel Size	7

Disparity in rate coding acts as contrast

Spiking Reconstructions

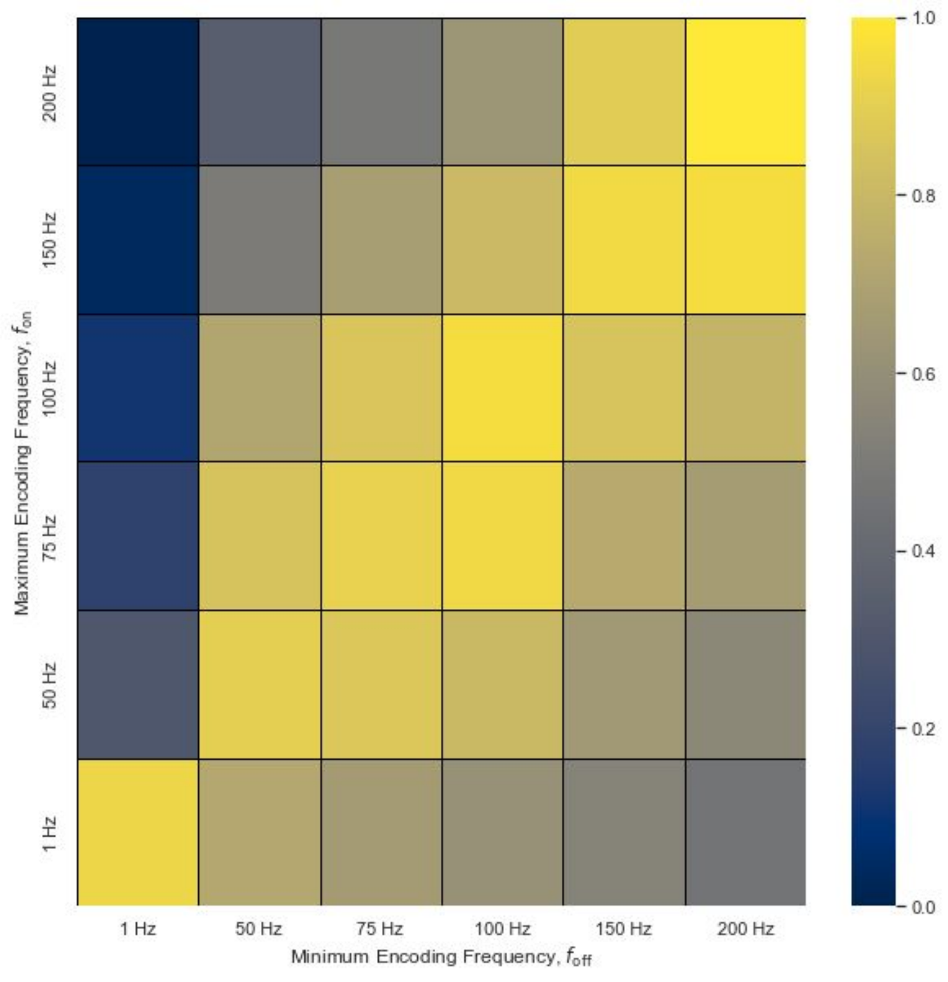


Loss v. Max Encoding Frequency

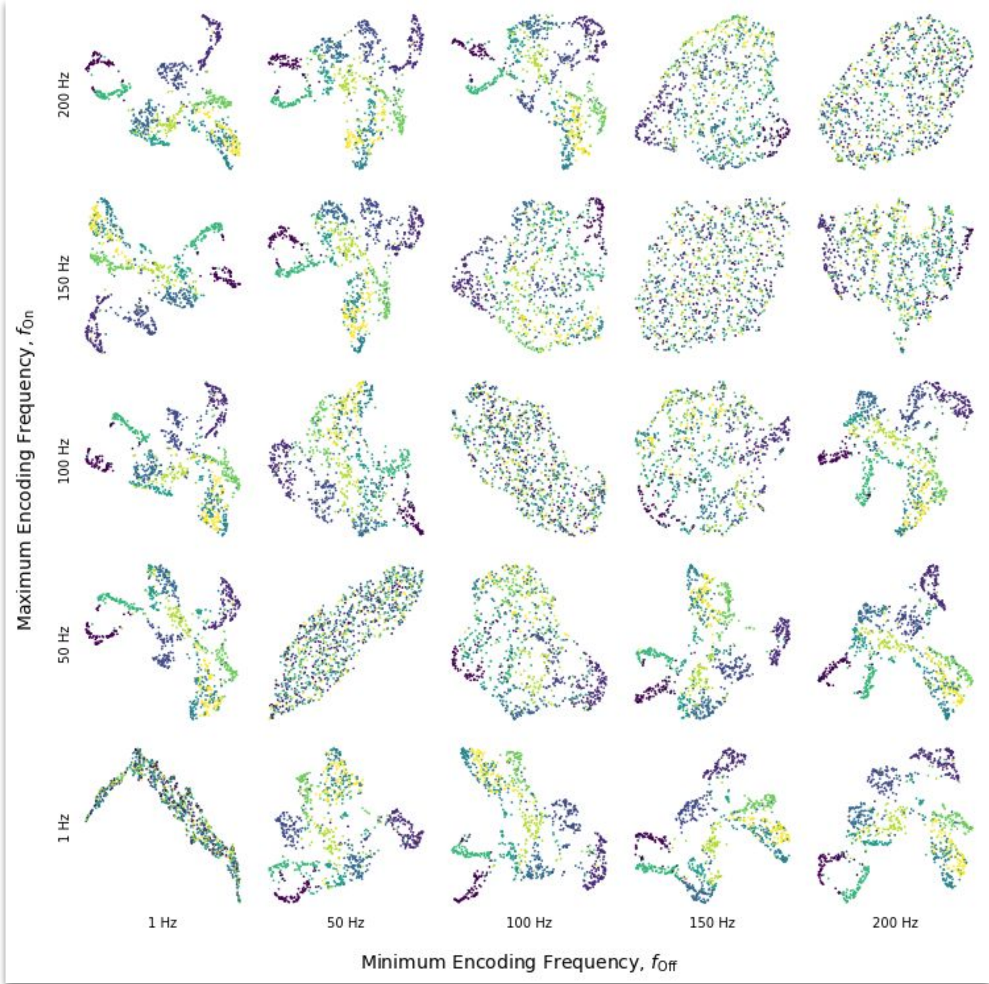


Disparity in rate coding acts as contrast

Loss Heatmap

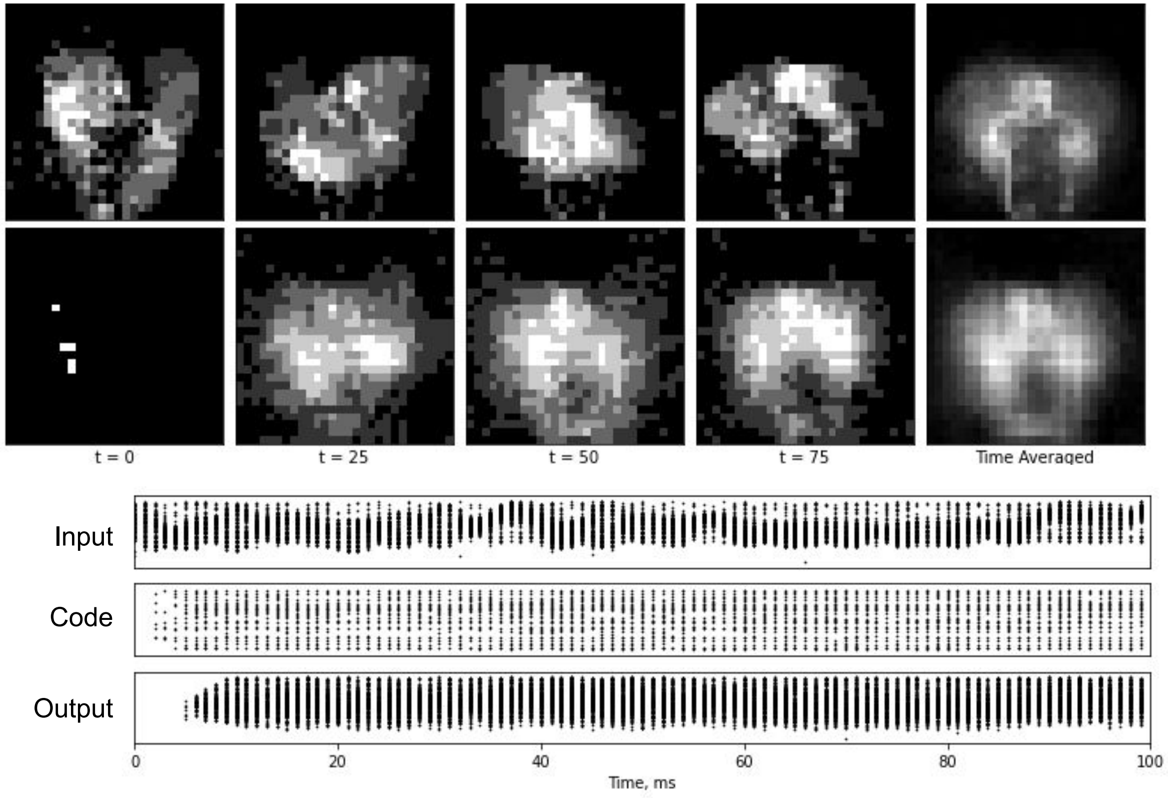


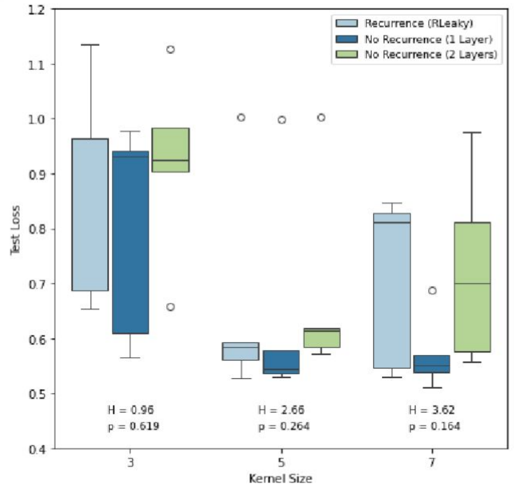
UMAP



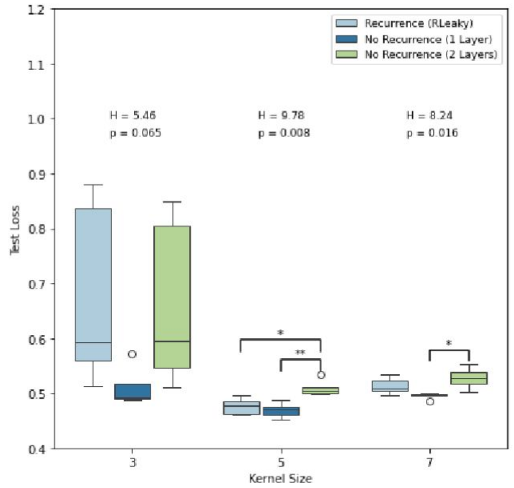
(UMAP method: McInnes *et al.*, 2018)

Recurrence did not reduce loss

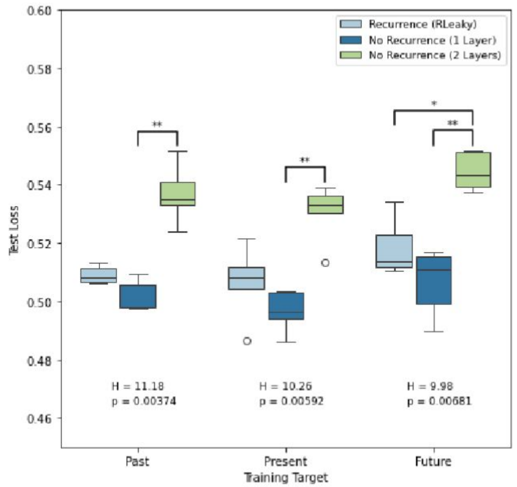




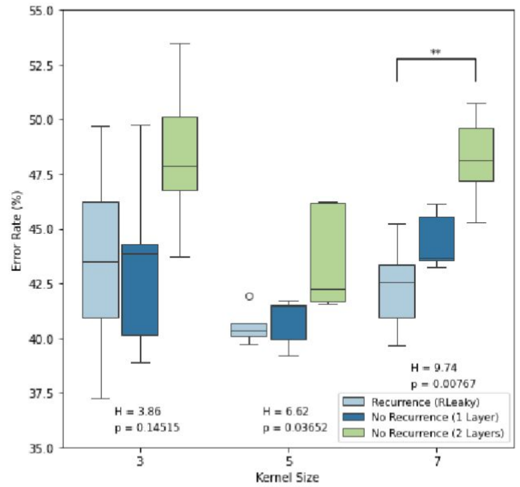
(a) SAE trained on MNIST data with varying kernel size



(b) SAE trained on DVS data with varying kernel size



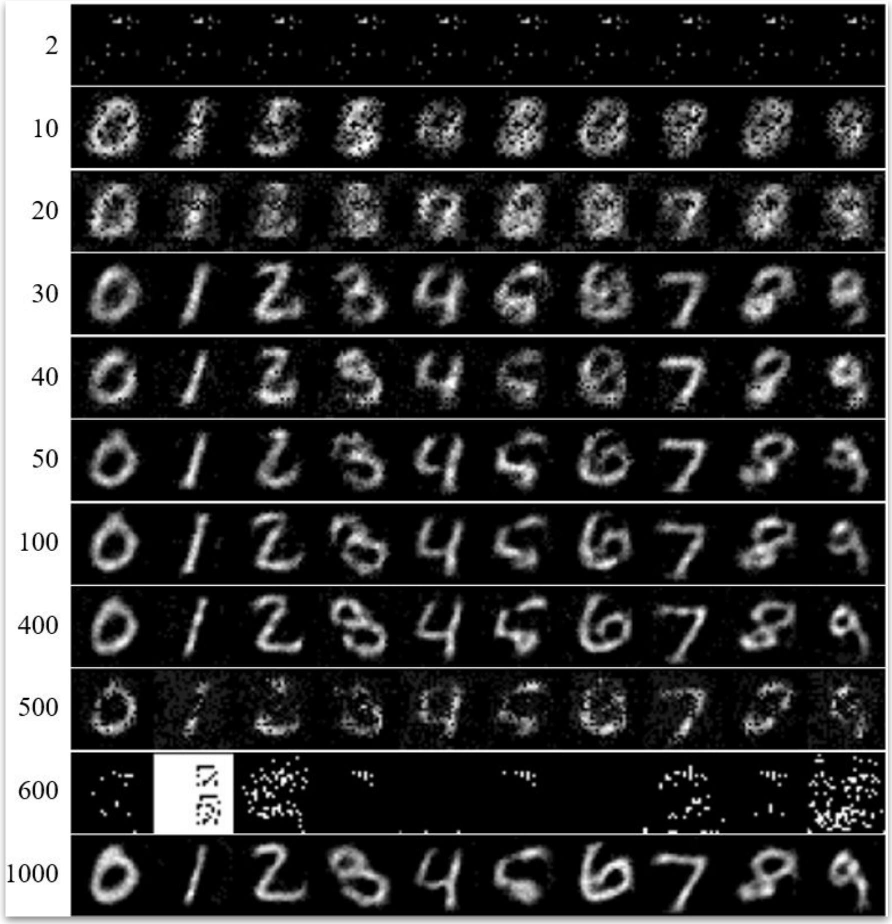
(c) SAE trained on DVS data with varying training targets



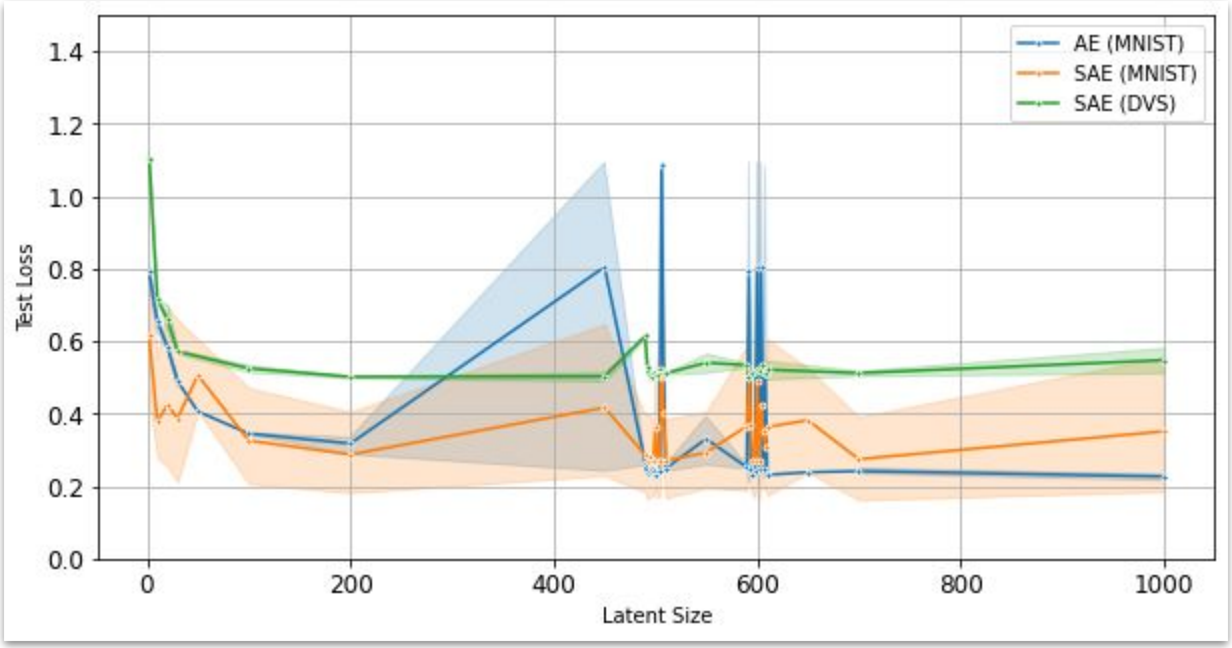
(d) Classifier trained on DVS data with varying kernel size

Embedding Size ***generally*** lowers loss

Spiking Reconstructions

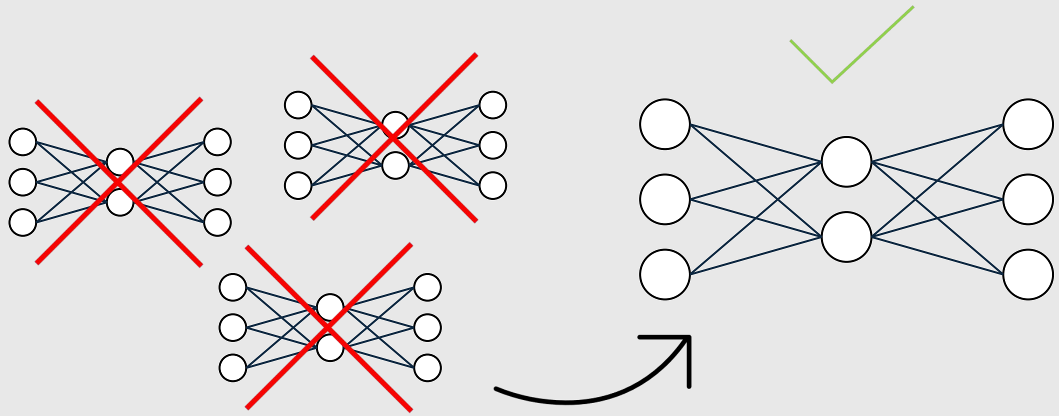


Loss v. Embedding Size

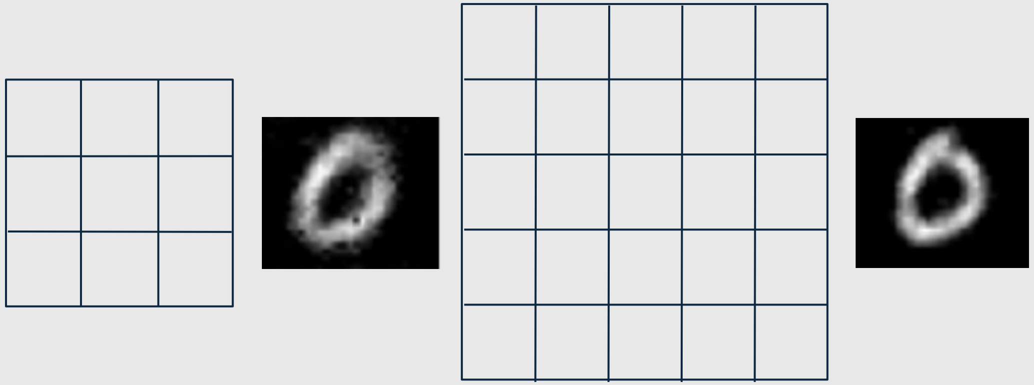


Discussion - Hyperparameter Sweep

Hyperparameter Sweep



Increasing kernel size reduced loss



(Putra and Shafique., 2024; Eshraghian *et al.*, 2023)

Discussion - Encoding Strategy

Encoding Frequency

$$s_i^{(t)} \sim Bin\left(T, p_i\right)$$

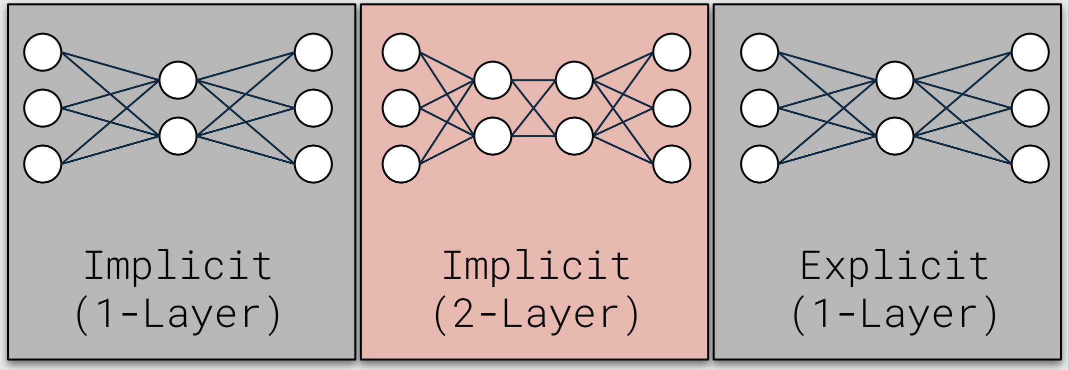
Frequency disparity improved embedding quality

$$\uparrow \Delta f \longrightarrow$$

(Ben-Shaul *et al.*, 2023; Eshraghian *et al.*, 2023)

Discussion - Recurrence

Recurrent Connections



<u>MNIST</u>

No effect (Bouanane *et al.*, 2023)

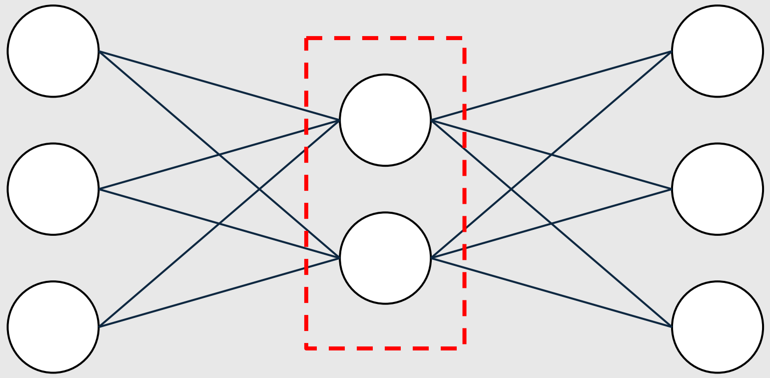
<u>DVS</u>

- 1-Layer, implicit or explicit - no effect.
- Try different loss function.
  - (Cramer *et al.*, 2020)

- 2-Layer - increased loss.
- Vanishing gradients?
  - (Zheng *et al*,. 2021; Ledinauskas *et al*,. 2020)

Discussion - Embedding Size

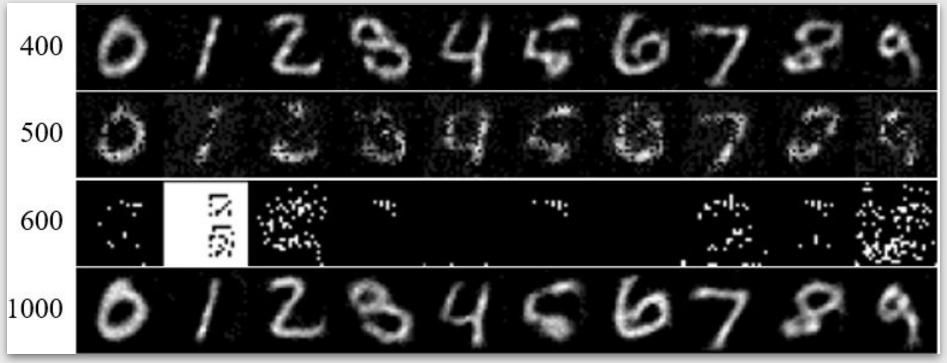
Embedding Size



$$\{x \in \mathbb{N} \,|\, 2 \leq x \leq 1000\}$$

Embedding size generally reduced loss

... with difficulty at specific sizes.



Occurred for spiking and non-spiking nets.

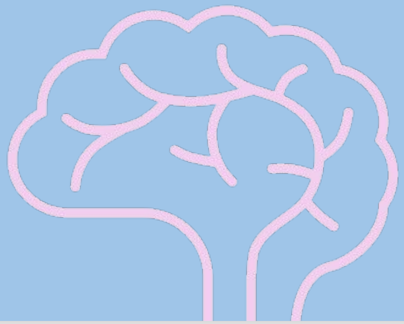
(Benmeziane *et al.*, 2023; Dobson, 2023; Li *et al.*, 2017)

Limitations and Future work

- Only encoding rate was varied.
  - Constrain hidden layer firing rates with regularisation (Hübotter *et al.*, 2021).
- Vanishing gradient problems.
  - E-prop instead of conventional backpropagation (Hoyer *et al.*, 2022).
- Reliance on rate coding.
  - Temporal / Delta / Novel Coding Schemes (Mehta *et al.*, 2002; Ainsworth *et al.*, 2012).

Thank you for listening!

Any questions?



Appendices - References

Ainsworth, M., Lee, S., Cunningham, M.O., Traub, R.D., Kopell, N.J. and Whittington, Miles A. (2012). Rates and Rhythms: A Synergistic View of Frequency and Temporal Coding in Neuronal Networks. *Neuron*, 75(4), pp.572–583. doi:https://doi.org/10.1016/j.neuron.2012.08.004.

Benmeziane, H., Ounnoughene, Amine Ziad, Hamzaoui, I. and Bouhadjar, Y. (2023). Skip Connections in Spiking Neural Networks: An Analysis of Their Effect on Network Training. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2303.13563.

Ben-Shaul, I., Shwartz-Ziv, R., Galanti, T., Dekel, S. and LeCun, Y. (2023). Reverse Engineering Self-Supervised Learning. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2305.15614.

Burkitt, A.N. (2006). A Review of the Integrate-and-fire Neuron Model: I. Homogeneous Synaptic Input. *Biological Cybernetics*, 95(1), pp.1–19. doi:https://doi.org/10.1007/s00422-006-0068-6

Cramer, B., Yannik Stradmann, Schemmel, J. and Zenke, F. (2022). The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks. *IEEE transactions on neural networks and learning systems*, 33(7), pp.2744–2757. doi:https://doi.org/10.1109/tnnls.2020.3044364.

Dobson, J.E. (2023). On reading and interpreting black box deep neural networks. *International Journal of Digital Humanities*, 5(2-3), pp.431–449. doi:https://doi.org/10.1007/s42803-023-00075-w.

Eimantas Ledinauskas, Ruseckas, J., Alfonsas Jursenas and Giedrius Burachas (2020). Training Deep Spiking Neural Networks. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2006.04436.

Eshraghian, J.K., Ward, M., Emre Neftci, Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Doo Seok Jeong and Lü, W. (2023). Training Spiking Neural Networks Using Lessons From Deep Learning. *Proceedings of the IEEE*, 111(9), pp.1016–1054. doi:https://doi.org/10.1109/jproc.2023.3308088.

Hoyer, M., Shahram Eivazi and Otte, S. (2022). Efficient LSTM Training with Eligibility Traces. *Lecture notes in computer science*, pp.334–346. doi:https://doi.org/10.1007/978-3-031-15934-3_28.

Hubel, D.H. and Wiesel, T.N. (1962). Receptive fields, Binocular Interaction and Functional Architecture in the cat's Visual Cortex. *The Journal of Physiology*, [online] 160(1), pp.106–154. doi:https://doi.org/10.1113/jphysiol.1962.sp006837.

Hübotter, J.F., Lanillos, P. and Tomczak, J.M. (2021). Training Deep Spiking Auto-encoders without Bursting or Dying Neurons through Regularization. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2109.11045.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4), pp.541–551. doi:https://doi.org/10.1162/neco.1989.1.4.541.

Li, H., Xu, Z., Taylor, G., Studer, C. and Goldstein, T. (2018). *Visualizing the Loss Landscape of Neural Nets*. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.1712.09913.

McInnes, L., Healy, J., Saul, N. and Großberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3(29), p.861. doi:https://doi.org/10.21105/joss.00861.

Mehta, M.R., Lee, A.K. and Wilson, M.A. (2002). Role of experience and oscillations in transforming a rate code into a temporal code. *Nature*, [online] 417(6890), pp.741–746. doi:https://doi.org/10.1038/nature00807.

Mohamed Sadek Bouanane, Cherifi, D., Chicca, E. and Lyes Khacef (2023). Impact of spiking neurons leakages and network recurrences on event-based spatio-temporal pattern recognition. *Frontiers in Neuroscience*, 17(17). doi:https://doi.org/10.3389/fnins.2023.1244675.

Putra, and Shafique, M. (2024). A Methodology for Improving Accuracy of Embedded Spiking Neural Networks through Kernel Size Scaling. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.2404.01685.

Zheng, H., Wu, Y., Deng, L., Hu, Y. and Li, G. (2021). Going Deeper With Directly-Trained Larger Spiking Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12), pp.11062–11070. doi:https://doi.org/10.1609/aaai.v35i12.17320.

Appendices - CNN

X

1	2	3	4	2	1
4	5	6	5	0	1
7	8	9	2	2	1
9	3	2	4	2	1
5	7	3	2	1	9
4	4	3	2	8	1

W

1	1	0
0	0	0
0	0	1

$+ b$

Y

12			

$$Y = \sigma(W \cdot X + b)$$

Assume b = 0 and the activation function is the identity.

Appendices - CNN

X

1	2	3	4	2	1
4	5	6	5	0	1
7	8	9	2	2	1
9	3	2	4	2	1
5	7	3	2	1	9
4	4	3	2	8	1

W

1	1	0
0	0	0
0	0	1

$+b$

Y

12	7		

$$Y = \sigma(W \cdot X + b)$$

Assume b = 0 and the activation function is the identity.

Appendices - CNN

X

1	2	3	4	2	1
4	5	6	5	0	1
7	8	9	2	2	1
9	3	2	4	2	1
5	7	3	2	1	9
4	4	3	2	8	1

W

1	1	0
0	0	0
0	0	1

$+b$

Y

12	7	9	7
11	15	13	6
18	19	12	13
15	7	14	7

$$Y = \sigma(W \cdot X + b)$$

Assume b = 0 and the activation function is the identity.

Appendices - Loss Function

$$L(\theta)_{nRMSE} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})^2}} = \frac{L(\theta)_{RMSE}}{\sigma_X}$$

where

$L(\theta)_{RMSE}$ is the RMSE loss as a function of the network parameters.

N is the number of neurons.

X_i is the input spike count of the ith neuron.

$\hat{X}_i$ is the output spike count of the ith neuron.

$\bar{X}$ is the mean input spike count.

σ_X is the standard deviation of the input data.

Appendices - Spike Encoding

$$p_i = \Delta t[(f_{on} - f_{off})x_i + f_{off}] \tag{6}$$

where

Δt is the size of the time-step for the simulation

x_i is the normalised value of the ith pixel in the input image

$f_{off} \in [0, \frac{1}{\Delta t}]$ is the minimum firing rate in Hz

$f_{on} \in [0, \frac{1}{\Delta t}]$ is the maximum firing rate in Hz

$$S_i^{(t)} \sim Bin(T, p_i) \tag{5}$$

where

$S_i^{(t)} \in \{0, 1\}$ is the random variable describing spike generation from neuron i at time t

T is the total number of time-steps in the spike train

p_i is the probability of a spike occurring from neuron i

Appendices - Backpropagation

Backpropagation faces challenges when applied directly to SNNs because spikes are modelled as instances of the Dirac-delta function, making them non-differentiable. The general form of the chain rule for a SNN is given by:

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial S_{out}} \frac{\partial S_{out}}{\partial U} \frac{\partial U}{\partial W_{ij}} \tag{2}$$

where L is the loss function, such as the Euclidean norm, W_{ij} represents the weight between the ith neuron of a given layer and the jth neuron in thee subsequent layer, and S_{out} is the spike released from the neuron as a function of the membrane potential, U.

The weight update is then computed as:

$$W_{ij}^{t+1} = W_{ij}^{t} - \alpha \frac{\partial L}{\partial W_{ij}^{t}} \tag{3}$$

where W_{ij}^{t+1} is the updated weight, W_{ij}^{t} is the current weight, and α is the learning rate.

Direct application of the chain rule in spiking neuron layers is problematic because it ultimately prevents useful weight updates as described below:

$$\frac{\partial S_{out}}{\partial U} \in \{0, \infty\} \implies \frac{\partial L}{\partial W_{ij}^{t}} \in \{0, \infty\} \implies W_{ij}^{t+1} \in \{W_{ij,-\infty}^{t}\} \tag{4}$$