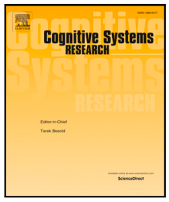
Cognitive Systems Research 88 (2024) 101277

Contents lists available at ScienceDirect

Cognitive Systems Research

journal homepage: www.elsevier.com/locate/cogsys



Hebbian spatial encoder with adaptive sparse connectivity

Petr Kuderov [a,b], Evgenii Dzhivelikian [a], Aleksandr I. Panov [a,b,c,*]

[a] Moscow Institute of Physics and Technology (MIPT), Dolgoprudny, Russia
[b] Artificial Intelligence Research Institute (AIRI), Moscow, Russia
[c] Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russia

ARTICLE INFO

Action editor: Alexei V. Samsonovich

Keywords:
Spatial encoding
Sparse distributed representations
Sparse neural networks
Adaptive neural connectivity

ABSTRACT

Biologically plausible neural networks have demonstrated efficiency in learning and recognizing patterns in data. This paper proposes a general online unsupervised algorithm for spatial data encoding using fast Hebbian learning. Inspired by the Hierarchical Temporal Memory (HTM) framework, we introduce the *SpatialEncoder* algorithm, which learns the spatial specialization of neurons' receptive fields through Hebbian plasticity and k-WTA (*k* winners take all) inhibition. A key component of our model is a two-part synaptogenesis algorithm that enables the network to maintain a sparse connection matrix while adapting to non-stationary input data distributions. In the MNIST digit classification task, our model outperforms the HTM SpatialPooler in terms of classification accuracy and encoding stability. Compared to another baseline, a two-layer artificial neural network (ANN), our model achieves competitive classification accuracy with fewer iterations required for convergence. The proposed model offers a promising direction for future research on sparse neural networks with adaptive neural connectivity.

1. Introduction

Data encoding is fundamental to any AI algorithm. A well-designed feature space can transform a computationally intensive task into a series of simple linear operations (Menache, Mannor, & Shimkin, 2005; Musavi, Ahmed, Chan, Faris, & Hummels, 1992). Supervised algorithms based on deep learning typically build features implicitly tailored to specific datasets. By integrating domain-specific knowledge into the processing system, it is possible to design more versatile feature spaces that are suitable for a variety of downstream tasks in an unsupervised manner. Examples of such versatile feature spaces include embeddings from language models (Su et al., 2019), convolutional filters (Hussain, Bird, & Faria, 2019), and latent spaces of autoencoders (Leeb, Bauer, Besserve, & Schölkopf, 2022), which can be applied in tasks ranging from classification (Keraghel, Morbieu, & Nadif, 2024) to world modeling for reinforcement learning algorithms (Matsuo et al., 2022). However, mainstream state-of-the-art deep learning methods for data representation are not always sample- and computationally efficient (Menghani, 2023), posing challenges for applications that require fully online learning, such as real-world robotics (Ibarz et al., 2021; Liu, Nageotte, Zanne, de Mathelin, & Dresp-Langley, 2021). Despite significant attention to this issue and various attempts to address it, current methods still struggle to optimally incorporate new information at test time (Dulac-Arnold, Mankowitz, & Hester, 2019). Therefore, exploring new learning and data representation frameworks remains crucial.

A more classical approach to data encoding involves the use of classifiers or autoencoders trained via error backpropagation. For instance, classifier-based encoding typically requires labeled data, providing meaningful features such as those found in Convolutional Networks (Zeiler & Fergus, 2014). Variational Autoencoders (VAEs) offer unsupervised learning that often results in interpretable feature spaces, making them effective for learning world models alongside RNN-like architectures (Hafner et al., 2019; Kingma & Welling, 2022). However, backpropagation-based models, despite their depth and data encoding power, often face challenges related to sample and computational inefficiencies, particularly in applications requiring test-time adaptation, such as Reinforcement Learning (Padakandla, 2022).

A promising alternative approach is found in biologically inspired learning algorithms and architectures (Hassabis, Kumaran, Summerfield, & Botvinick, 2017), which we investigate in this work. There is evidence that combining biologically inspired methods with established techniques can lead to more efficient and robust learning algorithms and architectures for AI (Amato, Carrara, Falchi, Gennaro, & Lagani, 2019; Ba, Hinton, Mnih, Leibo, & Ionescu, 2016; Iyer et al., 2022; Krithivasan, Sen, Venkataramani, & Raghunathan, 2022). One such model is the Hierarchical Temporal Memory (HTM) framework, which

* Corresponding author at: Artificial Intelligence Research Institute (AIRI), Moscow, Russia.
E-mail addresses: kuderov@airi.net (P. Kuderov), panov.ai@mipt.ru (A.I. Panov).

https://doi.org/10.1016/j.cogsys.2024.101277
Received 4 June 2024; Received in revised form 31 July 2024; Accepted 16 August 2024
Available online 22 August 2024

models pyramidal neurons with active dendrites in the cerebral cortex and hippocampus (Dzhivelikian, Latyshev, Kuderov, & Panov, 2022; Hawkins, Ahmad, & Cui, 2017). The HTM model is utilized in various applications, including anomaly detection and time-series predictive analytics (Cui, Ahmad, & Hawkins, 2016). The HTM Spatial Pooler is a neural network algorithm that encodes input stimulus patterns using the local Hebbian rule (Mnatzaganian, Fokoué, & Kudithipudi, 2017). Its primary function is to specialize the spatial aspects of neurons' receptive fields through k-WTA (k winners take all) inhibition (Oster, Douglas, & Liu, 2009). The output activity of the Spatial Pooler is a sparse binary vector, meaning that only a small fraction of neurons are active at any given time. This type of representation is referred to as Sparse Distributed Representation (SDR) (see Section 2).

Beyond HTM, other models like SoftHebb and Krotov have also explored the use of Hebbian learning and sparse representations. The SoftHebb model employs a softmax activation function instead of a strict k-WTA rule, allowing the network to specialize in inferring specific clusters of input data (Moraitis, Toichkin, Journé, Chua, & Guo, 2022). The authors show that their network can be viewed as generative model optimization, with theoretical guarantees on convergence and optimality if the dataset contains distinguishable clusters. SoftHebb's learning is based on the Oja rule (Oja, 1982) – a variant of Hebbian learning rule, – which forces weight vectors to approach a unit L^2 sphere during learning. The SoftHebb model is mainly contrasted with backpropagation algorithms, demonstrating that it can achieve competitive results as a spatial data encoder with significantly fewer iterations. In follow-up work (Journé, Rodriguez, Guo, & Moraitis, 2022), the authors extend their model to convolutional networks, showing that it benefits from deeper architectures. They also explore improvements to the learning mechanism, such as adaptive learning rates (based on the norm of weights) and anti-Hebbian learning. The SoftHebb model is also compared to hard WTA networks, as it can be considered a soft version of these networks.

Krotov et al. extended the Oja rule to accommodate arbitrary Lebesgue p-norms ($p \geq 2$), demonstrating benefits for encoding image data, especially in the context of image classification tasks (Krotov & Hopfield, 2019). Competition in the layer is simulated using global inhibition. While this method can be slow, as it requires several iterations to converge for each presented pattern, the authors also proposed a faster approximation of this competition using the k-WTA with anti-Hebbian plasticity. The key differences from the common k-WTA Hebbian learning are that only the strongest activation is reinforced, and also that the K-th strongest activation is inhibited via the anti-Hebbian rule to support stronger competitive specialization. Furthermore, the activation function is not a strict binary k-WTA but rather a soft counterpart — a non-linear Restricted Polynomial Unit (RePU).

Recent advancements in self-supervised learning and blockwise training offer valuable insights for neural network training, particularly in scenarios where backpropagation may not be feasible or desirable. Siddiqui et al. demonstrate that blockwise pretraining using methods like Barlow Twins can achieve comparable performance to end-to-end backpropagation on large datasets like ImageNet (Siddiqui, Krueger, LeCun, & Deny, 2023). This approach, which trains different network parts independently, supports the notion that intermediate layers can learn useful representations autonomously, aligning with the emphasis on local learning rules and reducing the need for a full backpropagation path.

A non-equilibrium memory approach for contrastive learning, updating synaptic weights without explicit memory storage or switching modes, is introduced in Falk, Strupp, Scellier, and Murugan (2023). This method, based on a dynamic feedback mechanism, offers a biologically plausible learning model that links physical dissipation costs to learning processes, highlighting the potential for energy-efficient, neuromorphic systems.

The use of predictive coding in associative memory models, as explored by Salvatori et al. provides a framework for storing and retrieving data even when inputs are incomplete or corrupted (Salvatori et al., 2021). This model, outperforming traditional autoencoders and modern Hopfield networks, demonstrates the effectiveness of gradient descent on a global energy function for error-driven learning, akin to processes in the visual cortex. Ororbia et al.'s Neural Generative Coding (NGC) framework introduces local prediction and parameter adjustment based on prediction errors, a concept inspired by predictive processing in the brain Ororbia and Kifer (2022). This framework addresses limitations of traditional backpropagation, such as the weight-transport problem, and shows promise in both generative and unsupervised learning tasks, suggesting a pathway toward more brain-like artificial neural systems.

In this paper, we further investigate biologically plausible neural networks with local learning rules, which have demonstrated the ability to learn and recognize patterns in data (Moraitis et al., 2022; O'Reilly, Russin, Zolfaghar, & Rohrlich, 2021). We propose a *SpatialEncoder* model, a general online unsupervised algorithm for spatial data encoding based on fast Hebbian learning (see Fig. 1). In summary, our SpatialEncoder model shares some conceptual foundations with the HTM Spatial Pooler, SoftHebb, and Krotov models, particularly in its use of Hebbian learning principles. However, it distinguishes itself in several key ways (see detailed discussion in Section 5):

- **Non-Binary Connections and SDRs**: Our model uses non-binary connections and Sparse Distributed Representation (SDR) encoding, contrasting with the binary nature of the HTM Spatial Pooler. This feature allows for a more fine-grained encoding scheme.
- **Sparse Connectivity**: Our model utilizes sparse connectivity, significantly reducing the number of active connections in the network. This property is crucial for efficient processing of high-dimensional data.
- **Linear Competition Mechanisms**: Unlike the strict binary competition mechanisms in the HTM Spatial Pooler and the softer but still strict mechanisms in the SoftHebb and Krotov models, our model employs much softer linear mechanisms for synaptic and neuronal competition. This approach is complemented by enforcing sparse connectivity, which inherently drives high competition among neurons. As a result, neuron clustering is supported by sparse connectivity, while intra-cluster competition remains soft, reducing sensitivity to learning regimes and initialization parameters.
- **Adaptability**: Our model features a dynamically adaptable sparse connection matrix, contrasting with the dense connectivity in the Krotov and SoftHebb models. This adaptability allows our model to efficiently handle non-stationary data distributions. Additionally, the explicit boosting mechanism, similar to that in the HTM Spatial Pooler but with dynamic strength depending on the synaptic weights' norm, prevents the occurrence of non-specialized "dead" neurons.
- **Anti-Hebbian Learning**: The incorporation of anti-Hebbian learning, akin to the approach used in the Krotov model, further enhances the competitive specialization among neurons, providing nuanced control over the learning dynamics.
- **Scalability and Efficiency**: By leveraging these mechanisms, our model achieves a balance between computational efficiency and robustness in learning, making it suitable for applications that require adaptive and efficient data encoding, such as real-time systems.

The main contributions of our work are as follows:

1. We propose a general online unsupervised algorithm for spatial data encoding using fast Hebbian learning.
2. We introduce a synaptogenesis algorithm that allows the network to maintain a sparse connection matrix and adapt to non-stationary input data distributions.

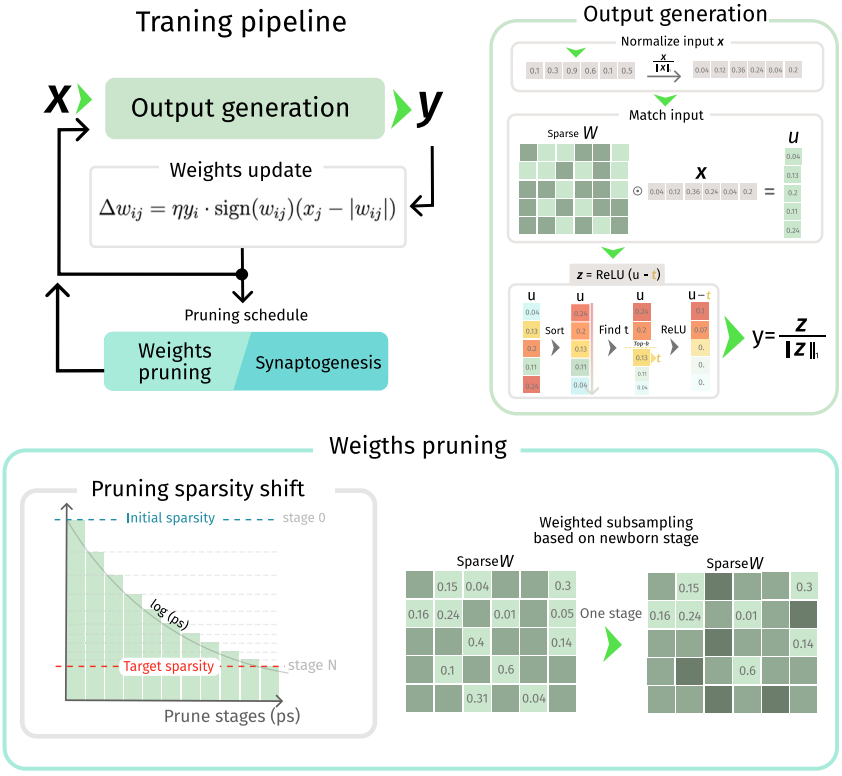


Fig. 1. Schematic representation of the SpatialEncoder algorithm. The network learns using a modified Hebb rule, supporting both excitatory and inhibitory weights, and ensuring neurons have weight vectors of unit L^1 norm. The SpatialEncoder's activation function combines k-WTA and ReLU with L^1 normalization. We employ weight pruning as part of synaptogenesis, which gradually sparsifies the connection matrix during the early phase of training. This process encourages high specialization of neurons while reducing the number of simultaneous competitors, thereby relaxing competition among them. Additional synaptogenesis mechanisms complement the pruning, allowing further adaptability of network connections.

In the following sections, we will provide a detailed explanation of our algorithm, discuss related work in the field, and present the background and methods used in our study.

2. Background

A key characteristic of our model is that all its components interact and function using Sparse Distributed Representations (SDRs). Evidence suggests that mammalian brains may employ a similar coding strategy (Graham & Field, 2007). SDRs offer several advantages in neural network models compared to the dense representations typically used in traditional artificial neural networks (ANNs):

- **Efficiency**: SDRs enable efficient use of computational resources, as only a small percentage of neurons are active at any given time.
- **Noise Resistance**: SDRs are inherently noise-tolerant. Even if some neurons are incorrectly activated or deactivated, the overall pattern can still be recognized due to the distributed nature of the representation.
- **Semantic Similarity**: SDRs preserve semantic similarity, meaning similar inputs produce similar outputs. This property is useful in tasks such as anomaly detection, where the goal is to identify inputs that deviate from known patterns, or in classification tasks, where the goal is to group similar inputs together.
- **Capacity and Robustness**: SDRs have a high exponential capacity for storing patterns and are robust to failures. The network can continue to function correctly even if some neurons are lost, thanks to the distributed nature of the representation.

Since our model operates in discrete time, SDRs are represented as sparse vectors corresponding to the discrete-time activity of neurons, with each vector element indicating the activity of a specific neuron in a layer. In HTM literature, SDRs are typically binary vectors, as this model uses binary activations. Our model, however, can work with both binary and real-valued activations. For clarity, we refer to binary SDRs as BinSDR and real-valued SDRs as Rate SDR (RateSDR), the latter reflecting the rate of neuron activations.

The Hebbian learning rule is a biological principle believed to underlie synaptic plasticity in the brain. It posits that if two neurons on either side of a synapse are activated simultaneously, the strength of the synapse between them increases. This can be mathematically expressed as:

$$\Delta w_{ij} = \eta x_j y_i, \tag{1}$$

where Δw_{ij} represents the change in the synaptic weight between the jth presynaptic neuron and the ith postsynaptic neuron, η is the learning rate, x_j is the activation of the presynaptic neuron, and y_i is the activation of the postsynaptic neuron.

The Hebbian-like rule is considered fundamental to learning and memory formation in the brain Rolls (2021). Unlike the backpropagation method commonly used in Artificial Neural Networks (ANNs), Hebbian learning allows for much faster adaptation in terms of learning iteration. It is inherently local, enabling fast parallel processing and does not impose additional memory requirements, which can be particularly advantageous for implementation on neuromorphic hardware.

However, there are several variations of the Hebbian learning rule. One of the most well-known is the Oja rule (Oja, 1982), which contains an additional term that forces the synaptic weights of each neuron to have unit L^2 norm. The Oja rule can be expressed as:

$$\Delta w_{ij} = \eta y_i (x_j - y_i w_{ij}). \tag{2}$$

Another popular variation of the Hebbian learning rule is known from works by Willshaw and von der Malsburg (Willshaw & Von Der Malsburg, 1976):

$$\Delta w_{ij} = \eta y_i (x_j - w_{ij}). \tag{3}$$

This rule utilizes heterosynaptic long-term depression (LTD) to maintain synaptic weights on each dendrite to be very similar. When complemented with input L^1 normalization, it also forces unit sum of the weights.

3. Methods

In this work, we propose SpatialEncoder, an online unsupervised algorithm for spatial data encoding based on a biologically plausible neural network model (see Fig. 1). SpatialEncoder functions as a clustering algorithm, converting input patterns into RateSDRs while preserving pairwise similarity among the inputs. The algorithm learns the spatial specialization of neurons' receptive fields using the local Hebbian rule and k-WTA (k winners take all) inhibition (Oster et al., 2009).

The SpatialEncoder maintains a sparse connection matrix W by storing the indices of existing connections, which define the neurons' receptive fields (RF). This approach allows us to store only the slice $W[\mathrm{RF}] = W_{ij} | (i,j) \in \mathrm{RF}$ instead of the entire matrix W. For clarity, we will use the dense version with the full matrix W in our calculations.

3.1. Connection matrix initialization

The connection matrix W is initialized dense, with random non-negative connections sampled from a zero-mean normal distribution (absolute values are taken to ensure non-negativity). The standard deviation is chosen based on the desired initial norm of the weights, following the method described in Journé et al. (2022).

Our model supports both excitatory and inhibitory connections; therefore, a small portion (5%–15%) of the connections are made inhibitory by switching their sign to negative. Compared to SoftHebb and Krotov models (discussed in Section 5), which also employ inhibitory connections, we adapt the learning rule to explicitly support and maintain the sign of connections. This approach allows us to control the balance between excitatory and inhibitory connections, and also make the inhibitory connections to support the competition among neurons.

Additionally, a portion (0%–40%) of connections can be safely zeroed out to help initial specialization of neurons. However, our experiments showed that it is not necessary.

3.2. Potentiation and activation

We define the potentiation of SpatialEncoder neurons as:

$$u_i = \beta_i W_i x, \tag{4}$$

where x is the input vector, W_i is a row vector representing the connection weights of the ith neuron, u_i is the value indicating the strength of the match between the input pattern and neuron i, and β_i is the boosting value for the ith neuron.

Interestingly, in Krotov and Hopfield (2019), potentiation utilizes non-linearity over weights $u_i = W_i^{p-1} x$ for $p \geq 2$. In similar fashion, we tested sublinear potentiation $u_i = \sqrt{W_i} x$ for our model, which also worked well in practice suggesting that even sublinear synaptic competition may be considered too.

The boosting term is an idea inherited from the HTM SpatialPooler algorithm. It defines an innate homeostatic plasticity mechanism that encourages inactive neurons to increase their sensitivity and find suitable specialization, thereby becoming more useful. Boosting depends on the neuron's output popularity, defined as relative output rate:

$$\mathrm{OP}_i = \frac{\text{i-th neuron avg output rate}}{\text{avg layer output rate}}, \tag{5}$$

which makes it a non-local measure. However, with input and output activity normalizations, it can be made entirely local. The boosting term β is calculated as following:

$$a = -\log(\mathrm{OP}_i), \tag{6}$$

$$q = \tanh(\frac{a}{\mu}), \tag{7}$$

$$\beta = B^q, \tag{8}$$

where B is the maximum strength constant for boosting and μ defines the boosting responsiveness. As q approaches to -1 for extremely active and to 1 for extremely inactive neurons, the boosting term β approaches to $\frac{1}{B}$ and to B, respectively. This regulates the neuron's responsiveness to input signals, promoting balanced activity across the network. In contrast to the HTM SpatialPooler, where a similar parameter to B is used as a fixed hyperparameter, our model employs a dynamically adapted boosting strength, $B = \log_2 \left(\sum_i |w_i|\right)$. This dynamic adaptation means that the effects of the boosting mechanism gradually decay during the learning process, as weights approach unit sphere.

The activation function is a k-WTA ReLU activation followed by L^1 normalization:

$$z_i = \begin{cases} u_i & \text{if } u_i > 0 \text{ and } i \in \mathrm{kWTA}(u) \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

$$\tilde{z}_i = z_i - \min(z_j | j \in \mathrm{kWTA}(u)), \tag{10}$$

$$y_i = \frac{\tilde{z}_i}{\sum_j |\tilde{z}_j|}, \tag{11}$$

where y_i is the resulting activation of the ith postsynaptic neuron. The function kWTA refers to a k-winners-take-all activation, returning the indices of the neurons with the highest matching values. In Eq. (9), we define the set of active neurons through k-WTA activation. In Eq. (10), the response of the active neurons is shifted relative to the minimum value among them, analogous to a threshold-shifted ReLU. Finally, the resulting values are L^1-normalized to obtain the output activations y_i in Eq. (11).

Our model supports both binary and real-valued activations. In the case of binary activations, we use the same activation function but set the output activations to 1 for all top K active neurons.

3.3. Learning

In our model, we use an analogy where each neuron has a fixed capacity to produce neurotransmitter receptors, which are distributed among its synaptic connections. Consequently, the sum of synaptic weights should tend to be constant or unit after normalization. This balance can be achieved through various forms of heterosynaptic plasticity. We employ the Willshaw rule (Eq. (3)), a variant of the Hebbian rule, to update the connection weights between neurons. The key modification in our application of the Willshaw rule is that it supports both excitatory and inhibitory connections:

$$\Delta w_{ij} = \eta y_i \cdot \mathrm{sign}(w_{ij})(x_j - |w_{ij}|). \tag{12}$$

This learning rule forces synaptic weights close to the simplex defined by L^1 norm of the input pattern, which in our model is also subject to L^1 normalization (Eq. (11)). As the result, the synaptic weight vector of each neuron converges to the unit absolute sum during the learning process.

To prove convergence of our learning rule to the unit L^1 sphere, we should take the derivative of a neuron weights' norm and substitute dw term with the learning rule:

$$\frac{d}{dt} \sum_i |w_i| = \sum_i \frac{d}{dt} |w_i| = \sum_i \mathrm{sign}(w_i) \frac{d}{dt} w_i \tag{13}$$

$$= \sum_i \eta y \cdot \mathrm{sign}^2(w_i)(x_i - |w_i|) \tag{14}$$

$$= \eta y (\sum_i x_i - \sum_i |w_i|). \tag{15}$$

where $\sum_i |w_i|$ is the L^1 norm of a single neuron's synaptic weights, y is the neuron's activation. As we see, each update an L^1 norm of weights tends to the sum of input if $y > 0$. Adding a non-negativity constraint to the input x_i and output y, and apply L^1 normalization to x, we can ensure the convergence of weights to the L^1 unit sphere.

To support better specialization and competition between neurons, we optionally apply anti-Hebbian learning similar to Journé et al. (2022), Krotov and Hopfield (2019) by introducing additional pair of hyperparameters M and α_{ah}, where M defines the number of neurons affected by anti-Hebbian learning and α_{ah} is the scale of the anti-Hebbian learning rate. Anti-Hebbian learning is applied to M least active neurons within the top K, that is the neurons having ranks from the $K - M + 1$ to the K. For them, we employ the same rule (Eq. (12)), but with the opposite sign of the learning rate and an additional scaling factor α_{ah}. As the result, the SpatialEncoder is enforced to have $K - M$ neurons specialized to any single pattern.

3.4. Newborn phase

The original HTM SpatialPooler algorithm has several drawbacks, including encoding instability caused by boosting. While boosting helps neurons specialize and increases overall adaptability, it can lead to unstable encoding. Additionally, the HTM SpatialPooler can suffer from slow processing on high-dimensional inputs, such as images. As a result, for models utilizing it to encode image inputs, the encoding overhead becomes significant relative to the overall model processing time. One approach to mitigating this issue is by setting a higher sparsity level for the Spatial Pooler's potential connection matrix. However, this can result in highly sparse random initialization, which may lead to suboptimal neuron specialization.

To address these issues, we introduce the "newborn stage", an idea broadly inspired by concepts proposed in Dobric, Pech, Ghita, and Wennekers (2022). The newborn stage in the SpatialEncoder occurs during the initial phase of the learning process, where rough specialization of neurons into pattern-matching clusters is expected. Initially, neuronal connections may have a relatively safe level of sparsity, but they become significantly more sparse during the newborn phase. In this stage, we gradually prune the majority of the weakest connections, resulting in neurons that are highly specialized due to their small receptive fields.

The final receptive field size is typically configured in relation to the average input pattern size, usually ranging from 25% to 200% of it, resulting in a connection sparsity of 0.1% to 10%. For instance, if binary input patterns have an average of 100 active bits out of 1000, we might set the target receptive field size to 25, which is 25% of the active input size and corresponds to a 2.5% connection matrix sparsity. Consequently, the instability of the spatial pooler – and thus its adaptability – is further constrained in the adult stage.

The newborn stage is divided into several identical pruning steps. During each step, we sample which connections should survive pruning, using L^2-normalized weights to determine the probabilities in the sampling process. There are two supported strategies for the newborn stage – linear and power-law – that define how the number of connections decays with each pruning step. In our experiments, we employ the power-law strategy, as it prunes the majority of weak connections during the early steps, thereby accelerating the experiments.

In summary, the newborn stage serves as the initial synaptogenesis mechanism that promotes neural specialization during the early stages of learning. For many offline tasks, this mechanism may suffice to ensure satisfactory encoding quality. However, in the context of online learning, an additional synaptogenesis mechanism is necessary to maintain the useful specialization of neurons over time.

3.5. Synaptogenesis

For the SpatialEncoder, we track several activation metrics in both fast and slow manners: input and output activation rates. These metrics are aggregated using an exponential moving average. The Input Rate (IR) measures the average activation rate of each presynaptic neuron over time, while the Output Rate (OR) measures the same for the postsynaptic neurons.

Synaptogenesis in our model is divided into two main actions: synaptogenesis score recalculation and synaptogenesis event application. The score recalculation occurs periodically at a constant rate, determined by a specific hyperparameter. The application of synaptogenesis events, on the other hand, occurs probabilistically after each processing timestep.

Given the current fast Input Rate (IR) and Output Rate (OR), we first normalize these rates with their corresponding average values to make them relative. These normalized values are referred to as Input Popularity (IP) and Output Popularity (OP), as they represent the relative frequency of activity for each presynaptic and postsynaptic neuron, respectively:

$$\mathrm{IP}_i = \frac{\mathrm{IR}_i}{\mathrm{avg(IR)}}, \tag{16}$$

$$\mathrm{OP}_i = \frac{\mathrm{OR}_i}{\mathrm{avg(OR)}}, \tag{17}$$

Next, we define Receptive Field Efficiency for matching input ($\mathrm{RFE}^{\mathrm{in}}$) and its normalized version:

$$\mathrm{RFE}^{\mathrm{in}} = W \cdot \mathrm{IP}, \tag{18}$$

$$\mathrm{NRFE}^{\mathrm{in}} = \frac{\mathrm{RFE}^{\mathrm{in}}}{\mathrm{avg}(\mathrm{RFE}^{\mathrm{in}})}, \tag{19}$$

and Receptive Field Efficiency for activating neuron ($\mathrm{RFE}^{\mathrm{out}}$) and its normalized version:

$$\mathrm{RFE}^{\mathrm{out}} = \frac{\mathrm{OP}}{\mathrm{NRFE}^{\mathrm{in}}}, \tag{20}$$

$$\mathrm{NRFE}^{\mathrm{out}} = \frac{\mathrm{RFE}^{\mathrm{out}}}{\mathrm{avg}(\mathrm{RFE}^{\mathrm{out}})}. \tag{21}$$

$\mathrm{RFE}^{\mathrm{in}}$ measures how well each neuron's receptive field is tuned to the input distribution, while $\mathrm{RFE}^{\mathrm{out}}$ assesses how effectively each neuron's receptive field tuning translates into the neuron's activation. The normalized versions of these metrics facilitate calculations by maintaining a more stable range. A value of $\mathrm{NRFE}^{\mathrm{in}} \ll 1$ indicates that the neuron has a poorly tuned receptive field relative to the input distribution (for example, due to unsuccessful initialization or changes in the input distribution that make this part of the input weak). Conversely, $\mathrm{NRFE}^{\mathrm{out}} \ll 1$ suggests that the neuron's receptive field is not competitive enough to activate the neuron.

Both $\mathrm{NRFE}^{\mathrm{in}}$ and $\mathrm{NRFE}^{\mathrm{out}}$ values contribute to defining the synaptogenesis score SS of a neuron, using the higher of the two. This score is then transformed into the probability of a synaptogenesis event for a neuron, with the probability being clipped to ensure that non-zero probabilities are assigned only to highly underperforming neurons:

$$p_i = \mathrm{clip}[\frac{-\log(\mathrm{SS}_i) - \mathrm{low}}{\mathrm{high} - \mathrm{low}}, 0.001, 1.0], \tag{22}$$

where low and high are hyperparameters defining the range for the synaptogenesis score to grow log-linearly. In our experiments, low = $\log(1.5)$ and high = $\log(20)$.

After calculating the synaptogenesis event probability for each neuron, we utilize it at each step when learning is enabled. We combine p_i with the current relative potential $\frac{u_i}{\mathrm{avg}(u_i)}$ for the neurons with non-negative potential u_i to get the probability of a synaptogenesis event at the current timestep $\hat{p}_i = p_i \frac{u_i}{\mathrm{avg}(u_i)}$. Finally, we sample from a Binomial distribution for each neuron to determine which ones can participate (if any), and select the neuron with the maximum probability $\hat{p}$ among the winners.

Once the winner neuron for the synaptogenesis event is selected, we randomly sample from non-zero presynaptic neurons to determine where a new connection should be established. The sign of the synapse is sampled with the same probabilities as for the initial weights distribution. The weakest existing connection of the neuron is then replaced with the new one, and the probability of the synaptogenesis event is reset to zero.

4. Experiments

We tested our SpatialEncoder model implementation on both classification and regression tasks. In all tests, our model successfully learned to encode images for use with a linear classifier or regressor. For the classification tasks, we selected the MNIST handwritten digits and CIFAR-10 datasets, while for the regression task, we used the outdoor DVS camera dataset.

In our experiments, we compared our model with several unsupervised spatial encoding baselines, including the HTM SpatialPooler (Mnatzaganian et al., 2017), SoftHebb (Moraitis et al., 2022), and Krotov (Krotov & Hopfield, 2019) models. Additionally, we compared the performance with that of a two-layer MLP used as a supervised encoder.

For our experiments, we did not specifically optimize hyperparameters for each model and dataset. Instead, we used fixed hyperparameters for each baseline, unless explicitly stated otherwise. This approach allowed us to assess the models' sensitivity to hyperparameters and compare their general performance. For the SoftHebb and Krotov models, we used the same hyperparameters as specified in the original papers. For the HTM SpatialPooler, we used the default parameters from the Numenta implementation.

4.1. MNIST digits classification

To test the MNIST handwritten digits, all images were normalized to values in the range $[0, 1]$. For binary networks, we further binarized the images using a binary threshold equal to the mean activation value in each frame. We followed an evaluation protocol similar to that used in Moraitis et al. (2022). In our experiments, the SpatialEncoder model consisted of 2000 neurons with an activation size of 25 neurons. As SpatialEncoder converges faster, we trained the network for 30 epochs, compared to the 100 epochs used in Moraitis et al. (2022). Each epoch, the network was presented with the training set of 60,000 randomly ordered digit images. After each training epoch, we evaluated the network using the test set containing 10,000 images.

For the classification of images, unsupervised encoders were evaluated by training a linear classifier on the encoded representations. After each training epoch, the encoder was frozen, and a linear classifier was trained on top of it for 30 epochs from scratch. For the MLP baseline, we used a fully connected network with a single hidden layer using SiLU activation and a linear output layer. Both the MLP and linear classifier networks were trained using backpropagation with cross-entropy loss, the Adam optimizer with a learning rate of 0.003, and a batch size of 64.

Fig. 2 shows the classification accuracy on the test dataset. The reported metrics are averaged over 5 runs with different random seeds, and the shaded area represents the standard deviation.

Fig. 2 also shows the performance for two versions of the two-layer MLP, with 2000 neurons (`ANN 2k`) and 125 neurons (`ANN`) in the hidden layer. The MLP with 2000 neurons matches the hidden size of the unsupervised encoders. However, given the connection sparsity of the best-performing SpatialEncoder, it is appropriate to also evaluate an MLP with a size that matches the total number of SpatialEncoder's weights (approximately 100k). This corresponds to an MLP with 125 neurons in the hidden layer. Both MLP versions outperform the SpatialEncoder in terms of achieved accuracy. However, the SpatialEncoder converges within a single epoch and exhibits the best performance during the first 5 epochs. This suggests that the SpatialEncoder with rate activations can be effectively used as a fast online learning algorithm for spatial data encoding.

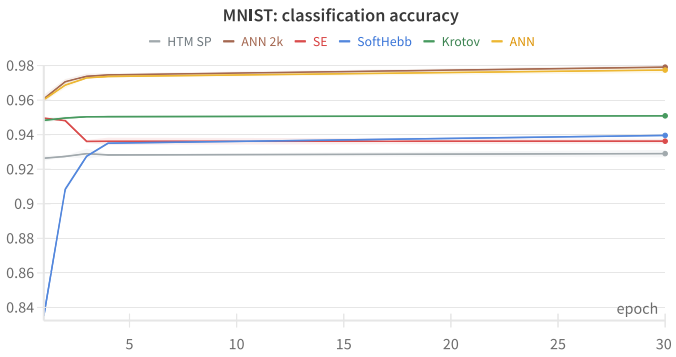


Fig. 2. Classification performance on the MNIST dataset.

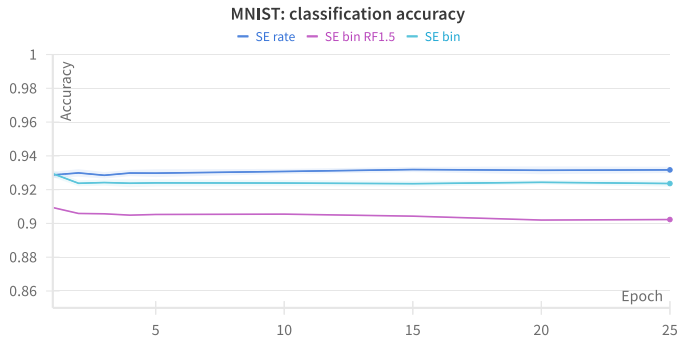
We also compared the performance of the SpatialEncoder with different receptive field sizes (see Fig. 3(a)). The best accuracy of 93% was achieved with a receptive field size (RF) of 0.3, which corresponds to 30% of the input pattern size. For the MNIST dataset, with an average active pattern size of 133, this means the receptive field size is 40, resulting in a connection sparsity of 5%. The results for the SpatialEncoder with `RF=0.3` for different output types – `SE bin` for binary SDRs and `SE rate` for rate SDRs – also highlight the benefits of using rate sparse encoding over binary encoding, as it provides richer information in the activation space.

We also examined the performance of the binary SpatialEncoder with a receptive field size (RF) of 1.5 to assess the implications of using a larger receptive field size. The results in Fig. 3(a) indicate that a larger receptive field size is not beneficial for this task. It is important to note that performance distribution typically exhibits two modes, depending on whether RF is greater or less than 1. The choice of RF depends on whether the input data possesses pronounced features. While a larger receptive field size does not prevent the network from learning sub-pattern features, having RF less than 1 provides a regularization effect, promoting feature-wise competition and specialization rather than pattern-wise specialization in k-WTA networks. For the MNIST dataset, which features distinctive handwritten digit characteristics, the SpatialEncoder with highly sparse connectivity ($RF < 1$) performs well. Testing the HTM SpatialPooler with varying potential sparsity levels corroborated our findings with the SpatialEncoder, showing that high sparsity benefits the SpatialPooler as well.

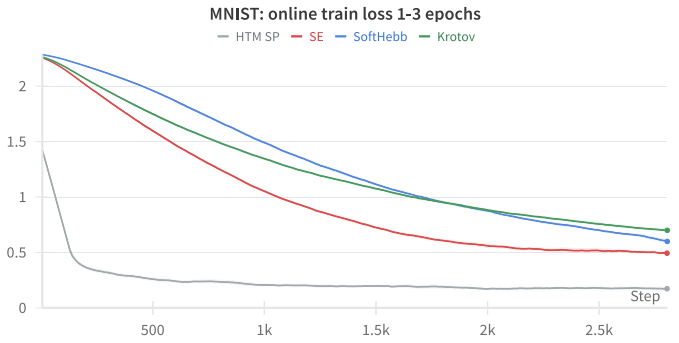
To evaluate the ease with which a linear classifier can learn data encoded by different unsupervised encoders, we trained an additional classifier fully online during the first three epochs and plotted the training losses. As shown in Fig. 3(b), the HTM SpatialPooler converges significantly faster than the other encoders, although it does not achieve the highest classification accuracy. In all our experiments, we observed a similar pattern: the HTM SpatialPooler exhibits faster convergence, while the other three encoders have comparable convergence speeds. We found no significant correlation between the online training loss curves and the final classification accuracy, although the majority of the classification accuracy is attained within the first three epochs.

4.2. CIFAR-10 classification

To test our model on the CIFAR-10 dataset, we employed the same protocol as with MNIST. For easier computational handling, we additionally transformed the RGB images to grayscale. Consequently, the results for the SoftHebb and Krotov baselines are lower than those reported in their respective papers (Journé et al., 2022; Krotov & Hopfield, 2019). Selective experiments with the full-RGB dataset showed an

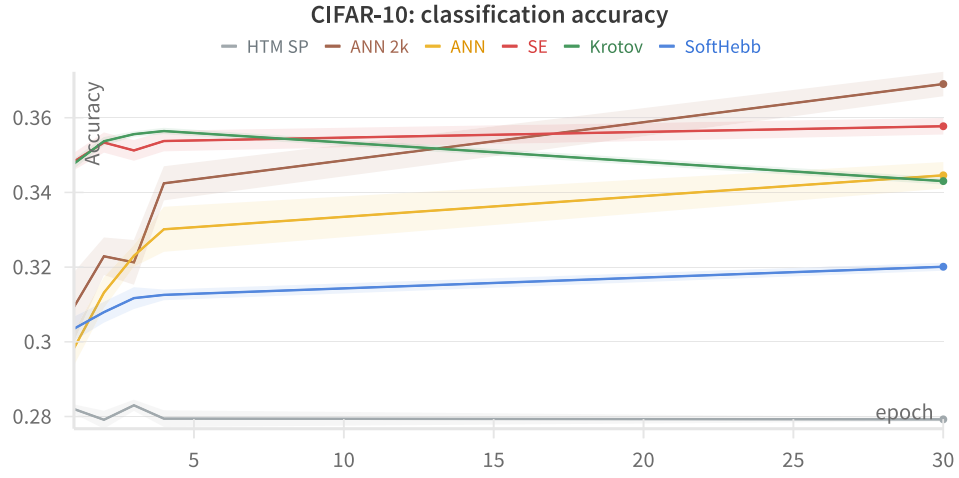


(a) Classification accuracy.

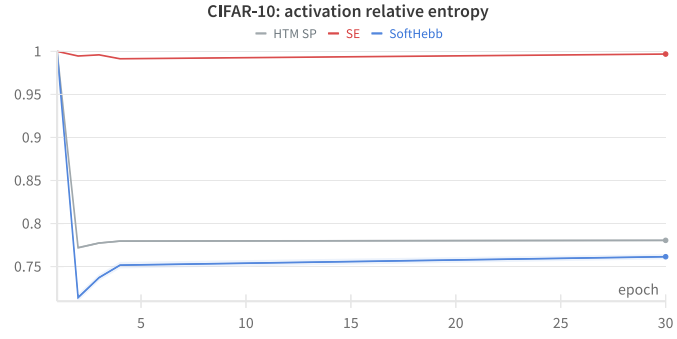


(b) Online training loss, epochs #1-3.

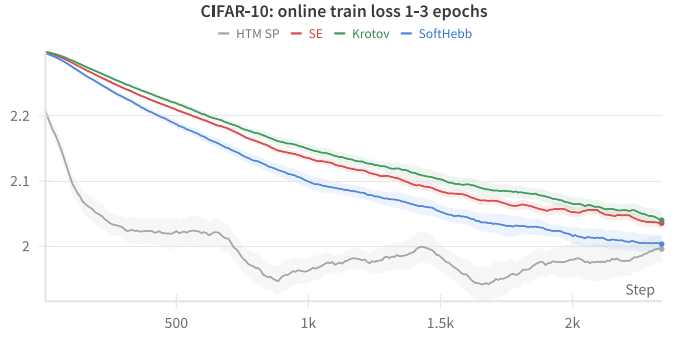
Fig. 3. (a) Performance of the SpatialEncoder (SE) on the MNIST classification task, depending on output binarization and receptive field size. (b) Training losses for the first three epochs of online training of a linear classifier on top of the unsupervised encoders for the MNIST classification experiment.



(a) Classification accuracy.



(b) Relative entropy.



(c) Train classification loss, online, 1-3 epochs.

Fig. 4. Classification performance on the CIFAR-10 dataset.

approximately 6%–8% increase in classification accuracy. The dataset was randomly split into 50,000 training images and 10,000 test images. Fig. 4 shows the classification accuracy on the test dataset and the relative entropy with online loss during training. The reported metrics are averaged over 5 runs with different random seeds, with the shaded area representing the standard deviation.

As shown in Fig. 4(a), the SpatialEncoder outperforms the other unsupervised encoders, achieving the highest classification accuracy. While the Krotov model initially shows comparable performance, its accuracy decays over time. This is consistent with the findings in (Krotov & Hopfield, 2019), where the Krotov model exhibited an early decline in performance before beginning to recover. In our experiments, the SpatialEncoder occasionally displayed a similar pattern, but the decline was less pronounced. The SoftHebb model performs poorly on this task.

The results in Fig. 4(b) show the relative entropy of the activation patterns for the baselines. Relative entropy measures neuron utilization in the layer and is calculated as the ratio of the entropy of activation rates in a layer to the maximum entropy of uniform activations, ranging from 0 to 1. Lower relative entropy values suggest that some neurons may dominate the activation patterns, while higher values indicate

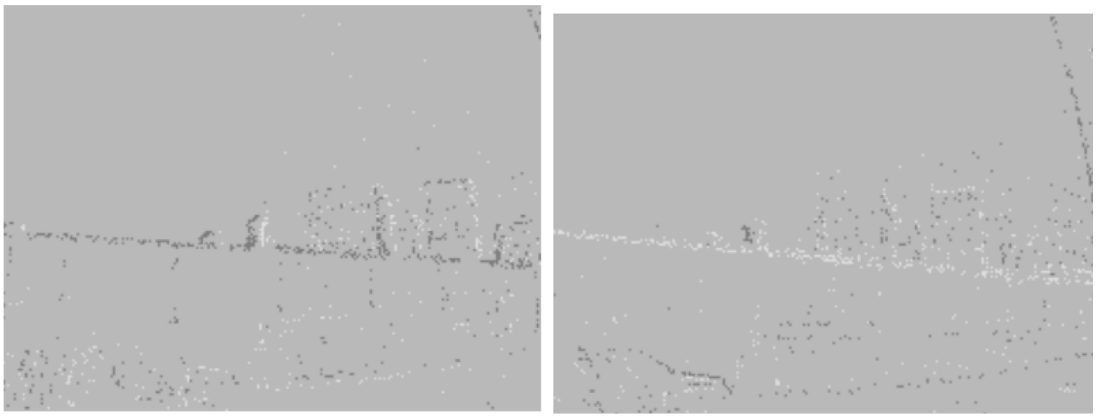


Fig. 5. Example of aggregated DVS camera frames from an outdoor walking dataset.

that the neurons are highly specialized with more distinct activation patterns. The results show that the SpatialEncoder has the highest relative entropy, indicating that it produces the most specialized activation patterns and effectively prevents the occurrence of inactive "dead neurons."

The online training loss for the first three epochs is shown in Fig. 4(c). We observed a pattern similar to that seen with the MNIST dataset, where the HTM SpatialPooler converges significantly faster than the other encoders, while the other three encoders have comparable convergence speeds.

4.3. DVS pose estimation

To evaluate our model on a regression task, we used a dataset containing outdoor walking scenes recorded with an event-based Dynamic Vision Sensor (DVS) camera (Mueggler, Rebecq, Gallego, Delbruck, & Scaramuzza, 2017), which closely simulates the working regime we aim for with the SpatialEncoder (see examples in Fig. 5).

We preprocessed the data to transform the stream of camera events into a RateSDR dataset. This involved aggregating events into 1.0 ms frames, downsampling the resulting images from 240×180 to 40×30, and averaging out the polarity for each pixel. For each image, we formed two RateSDR channels corresponding to each polarity sign. First, we created binary SDRs using threshold binarization, and then used the actual rate values for the SDRs to generate the RateSDRs. Finally, we concatenated the two channels into a single flattened RateSDR vector.

The results in Fig. 6(a) show the mean squared error (MSE) of the pose estimate on the test dataset, while Fig. 6(b) presents the online training MSE loss for the first three epochs. As shown, the SpatialEncoder outperforms the other unsupervised encoders, achieving the lowest MSE, with the Krotov model being the closest competitor. Interestingly, the SoftHebb model's performance on this task is even worse than that of the HTM SpatialPooler binary model, for which we have not found an explanation.

4.4. Equal weight tests for encoders

Finally, we repeated the tests on the CIFAR-10 and DVS datasets to compare the performance of the SpatialEncoder with the HTM SpatialPooler, SoftHebb, and Krotov models, ensuring a fair comparison by using the same number of weights for all models. For the SoftHebb and Krotov models, this involved reducing the number of neurons to 125, while for the HTM SpatialPooler, we increased the potential sparsity to 0.05.

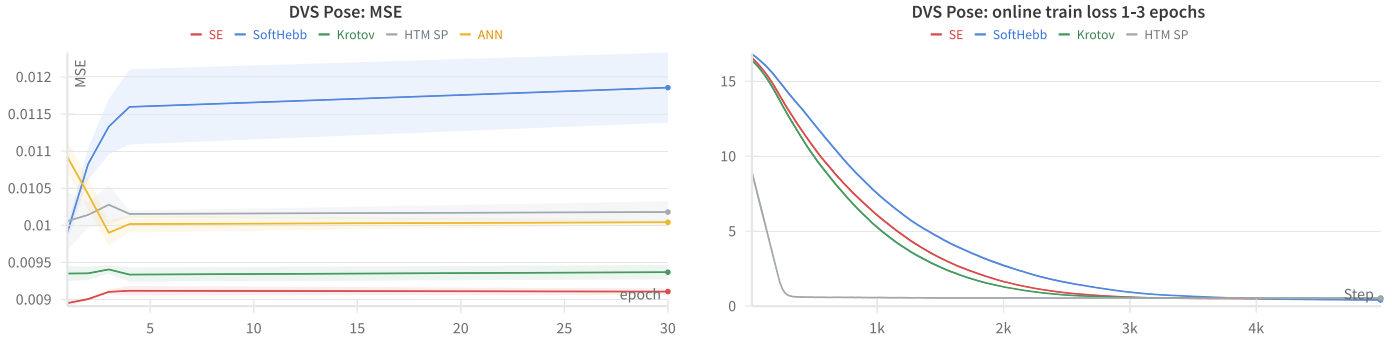
The results presented in Figs. 7 and 8 show that on the CIFAR-10 dataset, the SpatialEncoder outperforms all baseline models. As expected, reduced-size models exhibit significantly degraded performance compared to the full-size models. Interestingly, on the DVS dataset, size reduction does not affect the models' performance, with the Krotov model even slightly outperforming the SpatialEncoder. We believe this is due to the DVS dataset being more challenging, requiring more hyperparameter fine-tuning and slower learning. Consequently, all unsupervised models tend to show better performance before the learning process fully starts. Since the Krotov model learns much more slowly than the others (as each learning step affects only two neurons), it ultimately achieves better performance.

5. Comparative overview of key features with related models

Our SpatialEncoder model draws inspiration from the HTM framework, particularly the HTM Spatial Pooler (Mnatzaganian et al., 2017), but introduces significant innovations. While the HTM Spatial Pooler uses binary connections and representations, our model employs non-binary connections and representations to enhance its ability to represent patterns in high-dimensional spaces.

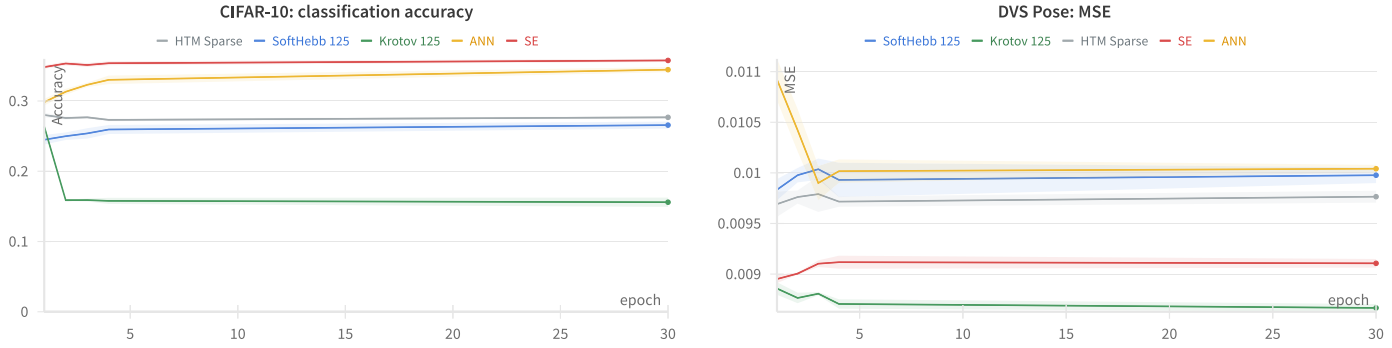
In the HTM Spatial Pooler, the connection matrix is divided into two types: potential and active connections. Potential connections are randomly initialized and define the potential connectivity of each neuron, remaining fixed throughout the learning process. Active connections, on the other hand, define the actual connectivity of the network used for inference. During learning, both types of connections are updated based on input patterns, allowing potential connections to become active or inactive. While this approach maintains sparse active connectivity and allows for adaptation to non-stationary data, it poses challenges in balancing adaptability, stability, and memory requirements. If the potential connectivity is too dense, the network requires significant additional memory to store these connections and may become unstable due to the boosting mechanism (Dobric et al., 2022). Conversely, if the potential connections are too sparse, the network's performance is largely constrained by the initialization, potentially leading to suboptimal results. Our model addresses these issues with a two-part synaptogenesis algorithm, enabling a sparse connection matrix that adapts to non-stationary input distributions, providing a more balanced approach to adaptability, stability, and memory requirements.

The SoftHebb and Krotov models are closely related to our approach. The SoftHebb model (Journé et al., 2022; Moraitis et al., 2022) replaces the strict k-WTA rule with a softmax activation function, resulting in computations that resemble Bayesian inference. This model is grounded in the Oja rule, which promotes weight vectors toward a unit L^2 sphere during learning. Krotov et al.'s model (Krotov & Hopfield, 2019) extends the Oja rule to higher p-norms (with $p = 3.5$ in the original work), introducing global inhibition to simulate



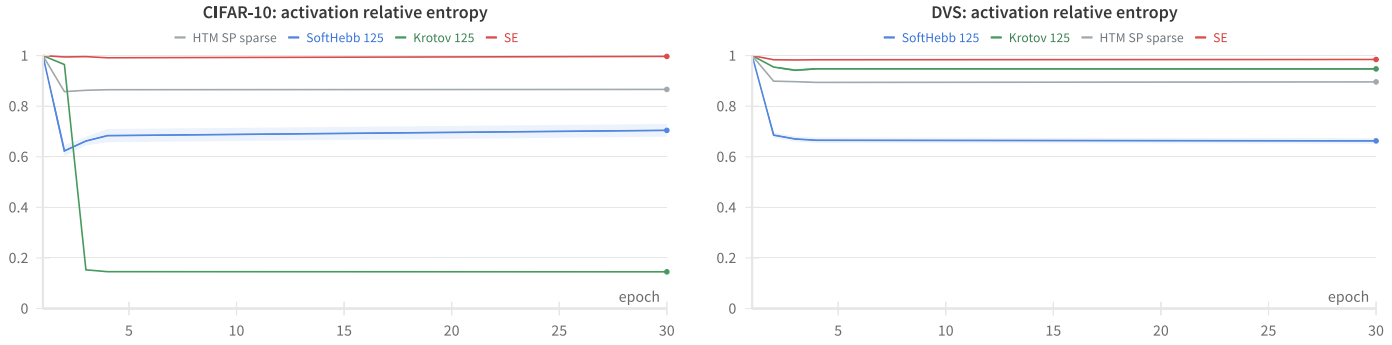
(a) Pose estimate MSE (lower better). **(b)** Train MSE loss, online, 1-3 epochs.

Fig. 6. Pose estimation performance on the DVS outdoor walking dataset.



(a) CIFAR-10 accuracy (higher is better). **(b)** Pose estimation MSE (lower is better).

Fig. 7. Performance on CIFAR-10 classification and DVS pose estimation tasks compared to the baselines, using a similar number of weights for each model.



(a) CIFAR-10 **(b)** DVS

Fig. 8. Activation relative entropy for the baselines on CIFAR-10 classification and DVS pose estimation tasks, using a similar number of weights for each model.

competition. Both models use strict activation functions (e.g., low-temperature softmax for SoftHebb and a polynomial with $n = 4.5$ for the Krotov model) and hypothesize that increasing strictness – while not approaching k-WTA extremity – enhances pattern separability. While effective, these approaches can be sensitive to the learning regime, initialization, and hyperparameters, potentially resulting in suboptimal performance and may not be ideal for generating rich, distributed k-WTA representations in high-dimensional spaces.

The SpatialEncoder takes a different approach by splitting both synaptic and neuronal competitions into two parts: extra- and intra-cluster. While extra-cluster competition remains strict via sparse connectivity, intra-cluster competition is largely softened to be linear. Linear synaptic competition is inspired by the concept of a fixed total number of receptors distributed among a neuron's dendrites, maintaining the sum of synaptic weights. Analogously to the biological neurons, neurons in our model start with an excess number of connections and receptors. During maturation, due to learning, unused connections are pruned, and the total number of receptors converges. With linear competition among the remaining synapses, we avoid strong synaptic dominance and maintain sensitivity across synapses, a common issue with higher p-norms. For instance, with higher p-norms, balancing competition among "active" synapses becomes challenging. In the extreme case, when a neuron repeatedly tunes to a specific pattern, high enough p-norms may cause the synaptic weight vector to converge to a one-hot encoding, matching a single presynaptic neuron. This results in minimal information transmission about the input and high sensitivity to noise. In contrast, our model allows synaptic weights to

mirror the pattern, maintaining sensitivity to patterns within a broad range. The piecewise-linear activation function was similarly chosen to ease competition within clusters of active neurons, promoting balanced specialization. The idea of linearizing strict activation functions is not new; for example, Martins and Astudillo (2016) proposes a piecewise linear variant of softmax for high-dimensional classification tasks to address similar issues. Additionally, the incorporation of anti-Hebbian learning, similar to Krotov's approach, enhances competitive specialization among neurons, providing nuanced control over learning dynamics.

There is also a noteworthy analogy between our model and the SoftHebb and Krotov models. In Journé et al. (2022), the authors of SoftHebb explore the implications of the initial weights' norm, concluding that it should be significantly greater than 1 (ranging from 5 to 10). This observation parallels the role of boosting in our model. The norm-converging property in all three models limits the "radius" of neuron specialization, meaning that each learning step not only reinforces a neuron's specialization to a specific pattern but also inhibits its specialization to other patterns. This allows space for other neurons to specialize. Since inactive neurons have their norms unchanged, a high initial norm increases the likelihood that even these neurons can successfully participate at some point (when others have relatively lower norms) and find their own specialization. This is similar to the effect provided by boosting in our model. Therefore, the initial norm in these models needs to be sufficiently high to support the specialization of as many neurons as possible. Otherwise, excessive competition may lead to a large number of highly suppressed, non-specialized "dead" neurons. In our model, this process is further supported by the explicit boosting mechanism, making the initial norm of the weights less critical compared to the SoftHebb and Krotov models.

Another key difference between the SpatialEncoder and the SoftHebb and Krotov models is the explicit handling of negative weights. In our model, connections maintain their assigned sign throughout the learning process, as specified in our learning rule (see Eq. (12)). This differs from the SoftHebb and Krotov models, where negative weights can turn positive through Hebbian learning and positive weights can turn negative through anti-Hebbian learning. During our experiments with these models, we noticed that anti-Hebbian learning could cause instability, requiring careful adjustment of the anti-Hebbian scale parameter. This instability often showed up as the divergence of weights' absolute values toward infinity. In contrast, our model did not exhibit these issues, potentially due to its consistent handling of negative weights.

In summary, while our model shares foundational concepts with HTM Spatial Pooler, SoftHebb, and Krotov models, it introduces unique features like non-binary SDRs, sparse connectivity, and softer competition mechanisms. These innovations make the SpatialEncoder model particularly suited for applications requiring efficient and adaptive data encoding.

6. Conclusion and discussion

In this work, we proposed an online unsupervised algorithm for spatial data encoding based on fast Hebbian learning, called SpatialEncoder. Inspired by the HTM model framework, SpatialEncoder aims to achieve spatial specialization of neurons' receptive fields through Hebbian plasticity and k-WTA inhibition. A key feature of our model is the two-part synaptogenesis algorithm, which allows the network to maintain a sparse connection matrix and adapt to non-stationary input data distributions. The model also shares similarities with the SoftHebb and Krotov models, using similar innate weight normalization through a modified Hebbian learning rule, but employing more linear normalization and activation functions. We hypothesize that, combined with sparse connectivity, our model serves as a locally linearized approximation of these models, providing granular encoding of the input while maintaining sufficient pattern separability.

We demonstrated through MNIST and CIFAR-10 classification tasks that our model outperforms the HTM SpatialPooler, owing to its richer rate-based encoding and enhanced ability to establish sparser connections. Compared to a two-layer ANN baseline, our model achieves competitive classification accuracy while requiring fewer iterations for convergence when matched for the total number of weights. The SpatialEncoder also shows comparable results with the SoftHebb and Krotov models, although further analysis is needed to fully understand the differences in learning behavior between these models and ours.

The SpatialEncoder model's design, leveraging sparse distributed representations and adaptable synaptogenesis mechanisms, offers significant potential for real-world applications that require efficient and adaptive data encoding. One of the key practical implications is its suitability for online learning environments, such as robotics and autonomous systems, where the ability to adapt to non-stationary data in real-time is crucial. The model's fast convergence and capacity to maintain rich, rate-based encoding make it an attractive choice for systems needing to quickly respond to changing conditions and inputs. Additionally, the model's sparse connectivity reduces computational and memory overhead, enabling deployment in resource-constrained settings such as edge devices or low-power hardware. This makes the SpatialEncoder particularly useful in applications like real-time video processing, sensor data analysis, and other domains where efficient and adaptive processing of high-dimensional data is essential. The experiments on the DVS camera dataset further demonstrate the model's potential for such applications.

6.1. Limitations and future work

However, the SpatialEncoder model has several limitations that could impact its performance and applicability.

Scalability and Computational Efficiency: The use of a large number of neurons (1000–4000) can challenge scalability, especially with large datasets or high-dimensional input spaces. The initial dense connection matrix requires time to prune connections and benefit from sparse connectivity, limiting early processing performance during the newborn stage. This phase incurs significant memory and computational overhead. Although initializing with a sparse connection matrix could mitigate these issues, it slows convergence due to the synaptogenesis mechanism requiring some pre-specialization. Future improvements could include a dual matrix system akin to the HTM SpatialPooler's active/potential connections approach, where potential connections are tracked separately and activated as needed, reducing computational overhead.

Hyperparameter Sensitivity: The model relies on several hyperparameters, including the number of neurons, output sparsity, receptive field size, and synaptogenesis settings. While defaults are provided to cover a broad range of tasks, some critical hyperparameters, particularly receptive field size and k for k-WTA activation, require careful tuning. The balance between competition and specialization among neurons hinges on these settings. The synaptogenesis cycle duration is also crucial; too short a cycle can cause instability, while too long a cycle reduces adaptability to non-stationary data.

Training Complexity and Hardware Utilization: Incorporating biologically inspired components like Hebbian and anti-Hebbian learning, k-WTA activation, and sparse connectivity adds complexity to the training process. This complexity makes it challenging to leverage modern processing accelerators like GPUs and TPUs, as the model's online learning approach favors sequential processing over parallel batch computations. However, for offline learning, the model can be adapted to support batch processing.

Comparative Performance and Understanding: Although the SpatialEncoder shows competitive performance, its specific advantages over other state-of-the-art models, particularly in efficiency and accuracy, need more exploration. The benefits of linear competition mechanisms and sparse connectivity are not fully understood and may

vary across more complex datasets and tasks. Future work could explore adaptive activation functions and sparse connectivity to balance pattern separability and computational efficiency. However, the success of linear activation functions in various domains suggests that they may suffice for many tasks (Agarap, 2019; Razzhigaev et al., 2024; Schlag, Irie, & Schmidhuber, 2021; Yue et al., 2024). The same applies to connection sparsity, which has shown promise in improving neural network efficiency (Gale, Elsen, & Hooker, 2019; Hoefler, Alistarh, Ben-Nun, Dryden, & Peste, 2021).

In summary, while the SpatialEncoder offers a promising approach to spatial data encoding, addressing these limitations will be crucial for broader applicability and practical deployment. Future research could focus on enhancing training methods, optimizing hyperparameter tuning, and improving the model's adaptability to non-stationary environments. Further exploration of the trade-offs associated with linear competition mechanisms and sparse connectivity could also provide valuable insights, positioning the SpatialEncoder for a wider range of applications.

CRediT authorship contribution statement

Petr Kuderov: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation. **Evgenii Dzhivelikian:** Validation, Data curation. **Aleksandr I. Panov:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Aleksandr Panov reports financial support was provided by Analytical Center for the Government of the Russian Federation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the Ministry of Science and Higher Education of the Russian Federation under Project 075-15-2024-544.

References

Agarap, A. F. (2019). Deep learning using rectified linear units (ReLU). arXiv:1803.08375, https://arxiv.org/abs/1803.08375.

Amato, G., Carrara, F., Falchi, F., Gennaro, C., & Lagani, G. (2019). Hebbian learning meets deep convolutional neural networks. In E. Ricci, S. Rota Bulò, C. Snoek, O. Lanz, S. Messelodi, & N. Sebe (Eds.), *Image analysis and processing – ICIAP 2019* (pp. 324–334). Cham: Springer International Publishing.

Ba, J., Hinton, G., Mnih, V., Leibo, J. Z., & Ionescu, C. (2016). Using fast weights to attend to the recent past. arXiv:1610.06258.

Cui, Y., Ahmad, S., & Hawkins, J. (2016). Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, *28*(11), 2474–2504. http://dx.doi.org/10.1162/NECO_a_00893.

Dobric, D., Pech, A., Ghita, B., & Wennekers, T. (2022). On the importance of the newborn stage when learning patterns with the spatial pooler. *SN Computer Science*, *3*(2), 179. http://dx.doi.org/10.1007/s42979-022-01066-4.

Dulac-Arnold, G., Mankowitz, D., & Hester, T. (2019). Challenges of real-world reinforcement learning. arXiv:1904.12901.

Dzhivelikian, E., Latyshev, A., Kuderov, P., & Panov, A. I. (2022). Hierarchical intrinsically motivated agent planning behavior with dreaming in grid environments. *Brain Informatics*, *9*(1), 8. http://dx.doi.org/10.1186/s40708-022-00156-6.

Falk, M., Strupp, A., Scellier, B., & Murugan, A. (2023). Contrastive learning through non-equilibrium memory. arXiv preprint arXiv:2312.17723.

Gale, T., Elsen, E., & Hooker, S. (2019). The state of sparsity in deep neural networks. arXiv preprint arXiv:1902.09574.

Graham, D., & Field, D. (2007). Sparse coding in the neocortex. *Evolution of Nervous Systems*, *3*, http://dx.doi.org/10.1016/B0-12-370878-8/00064-1.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., et al. (2019). Learning latent dynamics for planning from pixels. arXiv:1811.04551.

Hassabis, D., Kumaran, D., Summerfield, C., & Botvinick, M. (2017). Neuroscience-Inspired Artificial Intelligence. *Neuron*, *95*(2), 245–258. http://dx.doi.org/10.1016/j.neuron.2017.06.011.

Hawkins, J., Ahmad, S., & Cui, Y. (2017). A theory of how columns in the neocortex enable learning the structure of the world. *Frontiers in Neural Circuits*, *11*, 81. http://dx.doi.org/10.3389/fncir.2017.00081.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, *22*(241), 1–124.

Hussain, M., Bird, J. J., & Faria, D. R. (2019). A study on CNN transfer learning for image classification. In A. Lotfi, H. Bouchachia, A. Gegov, C. Langensiepen, & M. McGinnity (Eds.), *Advances in computational intelligence systems* (pp. 191–202). Cham: Springer International Publishing.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., & Levine, S. (2021). How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, *40*(4–5), 698–721. http://dx.doi.org/10.1177/0278364920987859.

Iyer, A., Grewal, K., Velu, A., Souza, L. O., Forest, J., & Ahmad, S. (2022). Avoiding catastrophe: Active dendrites enable multi-task learning in dynamic environments. *Frontiers in Neurorobotics*, *16*, http://dx.doi.org/10.3389/fnbot.2022.846219, URL https://www.frontiersin.org/articles/10.3389/fnbot.2022.846219.

Journé, A., Rodriguez, H. G., Guo, Q., & Moraitis, T. (2022). Hebbian deep learning without feedback. arXiv preprint arXiv:2209.11883.

Keraghel, I., Morbieu, S., & Nadif, M. (2024). Beyond words: A comparative analysis ofÂ LLM embeddings forÂ effective clustering. In I. Miliou, N. Piatkowski, & P. Papapetrou (Eds.), *Advances in intelligent data analysis XXII* (pp. 205–216). Cham: Springer Nature Switzerland.

Kingma, D. P., & Welling, M. (2022). Auto-encoding variational Bayes. arXiv:1312.6114.

Krithivasan, S., Sen, S., Venkataramani, S., & Raghunathan, A. (2022). Accelerating DNN training through selective localized learning. *Frontiers in Neuroscience*, *15*, http://dx.doi.org/10.3389/fnins.2021.759807, URL https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2021.759807.

Krotov, D., & Hopfield, J. J. (2019). Unsupervised learning by competing hidden units. *Proceedings of the National Academy of Sciences*, *116*(16), 7723–7731.

Leeb, F., Bauer, S., Besserve, M., & Schölkopf, B. (2022). Exploring the latent space of autoencoders with interventional assays. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems*: *vol. 35*, (pp. 21562–21574). Curran Associates, Inc., URL https://proceedings.neurips.cc/paper_files/paper/2022/file/87213955efbe48b46586e37bf2f1fe5b-Paper-Conference.pdf.

Liu, R., Nageotte, F., Zanne, P., de Mathelin, M., & Dresp-Langley, B. (2021). Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review. *Robotics*, *10*(1), http://dx.doi.org/10.3390/robotics10010022, URL https://www.mdpi.com/2218-6581/10/1/22.

Martins, A., & Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning* (pp. 1614–1623). PMLR.

Matsuo, Y., LeCun, Y., Sahani, M., Precup, D., Silver, D., Sugiyama, M., et al. (2022). Deep learning, reinforcement learning, and world models. *Neural Networks*, *152*, 267–275. http://dx.doi.org/10.1016/j.neunet.2022.03.037, URL https://www.sciencedirect.com/science/article/pii/S0893608022001150.

Menache, I., Mannor, S., & Shimkin, N. (2005). Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, *134*(1), 215–238. http://dx.doi.org/10.1007/s10479-005-5732-z.

Menghani, G. (2023). Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, *55*(12), http://dx.doi.org/10.1145/3578938.

Mnatzaganian, J., Fokoué, E., & Kudithipudi, D. (2017). A mathematical formalization of hierarchical temporal memory's spatial pooler. *Frontiers in Robotics and AI*, *3*, URL https://www.frontiersin.org/articles/10.3389/frobt.2016.00081.

Moraitis, T., Toichkin, D., Journé, A., Chua, Y., & Guo, Q. (2022). Softhebb: Bayesian inference in unsupervised hebbian soft winner-take-all networks. *Neuromorphic Computing and Engineering*, *2*(4), Article 044017.

Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., & Scaramuzza, D. (2017). The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research*, *36*(2), 142–149.

Musavi, M., Ahmed, W., Chan, K., Faris, K., & Hummels, D. (1992). On the training of radial basis function classifiers. *Neural Networks*, *5*(4), 595–603. http://dx.doi.org/10.1016/S0893-6080(05)80038-3, URL https://www.sciencedirect.com/science/article/pii/S0893608005800383.

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, *15*, 267–273.

O'Reilly, R. C., Russin, J. L., Zolfaghar, M., & Rohrlich, J. (2021). Deep Predictive Learning in Neocortex and Pulvinar. *Journal of Cognitive Neuroscience*, *33*(6), 1158–1196. http://dx.doi.org/10.1162/jocn_a_01708.

Ororbia, A., & Kifer, D. (2022). The neural coding framework for learning generative models. *Nature Communications*, *13*(1), 2064.

Oster, M., Douglas, R., & Liu, S.-C. (2009). Computation with spikes in a winner-take-all network. *Neural Computation*, *21*(9), 2437–2465. http://dx.doi.org/10.1162/neco.2009.07-08-829.

Padakandla, S. (2022). A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys*, *54*(6), 1–25. http://dx.doi.org/10.1145/3459991, URL https://dl.acm.org/doi/10.1145/3459991.

Razzhigaev, A., Mikhalchuk, M., Goncharova, E., Gerasimenko, N., Oseledets, I., Dimitrov, D., et al. (2024). Your transformer is secretly linear. arXiv preprint arXiv:2405.12250.

Rolls, E. T. (2021). *Brain computations: What and how*. USA: Oxford University Press.

Salvatori, T., Song, Y., Hong, Y., Sha, L., Frieder, S., Xu, Z., et al. (2021). Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, *34*, 3874–3886.

Schlag, I., Irie, K., & Schmidhuber, J. (2021). Linear transformers are secretly fast weight programmers. In *International conference on machine learning* (pp. 9355–9366). PMLR.

Siddiqui, S. A., Krueger, D., LeCun, Y., & Deny, S. (2023). Blockwise self-supervised learning at scale. arXiv preprint arXiv:2302.01647.

Su, D., Xu, Y., Winata, G. I., Xu, P., Kim, H., Liu, Z., et al. (2019). Generalizing question answering system with pre-trained language model fine-tuning. In A. Fisch, A. Talmor, R. Jia, M. Seo, E. Choi, D. Chen (Eds.), *Proceedings of the 2nd workshop on machine reading for question answering* (pp. 203–211). Hong Kong, China: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/D19-5827, URL https://aclanthology.org/D19-5827.

Willshaw, D. J., & Von Der Malsburg, C. (1976). How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, *194*(1117), 431–445.

Yue, Z., Wang, Y., He, Z., Zeng, H., McAuley, J., & Wang, D. (2024). Linear recurrent units for sequential recommendation. In *Proceedings of the 17th ACM international conference on web search and data mining* (pp. 930–938).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision – ECCV 2014* (pp. 818–833). Cham: Springer International Publishing.