The HTM Spatial Pooler – a neocortical algorithm for online sparse distributed coding

Yuwei Cui*, Subutai Ahmad, and Jeff Hawkins
Numenta, Inc,
Redwood City, California, USA

*Corresponding author
Emails: ycui@numenta.com, sahmad@numenta.com, jhawkins@numenta.com

A version of this manuscript has been submitted for publication as of Nov 1, 2016. This is a draft preprint. The final paper will be different from this version.

We welcome comments. Please contact the authors with any questions or comments.

The HTM Spatial Pooler – a neocortical algorithm for online sparse distributed coding

Yuwei Cui, Subutai Ahmad, and Jeff Hawkins
Numenta, Inc, Redwood City, California, United States of America

Abstract

Each region in the cortex receives input through millions of axons from sensory organs and from other cortical regions. It remains a mystery how cortical neurons learn to form specific connections from this large number of unlabeled inputs in order to support further computations. Hierarchical temporal memory (HTM) provides a theoretical framework for understanding the computational principles in the neocortex. In this paper we describe an important component of HTM, the HTM spatial pooler that models how neurons learn feedforward connections. The spatial pooler converts arbitrary binary input patterns into sparse distributed representations (SDRs) using competitive Hebbian learning rules and homeostasis excitability control mechanisms. Through a series of simulations, we demonstrate the key computational properties of HTM spatial pooler, including preserving semantic similarity among inputs, fast adaptation to changing statistics of the inputs, improved noise robustness over learning, efficient use of all cells and flexibility in the event of cell death or loss of input afferents. To quantify these properties, we developed a set of metrics that can be directly measured from the spatial pooler outputs. These metrics can be used as complementary performance indicators for any sparse coding algorithm. We discuss the relationship with neuroscience and previous studies of sparse coding and competitive learning. The HTM spatial pooler represents a neurally inspired algorithm for learning SDRs from noisy data streams online.

Introduction

Our brain continuously receives vast amounts of information about the external world through peripheral sensors that transform changes in light luminance, sound pressure, and skin deformations into millions of spike trains. Each cortical neuron has to make sense of a flood of time-varying inputs by forming synaptic connections to a subset of the presynaptic neurons. The collective activation pattern of population of neurons contributes to our perception and behavior. A central problem in neuroscience is to understand how individual cortical neurons learn to respond to specific input spike patterns, and how a population of neurons collectively represents features of the inputs in a flexible, dynamic, yet robust way.

In natural environments, a stimulus typically evokes a small set of cortical neurons at any time. A different subset of neurons is active for each stimulus. Existence of such sparse distributed representations (SDRs) has been documented in auditory, visual and somatosensory cortical areas (Vinje and Gallant, 2000; Weliky et al., 2003; Hromádka et al., 2008; Crochet et al., 2011). It has been proposed that sparse codes may be computationally efficient for sensory information processing (Olshausen and Field, 2004). From a theoretical perspective, sparse codes represent a favorable compromise between local codes and dense codes (Földiák, 2002). It allows simultaneous representation of distinct items with little interfere, while still has a large representational capacity (Kanerva, 1988; Ahmad and Hawkins, 2015).

The HTM spatial pooler (SP) is a biologically plausible neural network that continuously encodes streams of sensory inputs into sparse distributed representations (SDRs). HTM is a term used to describe our theory of neocortex (Hawkins et al., 2011). HTM spatial pooler is an important component of HTM and was originally described in the Numenta whitepaper (Hawkins et al., 2011). Recently there has been an increasing interest in the mathematical properties of the HTM spatial pooler (Mnatzaganian et al., 2016; Pietroń et al., 2016) and machine learning applications based on it (Thornton and Srbic, 2011; Ibrayev et al., 2016). However, the computational properties and design principles of HTM spatial pooler have not been well documented. We aim to fill this gap in this paper.

The HTM spatial pooler is inspired by working principles of the neocortex. It relies on competitive Hebbian learning (Hebb, 1949), homeostasis excitatory control mechanisms (Davis, 2006), topology of connections in sensory cortices (Udin and Fawcett, 1988; Kaas, 1997) and other physiological and anatomical properties of the neocortex. Unlike previous sparse coding studies, the HTM spatial pooler is not designed to minimize a single objective function of reconstruction error, but rather to achieve a set of computational properties that support further computation with SDRs. In addition to preserving the semantic similarity of the input space, the SP continuously adapts to statistics of the sensory input streams to ensure that the resulting representations are both efficient and noise robust. The sparsity of the SP output is tightly controlled around a target level. This stability of output sparsity reduces the recognition errors of downstream neurons.

The goal of this paper is twofold. First, we would like to fill a gap in the literature and provide a thorough discussion on computational properties of the HTM spatial pooler. Second, we provide a list of metrics that can be directly measured from any sparse coding algorithm. These metrics can be used as complementary performance indicators for sparse coding. The paper is organized as follows. We first introduce the HTM spatial pooler. We then list a set of desired computational properties for HTM spatial pooler and discuss how the spatial pooler achieved them through simulation. We demonstrate the value of learning in HTM spatial pooler on a real-world sequence prediction problem in an end-to-end HTM system. In the discussion, we propose potential neural mechanisms for HTM spatial pooler.

Materials and Methods

In this section we describe the implementation details of HTM spatial pooler (SP) and a set of statistical metrics that

quantify the performance of the spatial pooler. The implementation details of simulations are also provided.

HTM Spatial Pooler

The spatial pooler converts inputs with arbitrary sparsity to SDRs with fixed sparsity and dimensionality. The spatial pooler consists of a set of mini-columns. Cells in the same mini-column share the same feedforward receptive fields (Buxhoeveden, 2002). There is local inhibition among cells in neighboring mini-columns. This inhibition implements a *k*-winners-take-all computation. At any time, only a small fraction of the cells with the most active inputs become active. Feedforward connections onto active cells are modified according to Hebbian learning rules at each time steps. A homeostatic excitatory control mechanism, which we call "boosting", operates on a slower time scale. It increases the excitability of cells that are not active enough and decreases the excitability of cells that are active too often.

Each SP mini-column has a set of potential synapses to the input space. We denote the location of the *j*th input bits as $\mathbf{x}_\mathrm{j}$, and the center of potential synapses for the ith SP mini-column as $\mathbf{x}_i^c$. The potential synapses for the *i*th SP mini-column are located in a hypersphere of the input space centered at $\mathbf{x}_i^c$

$$PotentialInput(i) = \{ j \mid \|\mathbf{x}_\mathrm{j} - \mathbf{x}_i^c\| < \gamma \ and \ Z_{ij} < p\} \quad (1)$$

where γ is the radius of potential input, $Z_{ij} \sim U(0,1)$ is a random number uniformly distributed in [0, 1], p is the fraction of the inputs within a minicolumn's potential radius that a minicolumn is connected to.

The SP mini-columns are typically arranged topologically in any dimensional space. We denote the location of the *ith* SP mini-column as $\mathbf{y}_\mathrm{i}$, the neighboring mini-columns for this SP mini-column is given as

$$Neighbor(i) = \{ j \mid \|\mathbf{y}_\mathrm{i} - \mathbf{y}_\mathrm{j}\| < \phi, j \neq i\} \quad (2)$$

where $\|\cdot\|$ is the Euclidean distance between the mini-column *i* and *j*. The parameter ϕ controls the inhibition radius. If the input space does not have topology, we can use an infinitely large ϕ to implement global inhibition. Otherwise the inhibition radius is dynamically adjusted based on product of the average connected input spans of all SP mini-columns and the number of mini-columns per input. That is, ϕ increases if the average receptive field size increases. If SP inputs and mini-columns have the same dimensionality, $\phi = \gamma$ initially. This procedure ensures local inhibition affect mini-columns with inputs from the same region of the input space.

We model each synapse with a scalar permanence value and consider a synapse connected if its permanence value is above a connection threshold. We denote the set of connected synapses with a binary matrix $\mathbf{W}$,

$$W_\mathrm{ij} = \begin{cases} 1 \ if \ D_\mathrm{ij} \geq \theta_c \\ 0 \ otherwise \end{cases} \quad (3)$$

where D_ij gives the synaptic permanence from the *j*th input to the *i*th SP mini-column. The synaptic permanences are scalar values between 0 and 1, which are initialized to be randomly distributed between 0 and 1. The connection threshold θ_c is set to be 0.5 for all experiments, such that initially 50% of the potential synapses are connected.

Given an input $\mathbf{z}$, the activation of SP mini-columns is determined by first calculating the feedforward input to each mini-column, which we call the input overlap

$$o_i = b_i \sum_j W_\mathrm{ij} z_j \quad (4)$$

b_i is a positive boost factor that controls the excitability of the SP mini-column.

A SP mini-column becomes active if the feedforward input is above a stimulus threshold θ_{stim} and is among the top s% of its neighborhood,

$$a_i = \begin{cases} 1 \ if \ \mathrm{o}_i \geq \mathrm{prctile}(NeighborOverlap(i), 1-s) \ an \\ 0 \ otherwise \end{cases} \quad (5)$$

We typically set θ_{stim} to be a small positive number to prevent mini-columns without any input to become active. prctile(·,·) is the percentile function. $NeighborOverlap(i)$ is the overlap values for all neighboring mini-columns of the *i*th mini-column.

$$NeighborOverlap(i) = \{o_j | j \in Neighbor(i)\} \quad (6)$$

s is the target activation density. Typically we use *s*=0.02.

The feedforward connections are learned using a Hebbian rule. For each active SP mini-columns, we reinforce active input connections by increasing the synaptic permanence by p^+, and punish inactive connections by decreasing the synaptic permanence by p^-. The synaptic permanences are clipped at the boundaries of 0 and 1.

To update the excitability boost factors, we first calculate the time-averaged activation level for the each SP mini-column over the last *T* inputs.

$$\bar{a}_i(t) = \frac{(T-1) * \bar{a}_i(t) + a_i(t)}{T} \quad (7)$$

where $a_i(t)$ is the current activity of the *i*th mini-column at time *t*. *T* controls how fast the boost factors are updated. Because the activity is sparse it requires many steps before we can get a meaningful estimate of the activation level. Typically we chose *T* to be 1000.

The overall activation level in the neighborhood of a mini-column is also calculated as a reference activation level,

$$< \bar{a}_i(t) >= \frac{1}{|Neighbor(i)|} \sum_{j \in Neighbor(i)} \bar{a}_j(t) \quad (8)$$

The boost factor b_i is updated based on the difference of the time-averaged activation level of a mini-column and the reference activation level.

$$b_i = e^{-\beta(\bar{a}_i(t) - <\bar{a}_i(t)>)} \tag{9}$$

β is a positive parameter that controls the strength of the adaptation effect. The boost factor is one if the current activation level matches the reference activation level. It drops below one if a mini-column is active too often, and increases above one if a mini-column has less activity than its neighbors.

Statistical metrics for the spatial pooler

Spatial pooler is an unsupervised algorithm designed to achieve multiple functions. Although one could gauge the merit of spatial pooler in an end-to-end system for prediction or anomaly detection tasks (Ahmad and Purdy, 2016; Cui et al., 2016), it is desirable to quantify the performance of spatial pooler as a standalone component. Such metrics are particularly useful if configurations of the spatial pooler caused the end-to-end system to have poor performance. In this section, we describe several statistical metrics that can be directly calculated based on the inputs and outputs of the spatial pooler.

Sparseness. We define the population sparseness as

$$s^t = \frac{1}{N}\sum_{i=1}^{N} a_i^t \tag{10}$$

N is the number of SP mini-columns. The metric reflects the number of active neurons at each time step. Since we consider binary activations, the sparsity is straightforward to calculate. This metric has the same sprit as other population sparseness metrics for scalar value activations (Willmore and Tolhurst, 2001).

Entropy. Given of dataset M inputs, we first calculate the activation frequency of each SP mini-column as

$$\bar{a}_i = \frac{1}{M}\sum_{t=1}^{M} a_i^t \tag{11}$$

and then normalize the activation frequency to get a probability distribution

$$P(a_i) = \frac{\bar{a}_i}{\sum_{i=1}^{N} \bar{a}_i} \tag{12}$$

For practical numerical stability concerns, we enforce a minimum activation frequency of 10^{-7} before the normalization.

The entropy of the spatial pooler is defined as

$$Entropy = -\sum_{i=1}^{N} P(a_i)\log_2 P(a_i) + (1 - P(a_i))\log_2(1 - P(a_i)) \tag{13}$$

The entropy is maximized if every SP mini-column has the same activation frequency. The spatial pooler will have low entropy if a small amount of the SP mini-columns are active very frequently and the rest of the mini-columns are not active.

Noise robustness. It is desirable to have good noise robustness for frequently occurring inputs. There should be significant overlap between SP responses to noisy inputs and SP responses to clean inputs, such that one can still recognize the input even if it is contaminated by noise. This property is critical to ensure that a HTM based system is tolerant to noise in the input data stream.

We denote a clean input and a noise contaminated input as $\mathbf{z}_i$ and $\mathbf{z}_i'(k)$ respectively, where k denotes the amount of noise added to the input. The noisy inputs are generated by randomly changing k% of the active input bits to inactive, and simultaneously changing k% of the inactive input bits to active. This procedure randomizes inputs without changing the input sparsity. We denote the corresponding SP output as $\mathbf{a}_i$ and $\mathbf{a}_i'(k)$ respectively. We vary the amount of noise between 0 and 100%, and measure the fraction of shared active mini-columns in the SP output, averaged over a set of M inputs. The noise robust index is defined as

$$NoiseRobustness = \frac{1}{M}\sum_{i=1}^{M}\int_{k=0}^{100} \frac{\|\mathbf{a}_i \circ \mathbf{a}_i'(k)\|_0}{\|\mathbf{a}_i\|_0} \tag{14}$$

The L0-norm $\|\cdot\|_0$ gives the number of non-zero bits in a binary vector, $\circ$ represents element-wise multiplication. The noise robust index measures the area under the output overlap curve in Fig. 2C. The fraction of shared active mini-column starts at 100% when noise is zero, and decreases towards 0 as the amount of noise increases. The noise robustness thus lies between 0 and 1.

Stability over continuous learning The stability metric reflects the reliability of SP responses given repeated presentations of the same set of inputs. We measure stability of the spatial pooler by periodically presenting a set of test inputs. The test input is typically a subset of the training input data. Since the spatial pooler is continuously learning, the internal connections could change across consecutive presentations of the same set of test inputs. Instability without changes in the input statistics could negatively impact downstream processes. The stability index measures the fraction of shared active mini-columns for the same test input.

Denote the SP output to the ith test input at the jth test point as $\mathbf{a}_i^j$, the stability index at test point j is given as,

$$Stability(j) = \frac{1}{M}\sum_{i=1}^{M} \frac{\|\mathbf{a}_i^j \circ \mathbf{a}_i^{j-1}\|_0}{\|\mathbf{a}_i^{j-1}\|_0} \tag{15}$$

M is the number of test inputs. The stability lies between 0 and 1. It should be close to 1 for a stable spatial pooler.

Simulation details

We used either a 2-dimensional spatial pooler with 32x32 mini-columns for experiments with topology, or a dimensionless spatial pooler with 1024 mini-columns for experiments without topology. The complete set of spatial pooler parameters is given in Table 1.

Table 1 Parameters for the HTM spatial pooler

Common parameters	Value
Activation density	2%
Connection threshold for synaptic permanence θ_c	0.5
Synaptic permanence increment p^+	0.1
Synaptic permanence decrement p^-	0.02
Boosting strength β	100
Activation frequency duty cycle T	1000
Stimulus threshold θ_{stim}	1
Fraction of potential inputs p	1
Parameters for the 1-dimensional spatial pooler (no topology)	
Column dimensions	1024x1
Potential input radius γ	∞
Parameters for the 2-dimensional spatial pooler (with topology)	
Column dimensions	32x32
Potential input radius γ	5

Random Sparse Inputs. In this experiment we created a set of 100 random inputs with varying sparsity levels. Each input is a 32x32 image where a small fraction of the bits are active. The fraction of active inputs is uniformly chosen between 2% to 20%. We used a spatial pooler with topology in this experiment.

Random Bar Pairs and Random Crosses. Each random bar pair stimuli is a 10x10 image, with a horizontal bar and a vertical bar placed at random locations. The bars have a length of 5 pixels. Each random cross stimuli is a 10x10 image with a single cross. The cross consists of a horizontal bar and a vertical bar that intersect at the center. Each bar has a length of 5 pixels.

MNIST. We trained the spatial pooler on the MNIST database of handwritten digits (Lecun et al., 1998). We presented the full training set of 60,000 examples in a single pass to the spatial pooler. We visually examined the receptive field structures of a subset of randomly selected SP mini-columns after training.

Fault tolerance with topology. For the fault tolerance experiment, we used images of random bar sets as input. Each input has the dimensionality of 32x32 and contains 6 randomly located horizontal or vertical bars. Each bar has a length of 7. We used a two dimensional SP with 32x32 mini-columns that has topological connections to the input space. We first trained the intact SP on the random bars input until it stabilizes (after 18000 inputs). We then tested two different types of trauma: simulated stroke or simulated input lesion. During the simulated stroke experiment, we permanently eliminated 121 SP mini-columns that lie in an 11x11 region at the center of the receptive field. For the simulated input lesion experiment, we did not change the SP during the trauma. Instead, we permanently blocked the center portion of the input space (121 inputs that lie in an 11x11 region at the center of the input space). For both experiments, we monitored the recovery of SP for another 42000 steps.

Results

HTM spatial pooler

The HTM spatial pooler transforms a continuous stream of input patterns into sparse distributed representations. SP is an integral component of HTM. A single layer in HTM contains a set of mini-columns, each with a set of cells (Fig. 1A). In HTM theory, neurons use proximal and distal dendritic segments for different purposes (Fig. 1B) (Hawkins and Ahmad, 2016). In this paper we focus on the feedforward input connections that target the proximal dendritic zone. Since all cells in a given mini-column share the same feedforward receptive field, the spatial pooler represents a computation that occurs at the mini-column level. The inputs to the spatial pooler are binary vector representations. Each mini-column is connected to a subset of the input bits. There is local inhibition among neighboring mini-columns. Typically each inhibitory cell reciprocally connects to several hundred of nearby mini-columns. It ensures that only a small fraction of the connected mini-columns that receive the most inputs become active at any time, similar to a *k*-winners-take-all system (Majani et al., 1988; Makhzani and Frey, 2015).

The synaptic connections in the spatial pooler continuously learn from the sensory input stream. Learning occurs via growth of synapses. Each mini-column has the potential to connect to a set of pre-defined input bits, which we call potential connections. At any time, we apply Hebbian learning rules to active mini-columns (see Methods). To ensure that all mini-columns participate in representing the input, we include a homeostasis regulation mechanism of neural excitability (Davis, 2006), which we also call "boosting" in the software implementation. For each mini-column, we keep an online running average of its activation

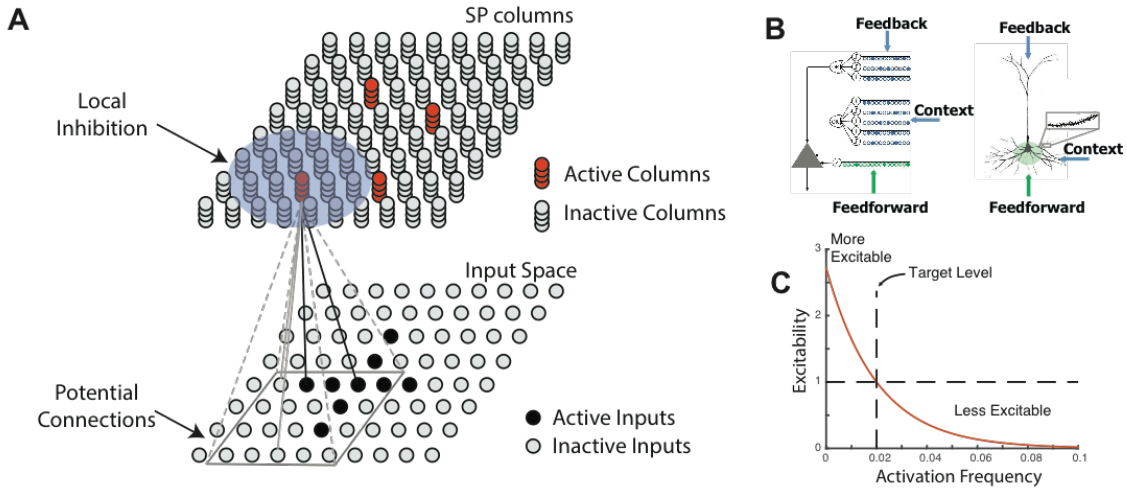


Figure 1 HTM Spatial Pooler. A. The HTM spatial pooler (SP) converts one or more inputs (*bottom*) to a sparse distributed representation (SDR) (*top*). Each SP mini-column forms synaptic connections to a subset of the input space (gray square, potential connections). A local inhibition mechanism ensures that a small fraction of the SP mini-columns that receive most of the inputs are active within the local inhibition radius (shaded blue area). Synaptic permanences are adjusted according to the Hebbian rule: for each active SP mini-column, active inputs (black lines) are reinforced and inactive inputs (gray lines) are punished. **B.** HTM neuron models used in the spatial pooler. An HTM neuron (left) has three distinct dendritic integration zones, corresponding to different parts of the dendritic tree of pyramidal neurons (right). We focus on the feedforward connections onto the proximal dendrite in this study. **C**. The excitability of a SP mini-column depends on its past activation frequency.

frequency. The excitability of a mini-column is low if a mini-column is more active than a target level and high if a mini-column is less active than the target level (Fig. 1C). The excitability acts as a positive weighting factor when calculating feedforward inputs (see Methods). The homeostasis mechanism encourages neurons with insufficient connections to become active and participate in representing the input.

Properties of HTM spatial pooler

The HTM spatial pooler achieves a set of computational properties to ensure flexible and robust representations of input streams with changing statistics. These properties are important for downstream neural computation. To illustrate these properties, we train the spatial pooler on a simple task of representing a set of 100 randomly generated input patterns with varying amount of sparsity. The input patterns are presented repeated to the HTM spatial pooler in a streaming fashion with random order.

A first property of the SP is to form representations of the input with a high and fixed level of sparsity. To contribute for further neural computation, the sparse activations of the SP have to be recognized by downstream neurons. A cortical neuron recognizes presynaptic input patterns by initiating nonlinear dendritic spikes (Major et al., 2013) or somatic action potentials (Bean, 2007), with thresholds depending on intrinsic cellular properties. Although a single dendritic segment can only sample a small fraction of the input space, the recognition is robust and reliable if the presynaptic inputs have a high fixed level of sparsity (Ahmad and Hawkins, 2016). With a highly variable sparsity, some input patterns would be much easier to detect than others, which could contribute to high false positive error. A fixed sparsity ensures all input patterns can be equally detected. In the spatial pooler, the fixed sparsity is achieved by the local inhibition circuit that enables k-winner-take-all activation. In our simulation, the population sparsity of the spatial pooler is always close to the target level of 2%, even though the input sparsity varies widely in the range of 2%~20% (Fig. 2A).

A second property of SP is that the system utilizes all available resources to learn optimal representations of the inputs. Given a limited number of neurons, it is better to ensure all neurons participate in representing the input space. We compute the distribution of activation frequency on the entire input dataset over the neural population. Before learning, about 30% the neurons are not responsive to any of the input vectors, whereas other neurons are responsive to many input vectors (Fig. 2B). After learning, almost all neurons are responsive to ~2% of the input patterns. The representation before learning is less efficient because the neither the non-responsive neurons nor the frequently responsive neurons convey meaningful information regarding the inputs. We measure the entropy of the activation frequency distribution and use it as a metric for efficiency. The entropy increases from 10.66 bits to 11.43 bits with training.

A third property of the spatial pooler is that the frequently occurring inputs are robustly represented even if the inputs are contaminated by noise. We test the noise robustness by adding varying amount of noise to the input vectors and measure the percentage of SP mini-columns that are active both for the original noiseless inputs and for the noisy inputs. Before learning, a small change in the input will cause a large change in the SP output, suggesting high noise sensitivity

(Fig. 2C, blue). With learning, the noise robustness gradually improves. After 40 repetitions of the dataset, there is no change on the SP output even if 40% of the active input bits are changed. We measure the noise robustness as the area under the noise robustness curves in Fig. 2C (see Methods).

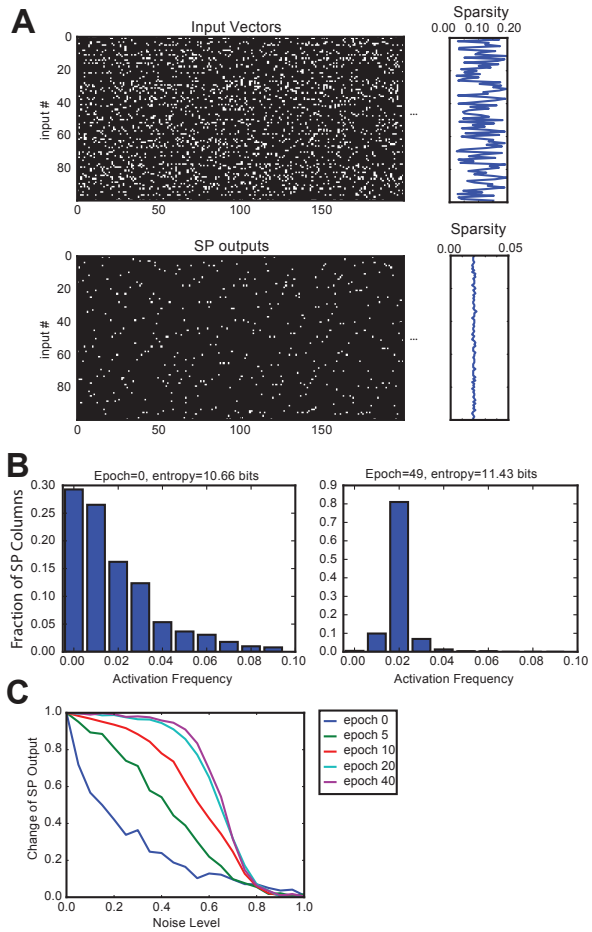


Figure 2 Spatial pooler forms SDRs with fixed sparsity and good noise robustness. **A.** We trained SP on a set of 100 randomly generated inputs (*top*). The input sparsity varies between 2% to 20%. The sparsity of the SP output lies close to 2% (*bottom*), despite the large variation of input sparsity. **B.** The distribution of activation frequency of SP mini-columns. Before learning (*left*), a significant fraction of the SP mini-columns (~30%) are not being used at all, while other mini-columns are active much more frequently. After learning (*right*), almost every mini-column is active for 2% of the time, suggesting every SP mini-column participate in representing the input. As a result, the entropy of the distribution is much higher. **C.** Noise robustness of SP. We tested SP on noisy inputs during learning. The change of the SP outputs is plotted as a function of the noise level. Before learning, a small amount of noise will lead to significant change in the SP output (*blue*), whereas after learning, there is almost no change in the SP output when 50% of the input bits changed.

A fourth property is the system should be flexible and capable of adapting to changing inputs. The cortex is highly flexible and plastic. Regions of the cortex can learn to represent different inputs in reaction to changes in the input data. The spatial pooler achieves this by learning continuously on each input. In a simulation experiment, we train the spatial pooler until it stabilizes on one set of inputs. We then present a completely different input dataset (Fig. 3A). Right after we switch to a new dataset, the noise robustness drops sharply (Fig. 3B, green). A large fraction of the SP mini-columns are not responsive to any input in the new dataset (Fig. 3C, left). This is because connections of the SP are optimized to represent the original, but not the new dataset. However, the spatial pooler quickly adapts the new input dataset and performance metrics recover back to the levels before the change. The SP adapts to the new dataset by first forming many more new synapses (Fig. 3A, 4th row) and then pruning unnecessary connections later (Fig. 3A, 5th row). The delay is caused by the difference between synapse permanence increases and decreases, as well as the initial state of SP before we changed the input dataset.

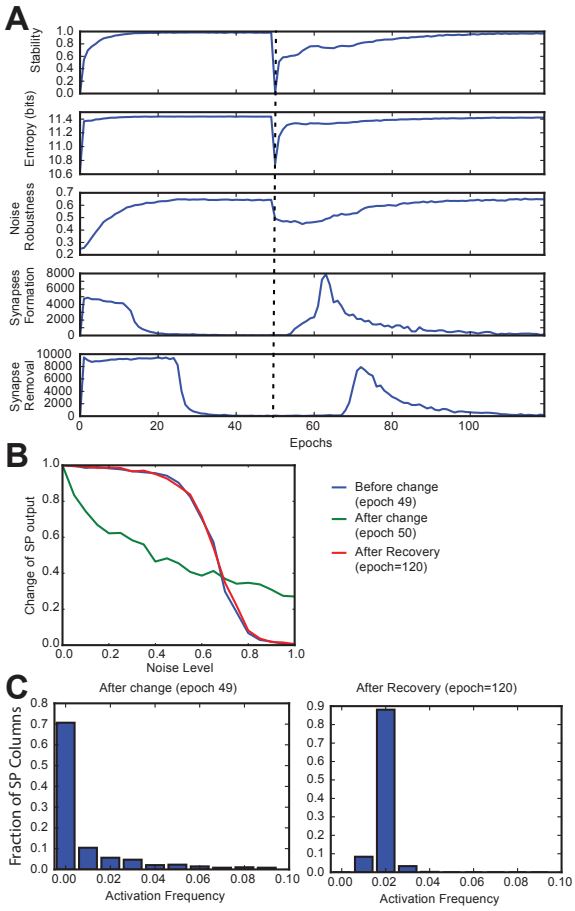


Figure 3 Continuous learning with HTM spatial pooler. SP continuously adapts to the statistics of the input data. SP is trained on a set of random inputs (described in Fig. 2) until it stabilizes. We then switch to a new set of inputs (black dashed line) and monitor the continuous adaptation of SP to the new dataset. **A.** Statistical metrics on SP during continuous learning: top: stability, 2nd row: entropy; 3rd row: noise robustness; 4th row: formation of new synapses; 5th row: removal of synapses. **B.** The noise robustness decreases right

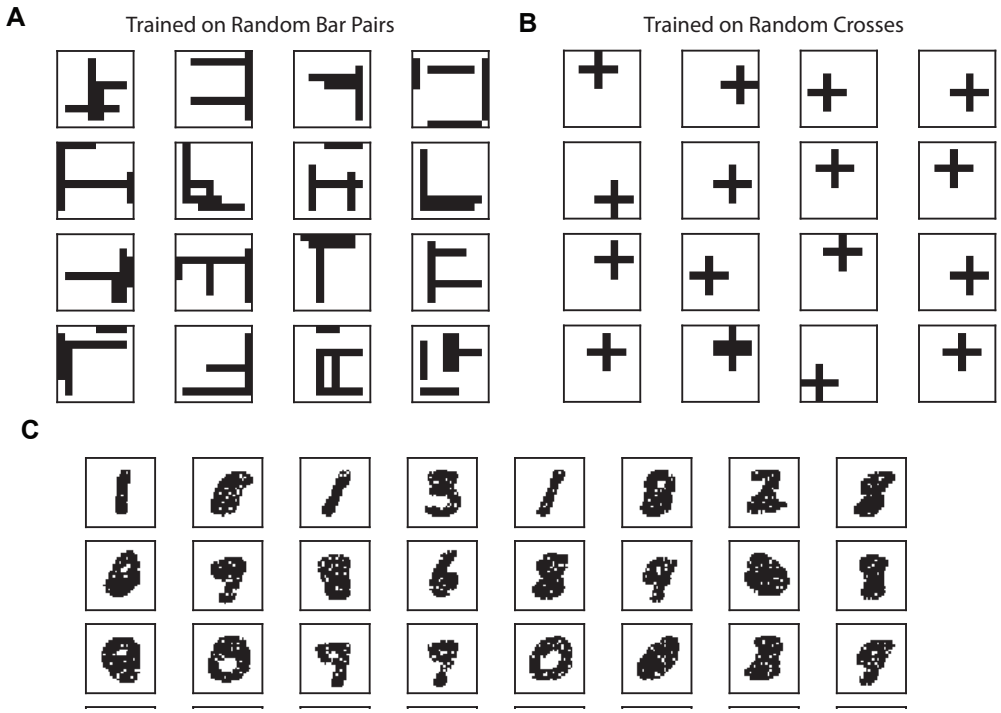


Figure 4 Receptive field of SP. The receptive fields of SP mini-columns capture statistics of the input data. **A.** Example SP Receptive fields trained on random bar pairs. **B.** Example SP receptive fields trained on random crosses. **C.** Example SP receptive fields trained on MNIST dataset.

after the change of the input dataset (blue vs. green), and recovers completely after the SP is trained on the new dataset for long enough time (red). **C.** Distribution of activation frequency of SP mini-columns right after the change in dataset (left) and after recovery (right).

We illustrate the computational properties using a set of complementary metrics including stability, entropy and noise robustness. It is important to note that no single metric is sufficient to ensure the system is behaving properly. For example, one can achieve good noise robustness by always using a small set of SP mini-columns, but that will give bad entropy. It is easy to achieve high entropy by using a random output at each time step, but that will cause bad stability. It is important to consider all the metrics together.

Receptive fields of the HTM spatial pooler

The spatial pooler achieves the aforementioned computational properties by adapting feedforward connections to statistics of the input data. To illustrate how receptive field structures of SP are shaped by the input data, we trained the spatial pooler on an artificial dataset. In this experiment, the training data stream consists of either pairs of randomly generated bars, or a single cross at a random location. Each input sample is independently generated in an online fashion (see Methods). We expect to see the receptive fields representing frequently occurring input patterns in the input dataset.

We plot example receptive fields for a set of randomly selected SP mini-columns in Fig. 4. When trained on the random bar data stream, the receptive field typically contains horizontal and vertical bar structures at random locations (Fig. 4A). Each SP mini-columns respond to more than one bar in order to achieve the target activation frequency of 2%. When trained on the random cross data stream, the resulting receptive field consists of cross structures that resemble statistics of the inputs (Fig. 4B).

We also trained the spatial pooler on the MNIST dataset. In this case the receptive field contains digit-like structures (Fig. 4C). Although some receptive field clearly detects single digits, others are likely responsive to multiple digits. This is because individual SP mini-columns do not behave like "grandmother cells" -- they are not meant to detect single instances of the inputs. Instead, a single input would be collectively encoded by a set of SP mini-columns. When trained on complex natural datasets, we expect to see a diversity of receptive structures within the spatial pooler, which may not resemble specific instances of the inputs.

Fault tolerance of the HTM spatial pooler

If part of the cortex is damaged, as often occurs in stroke or traumatic brain injury, there is often an initial deficit in perceptual abilities and motor functions which is followed by substantial recovery that occurs in the weeks to months following injury (Nudo, 2013). It has also been documented that sensory neurons receptive fields reorganize following

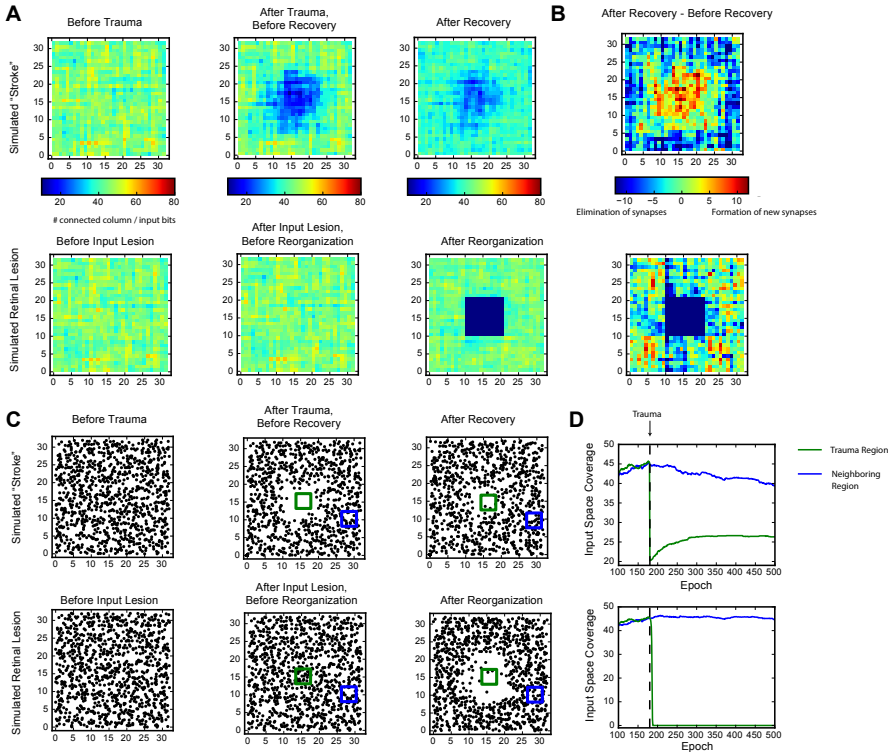


Figure 5 Recovery of HTM spatial pooler after simulated "stroke" or simulated retinal lesion. During the simulated stroke, a fraction of SP mini-columns that are connected to the center region of the input space is killed. During the simulated retinal lesion, the center portion of the input space is blocked while the spatial pooler and its feedforward inputs are kept intact. **A.** The number of SP mini-columns connected to each input bits before trauma (*left*), right after trauma (*middle*) and after recovery (*right*). The simulated stroke experiment is shown at the top and the simulated retinal lesion experiment is shown at the bottom. **B.** Growth and elimination of synapses during the recovery process for the simulated stroke (*top*) and retinal lesion (*bottom*) experiment. **C.** Receptive field centers of all SP mini-columns before trauma (*left*), right after trauma (*middle*) and after recovery (*right*). **D.** Number of mini-columns connected to the center region (green square in Fig. 5C) and a neighboring region (blue square) during the recovery process. The recovery is very fast for the retinal lesion experiment (bottom), and slower for the simulated stroke experiment (top).

restricted lesions of afferent inputs, such as retina lesion (Gilbert and Wiesel, 1992; Baker et al., 2005). We evaluated whether the HTM spatial pooler has the ability to recover from simulated trauma on either the afferent inputs or the SP mini-columns. If we block out part of the afferent inputs (simulated retinal lesion), we expect the SP mini-columns that normally respond to the damaged afferent inputs would instead respond to neighboring undamaged inputs. If we damage part of the SP mini-columns (simulated stroke), we expect neighboring mini-columns of the trauma area to take over the inputs that were represented by the damaged mini-columns.

In this experiment, we used a spatial pooler with 32x32 mini-columns that has topological connections to the input space. The inputs consist of images of random bars. We computed the input coverage, which gives the number of SP mini-columns that are connected to each input bit (Fig. 5A). We also estimated the receptive field centers for every SP mini-columns and plotted in the input space (Fig. 5C). After the network reached stable state, we eliminated afferent inputs that are near the center of the input space (simulated retinal lesion) or SP mini-columns that are located near the center of the network (simulated stroke). During this simulated trauma, 12% of inputs or 12% of the SP mini-columns (121/1024, within a 11x11 square) are permanently removed from the network.

In the simulated stroke experiment, the center portion of the input space becomes much less represented right after the trauma because the corresponding SP mini-columns are eliminated. The network partially recovers from the trauma after a few hundreds of epochs, with each epoch consisting of 100 inputs. During the recovery process, SP mini-columns near the trauma region shift their receptive field toward the trauma region and start to represent stimuli near the center (Fig. 5C, supplemental movie 1). The network forms many more new synapses in the center, which is accompanied by loss of synapses in the non-trauma region (Fig. 5B, top).

There is a clear recovery on the coverage of the trauma region (Fig. 5D, bottom).

In the simulated input lesion experiment, the center portion of the input space is blocked. Since there is no change on the SP mini-columns or the associated synaptic connections, there is no immediate change on the input space coverage (Fig. 5A, bottom) or the receptive field center distributions (Fig. 5C, bottom). However, the SP mini-columns quickly reorganize their receptive field within a few epochs. The SP mini-columns that respond to the center inputs starts to respond to inputs on the surrounding, non-damaged areas (Fig. 5C, supplemental movie 2). Almost all the connections to the lesion region are lost after the reorganization (Fig. 5B, bottom).

Our results agree with experimental studies on traumatic brain injury and afferent input lesions. If the lesion is confined to the afferent inputs (e.g., retinal or cochlear lesion), the receptive fields reorganize quickly, sometimes within a few minutes (Gilbert and Wiesel, 1992). In contrast, if part of the cortex is damaged, the recovery is partial and occurs on a much slower time scale (Nudo, 2013).

The role of HTM spatial pooler in streaming data prediction task

How does computational properties of the HTM spatial pooler contribute to better performances in an end-to-end cortical system? In this section, we evaluate the value of SP in sequence learning and prediction, a problem that is both central to neuroscience (Hawkins and Ahmad, 2016) and relevant to practical machine learning applications (Cui et al., 2016). This problem requires both learning reliable sparse representations for the inputs with spatial pooler as well as downstream computations that rely on recognition of these sparse representations with active dendrites. We used the HTM spatial pooler in an end-to-end HTM system (Fig. 6A). A stream of input data is first converted into binary representations using a set of encoders (Purdy, 2016). The spatial pooler takes the output of one or more encoders and learns sparse distributed representations of the inputs. The HTM sequence memory then learns sequences of SDRs and represents sequences with a sparse temporal code. Finally, we use a classifier to map outputs of HTM sequence memory into real-time predictions for future inputs. The spatial pooler and sequence memory and classifier all learn continuously on this task. It is important for the spatial pooler to output robust and efficient representations in order for the downstream components to learn.

We consider the problem of real-time prediction of the number of taxi passenger in New York City. We have studied the same problem in a previous paper using a fixed pre-trained spatial pooler in order to focus on the role of sequence memory (Cui et al., 2016). Here we consider the role of learning in the spatial pooler and evaluate three scenarios: using a randomly initialized SP without learning, allowing SP learning but without the homeostasis excitability control (No Boosting), and using a SP with both continuous learning and homeostasis control.

We fed the inputs in a single pass to mimic the scenario of real-time online prediction. The prediction error for the three cases is plotted as a function of training time (Fig. 6B). At the beginning of learning, the prediction error rapidly decreases, which representing the initial learning phase of the system. The increased error corresponds to real changes in the data that correspond to events of US holidays (e.g., Thanksgiving and Christmas) or New York City events (e.g., New York City Marathon).

The spatial pooler with both learning and boosting has the best performance throughout the prediction task. The spatial pooler with learning but not boosting initially has worse performance than a random SP, and later slightly outperformed the random SP. This suggest the importance of both the continuous Hebbian learning and the homeostasis excitability control mechanism in the spatial pooler. The difference of the performance can be understood by observing the distribution of activation frequency across SP mini-columns. Without the homeostasis excitability control, a large fraction of the SP mini-columns are not being used at all (Fig. 6C). It is more error-prone for the HTM sequence memory to learn transitions of such ill-behaved SDRs.

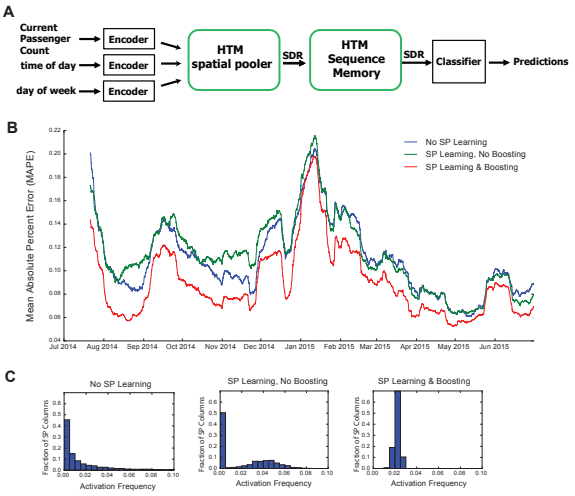


Figure 6 The role of HTM spatial pooler in prediction task. A. A complete HTM system of encoder -> spatial pooler -> sequence memory -> classifier is used for predicting the NYC taxi passenger count. B. Prediction error with an untrained random SP (blue), a SP with continuous learning but without boosting (green), and a SP with both continuous learning and boosting (red). C. Distribution of activation frequency of SP mini-columns. A large fraction of SP mini-columns are not being used for an untrained SP or for a SP without boosting. In contrast, almost all mini-columns are active for about 2% of the time when boosting is enabled.

Discussion and Conclusions

In this paper, we described the HTM spatial pooler, a neurally inspired algorithm for learning sparse distributed representations online. The spatial pooler is derived from computational principles of the neocortex. The goal of the HTM spatial pooler is to create SDRs and support essential neural computations such as sequence learning and memory.

The model satisfies a set of important properties, including tight control of output sparsity, efficient use of mini-columns, preserving similarity among inputs, noise robustness, fault tolerance and fast adaptation to changes. These properties are achieved using biologically plausible Hebbian learning rules and homeostasis excitability control mechanisms. The HTM spatial pooler leads to a flexible sparse coding scheme that can be used both in practical machine learning applications as well as to explain neuroscience experiments.

Potential Neural Mechanisms of spatial pooler

A layer in the HTM system contains a set of mini-columns. Each mini-column contains cells with the same feedforward receptive field. This is consistent with the columnar structures of the neocortex (Mountcastle, 1997; Buxhoeveden, 2002). A cortical mini-column is arguably the smallest repetitive functional unit of the mammalian cortex. A mini-column generally consists of 80-100 excitatory neurons that vertically traverse layer II-VI and typically has a diameter of 20-60 µm (Buxhoeveden, 2002). In the HTM theory, we propose cells within the same mini-column has the same feedforward connections but different lateral connections, thus representing the same feedforward input in different temporal contexts (Hawkins and Ahmad, 2016). It is unlikely for real neurons within a cortical mini-column to have the exact same connections. Instead, they may acquire the same receptive fields by sampling from a population of presynaptic afferent inputs. We model them as having the same feedforward connections for simplicity.

We propose the spatial pooler models feedforward receptive field learning at the mini-column level. This requires circuitry mechanisms to ensure that cells in the same mini-columns acquire the same receptive field. There is strong support for the existence of such mechanisms. Neurons within the same mini-column have almost identical receptive field locations, sizes and shapes, whereas RFs of neurons in neighboring mini-columns can differ significantly (Favorov and Kelly, 1996; Jones, 2000). This variability cannot be explained by the difference in feedforward inputs, because the extent of arborization of single thalamic afferent fiber is much more extensive than the dimensions of minicolumns, which can be as much as 900 µm in cats (Jones, 2000).

We propose two possible neural circuit mechanisms for the spatial pooler. In the first proposal (Fig. 7B, left), the feedforward thalamic inputs innervate both excitatory pyramidal neurons as well as an inhibitory neuron (*green*). This inhibitory neuron can indirectly activate the pyramidal neurons through a disynaptic dis-inhibition circuit. It acts as a "teacher" cell that guides the receptive field formation of excitatory neurons. There are many distinct classes of inhibitory neurons in the cortex. Some classes, such as bipolar cells and double bouquet cells exclusively innervate cells within a cortical mini-column (Markram et al., 2004; Wonders and Anderson, 2006). It is well documented that feedforward thalamacortical input strongly activates specific subtypes of inhibitory neurons (Gibson et al., 1999; Porter et al., 2001; Swadlow, 2002). It is likely these inhibitory neurons participate in defining and maintaining the feedforward receptive field of cortical mini-columns.

In the second proposal, a single excitatory cell receives thalamic inputs and innervates all excitatory cells in a mini-column. This excitatory neuron guides the receptive field formation of other excitatory cells in the mini-column. A similar circuit has been observed during early development. Subplate neurons, a transient population of neurons, receive synaptic inputs from thalamic axons, establishing a temporary link between thalamic axons and their final targets in layer IV (Friauf et al., 1990; Ghosh and Shatz, 1992; Kanold et al., 2003). It remains to be tested whether a similar circuit exists in adult brain. It is possible that both mechanisms exist, one that assists learning during development and one that continues through adulthood.

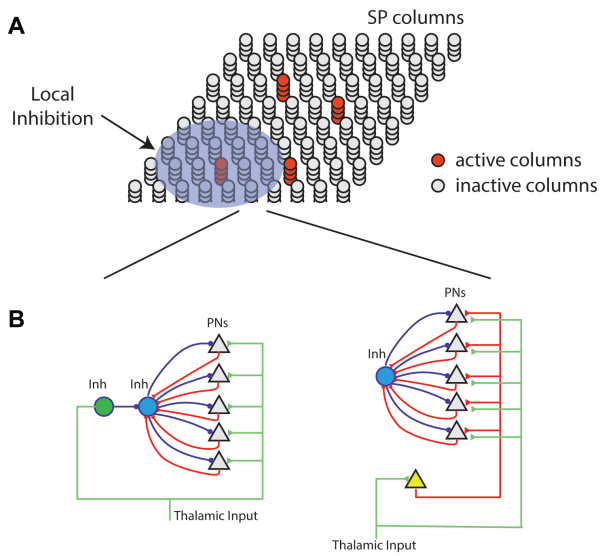


Figure 7 Neural mechanism of HTM Spatial pooler. **A.** Spatial pooler requires local across mini-column inhibition to ensure that a small fraction of the mini-columns are active at any time. **B.** Potential mechanisms to ensure neurons within the same mini-column share the same feedforward receptive field. *left*: the green inhibitory neuron controls the receptive fields of excitatory pyramidal neurons (gray triangles) through a dis-inhibition circuit. *right*: A single (or small number of) excitatory neurons (yellow) controls the receptive field of excitatory neurons. In both cases, PNs indirectly inhibit other PNs in the same mini-column through the within mini-column inhibition (blue).

The spatial pooler relies on several other neural mechanisms as well. First, the learning rule is based on competitive Hebbian learning. In the brain, such learning can be achieved via synaptic plasticity rules such as long-term potentiation (Teyler and DiScenna, 1987), long-term depression (Ito, 1989), or spike-time dependent plasticity (Song et al., 2000). Second, to encourage efficient use of SP mini-columns, we included homeostasis excitability control mechanism, which has been observed in cortical neurons (Davis, 2006). Finally, activity of the SP is computed using *k*-winners-take-all

computation, which can be achieved using local lateral inhibitions (Jonas and Buzsaki, 2007).

Relationship with other sparse coding techniques

The theory of sparse coding suggests that sparse activations in sensory cortices reduce energy consumption of the brain while preserving most of the information (Földiák, 2002; Olshausen and Field, 2004). Early studies of sparse coding explicitly optimize a cost function that combines low reconstruction error and high sparseness (Olshausen and Field, 1996, 1997). When applied on natural images, these techniques led to receptive fields that resemble those of V1 neurons (Olshausen and Field, 1996; Lee et al., 2006), suggesting that functions of early sensory neurons can be explained in the sparse coding framework. The sparse coding idea has also been implemented with biologically plausible local learning rules. Földiák showed that a neural network could learn a sparse code using Hebbian forward connections combined with a local threshold control mechanism (Földiák, 1990). It has been recently shown that such learning rules can be derived analytically by optimizing a similar cost function (Hu et al., 2014).

Most of the previous studies propose the goal of sparse coding is to avoid information loss, reduce energy consumption, and form associative memory with minimum cross talk (Olshausen and Field, 2004). A commonly used criterion is how well one can reconstruct the inputs given the sparse activations. Although these studies explain receptive structure in primary visual cortex and lead to practical machine learning algorithms for feature selection (Gui et al., 2016) and data compression (Pati et al., 2015), the purpose of neural computation is more than preserving information. In this paper, we take a different perspective and ask how computational properties of the HTM spatial pooler contribute to downstream cortical processing and behaviorally relevant computation. We listed a set of properties that can be directly measured from a sparse coding algorithm, including population entropy, noise robustness, stability and fault tolerance. These metrics provide additional metrics to evaluate the quality of sparse codes. We demonstrated that the HTM spatial pooler achieved these properties. It remains to be determined how other sparse coding techniques perform on these metrics, and whether these metrics can be incorporated as additional constraints in an optimization framework.

The Hebbian learning rules of HTM spatial pooler resemble many previous sparse coding algorithms (Földiák, 1990; Zylberberg et al., 2011; Hu et al., 2014). There are several differences. First, we included homeostasis excitability control as a gain modulation mechanism. The role of homeostasis is to make sure that the distribution of neural activity is homogeneous. It has been previously proposed that homeostasis is crucial in providing an efficient solution when learning sparse representations (Perrinet, 2010). Second, we used a local inhibition circuit that implements *k*-winners-take-all computation to have a tight control on the output sparseness, which is important when SP activations are used by downstream neurons with dendrites that has threshold nonlinearities. Finally, we used binary synapses where learning occurs via synaptogenesis (Zito and Svoboda, 2002). The use of binary synapses can dramatically speed up the computation. Although HTM spatial pooler may not perform as well as other sparse coding techniques (Lee et al., 2006) on tasks that require accurate input reconstruction, it represents a suitable candidate for learning sparse representations online from streaming data.

REFERENCES

Ahmad S, Hawkins J (2015) Properties of sparse distributed representations and their application to hierarchical temporal memory. arXiv:1503.07469.

Ahmad S, Hawkins J (2016) How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites. arXiv:1601.00720 [q-NC].

Ahmad S, Purdy S (2016) Real-Time Anomaly Detection for Streaming Analytics. arXiv:1607.02480.

Baker CI, Peli E, Knouf N, Kanwisher NG (2005) Reorganization of visual processing in macular degeneration. J Neurosci 25:614–618.

Bean BP (2007) The action potential in mammalian central neurons. Nat Rev Neurosci 8:451–465.

Buxhoeveden DP (2002) The minicolumn hypothesis in neuroscience. Brain 125:935–951.

Crochet S, Poulet JFA, Kremer Y, Petersen CCH (2011) Synaptic mechanisms underlying sparse coding of active touch. Neuron 69:1160–1175.

Cui Y, Ahmad S, Hawkins J (2016) Continuous online sequence learning with an unsupervised neural network model. Neural Comput 28:2474–2504.

Davis GW (2006) Homeostatic control of neural activity: from phenomenology to molecular design. Annu Rev Neurosci 29:307–323.

Favorov O V, Kelly DG (1996) Stimulus-response diversity in local neuronal populations of the cerebral cortex. Neuroreport 7:2293–2301.

Földiák P (1990) Forming sparse representations by local anti-Hebbian learning. Biol Cybern 64:165–170.

Földiák P (2002) Sparse coding in the primate cortex. In: The handbook of brain theory and neural networks, Second Edi. (Arbib M, ed), pp 1064–1068. MIT Press.

Friauf E, McConnell SK, Shatz CJ (1990) Functional synaptic circuits in the subplate during fetal and early postnatal development of cat visual cortex. J Neurosci 10:2601–2613.

Ghosh A, Shatz CJ (1992) Involvement of subplate neurons in the formation of ocular dominance columns. Science 255:1441–1443.

Gibson JR, Beierlein M, Connors BW (1999) Two networks of electrically coupled inhibitory neurons in neocortex. Nature 402:75–79.

Gilbert CD, Wiesel TN (1992) Receptive field dynamics in adult primary visual cortex. Nature 356:150–152.

Gui J, Sun Z, Ji S, Tao D, Tan T (2016) Feature Selection Based on Structured Sparsity: A Comprehensive Study. IEEE Trans neural networks Learn Syst.

Hawkins J, Ahmad S (2016) Why neurons have thousands of synapses, a theory of sequence memory in neocortex. Front Neural Circuits 10.

Hawkins J, Ahmad S, Dubinsky D (2011) Cortical learning algorithm and hierarchical temporal memory. Numenta Whitepaper:1–68 Available at: http://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf.

Hebb D (1949) The organization of behavior: a neuropsychological theory. Sci Educ 44:335.

Hromádka T, Deweese MR, Zador AM (2008) Sparse representation of sounds in the unanesthetized auditory cortex. PLoS Biol 6:e16.

Hu T, Pehlevan C, Chklovskii DB (2014) A Hebbian/Anti-Hebbian network for online sparse dictionary learning derived from symmetric matrix factorization. In: 2014 48th Asilomar Conference on Signals, Systems and Computers, pp 613–619. IEEE.

Ibrayev T, James A, Merkel C, Kudithipudi D (2016) A design of HTM spatial pooler for face recognition using Memristor-CMOS hybrid circuits. In: IEEE International Symposium on Circuits and Systems.

Igel C, Hüsken M (2003) Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing 50:105–123.

Ito M (1989) Long-term depression. Annu Rev Neurosci 12:85–102.

Jonas P, Buzsaki G (2007) Neural inhibition. Scholarpedia 2:3286.

Jones EG (2000) Microcolumns in the cerebral cortex. Proc Natl Acad Sci U S A 97:5019–5021.

Kaas JH (1997) Topographic maps are fundamental to sensory processing. Brain Res Bull 44:107–112.

Kanerva P (1988) Sparse Distributed Memory. The MIT Press.

Kanold PO, Kara P, Reid RC, Shatz CJ (2003) Role of subplate neurons in functional maturation of visual cortical columns. Science 301:521–525.

Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86:2278–2324.

Lee H, Battle A, Raina R, Ng AY (2006) Efficient sparse coding algorithms. In: Advances in Neural Information Processing Systems, pp 801–808.

Liang N-Y, Huang G-B, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17:1411–1423.

Majani E, Erlanson R, Abu-Mostafa Y (1988) On the K-winners-take-all network. Proc 1st Int Conf Neural Inf Process Syst:634–642.

Major G, Larkum ME, Schiller J (2013) Active properties of neocortical pyramidal neuron dendrites. Annu Rev Neurosci 36:1–24.

Makhzani A, Frey B (2015) k-Sparse Autoencoders. In: Advances in Neural Information Processing Systems 28, pp 2791--2799.

Markram H, Toledo-Rodriguez M, Wang Y, Gupta A, Silberberg G, Wu C (2004) Interneurons of the neocortical inhibitory system. Nat Rev Neurosci 5:793–807.

Mnatzaganian J, Fokoué E, Kudithipudi D (2016) A mathematical formalization of hierarchical temporal memory's spatial pooler. arXiv:1601.06116 [stat.ML].

Mountcastle VB (1997) The columnar organization of the neocortex. Brain 120 (Pt 4:701–722.

Nudo RJ (2013) Recovery after brain injury: mechanisms and principles. Front Hum Neurosci 7:887.

Olshausen BA, Field DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381:607–609.

Olshausen BA, Field DJ (1997) Sparse coding with an overcomplete basis set: A strategy employed by V1? Vision Res 37:3311–3325.

Olshausen BA, Field DJ (2004) Sparse coding of sensory inputs. Curr Opin Neurobiol 14:481–487.

Pati N, Pradhan A, Kanoje LK, Das TK (2015) An Approach to Image Compression by Using Sparse Approximation Technique. Procedia Comput Sci 48:769–775.

Perrinet LU (2010) Role of homeostasis in learning sparse representations. Neural Comput 22:1812–1836.

Pietroń M, Wielgosz M, Wiatr K (2016) Formal analysis of HTM spatial pooler performance under predefined operation conditions. In, pp 396–405. Springer International Publishing.

Porter JT, Johnson CK, Agmon A (2001) Diverse types of interneurons generate thalamus-evoked feedforward inhibition in the mouse barrel cortex. J Neurosci 21:2699–2710.

Purdy S (2016) Encoding Data for HTM Systems. arXiv:1602.05925.

Schaul T, Bayer J, Wierstra D, Sun Y, Felder M, Sehnke F, Rückstieß T, Schmidhuber J (2010) PyBrain. J Mach Learn Res 11:743–746.

Song S, Miller KD, Abbott LF (2000) Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. Nat Neurosci 3:919–926.

Swadlow HA (2002) Thalamocortical control of feed-forward inhibition in awake somatosensory "barrel" cortex. Philos Trans R Soc Lond B Biol Sci 357:1717–1727.

Teyler TJ, DiScenna P (1987) Long-term potentiation. Annu Rev Neurosci 10:131–161.

Thornton J, Srbic A (2011) Spatial pooling for greyscale images. Int J Mach Learn Cybern 2:1–10.

Udin SB, Fawcett JW (1988) Formation of topographic maps. Annu Rev Neurosci 11:289–327.

Vinje WE, Gallant JL (2000) Sparse coding and decorrelation in primary visual cortex during natural vision. Science 287:1273–1276.

Weliky M, Fiser J, Hunt RH, Wagner DN (2003) Coding of natural scenes in primary visual cortex. Neuron 37:703–718.

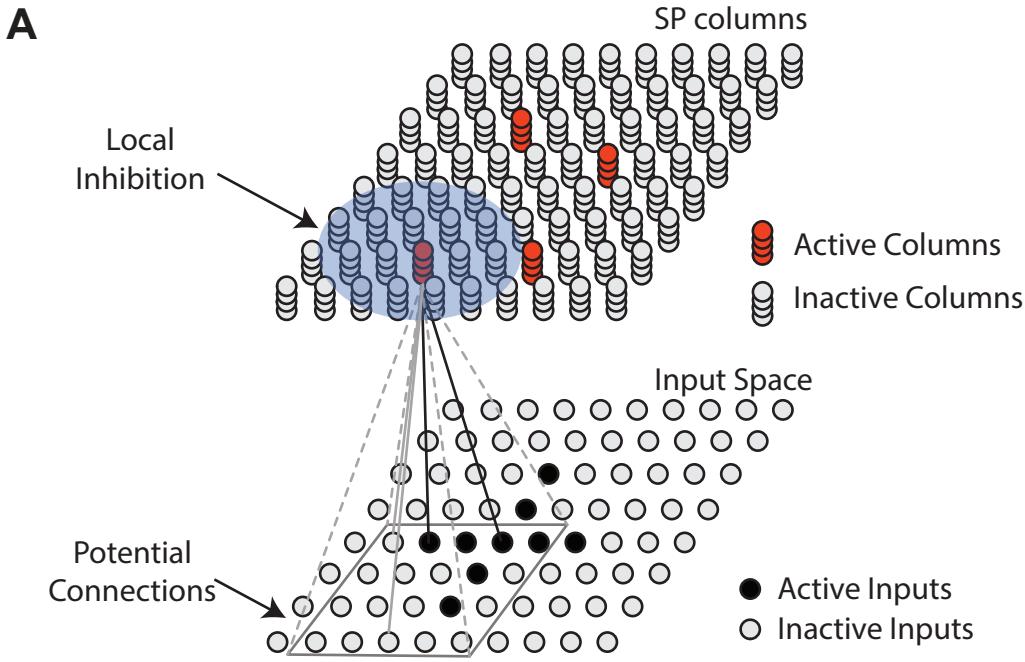
Williams RJ, Peng J (1990) An efficient gradient-based algorithm for on-line training of recurrent network trajectories. Neural Comput 2:490–501.

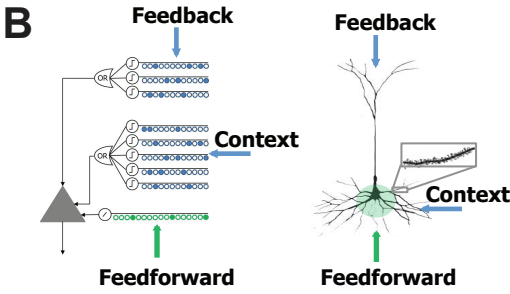
Willmore B, Tolhurst DJ (2001) Characterizing the sparseness of neural codes. Network 12:255–270.

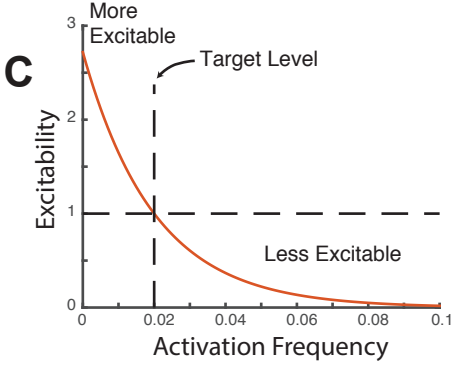
Wonders CP, Anderson SA (2006) The origin and specification of cortical interneurons. Nat Rev Neurosci 7:687–696.

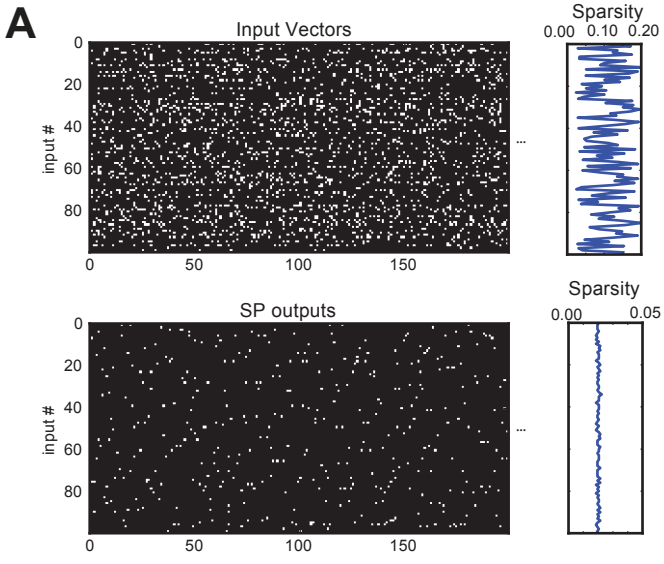
Zito K, Svoboda K (2002) Activity-dependent synaptogenesis in the adult Mammalian cortex. Neuron 35:1015–1017.

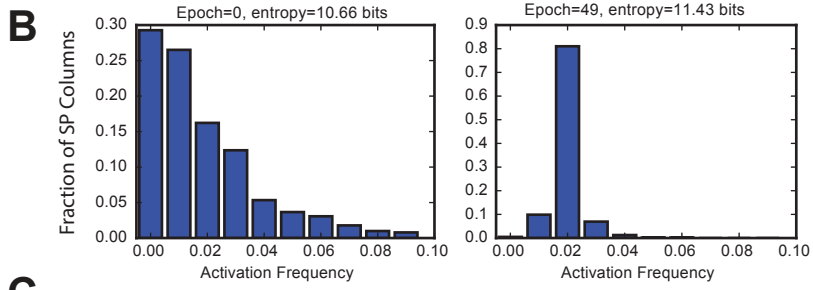
Zylberberg J, Murphy JT, DeWeese MR (2011) A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields. PLoS Comput Biol 7:e1002250.

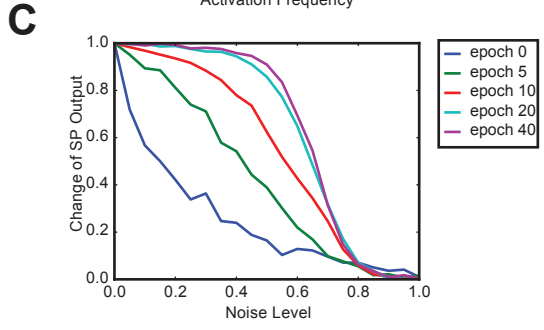

A
SP columns
Local
Inhibition
Active Columns
Inactive Columns
Input Space
Potential
Connections
Active Inputs
Inactive Inputs

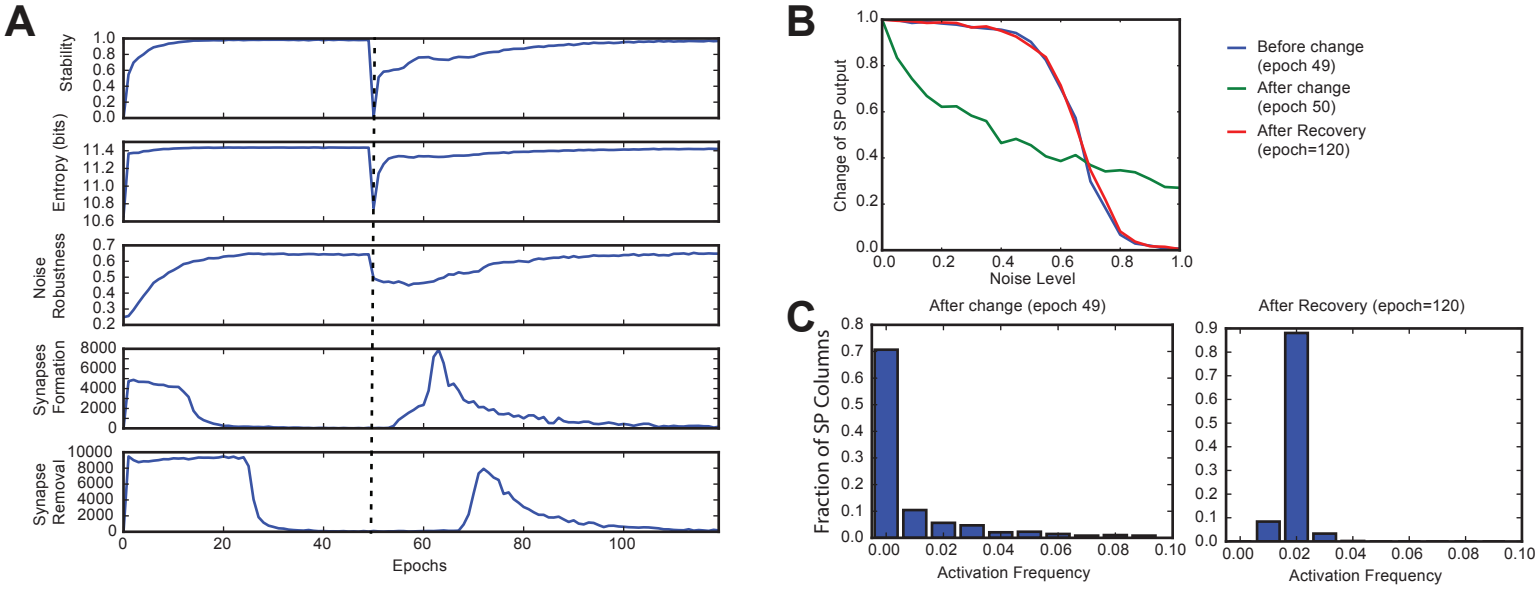

B
Feedback
Context
Feedforward
Feedback
Context
Feedforward

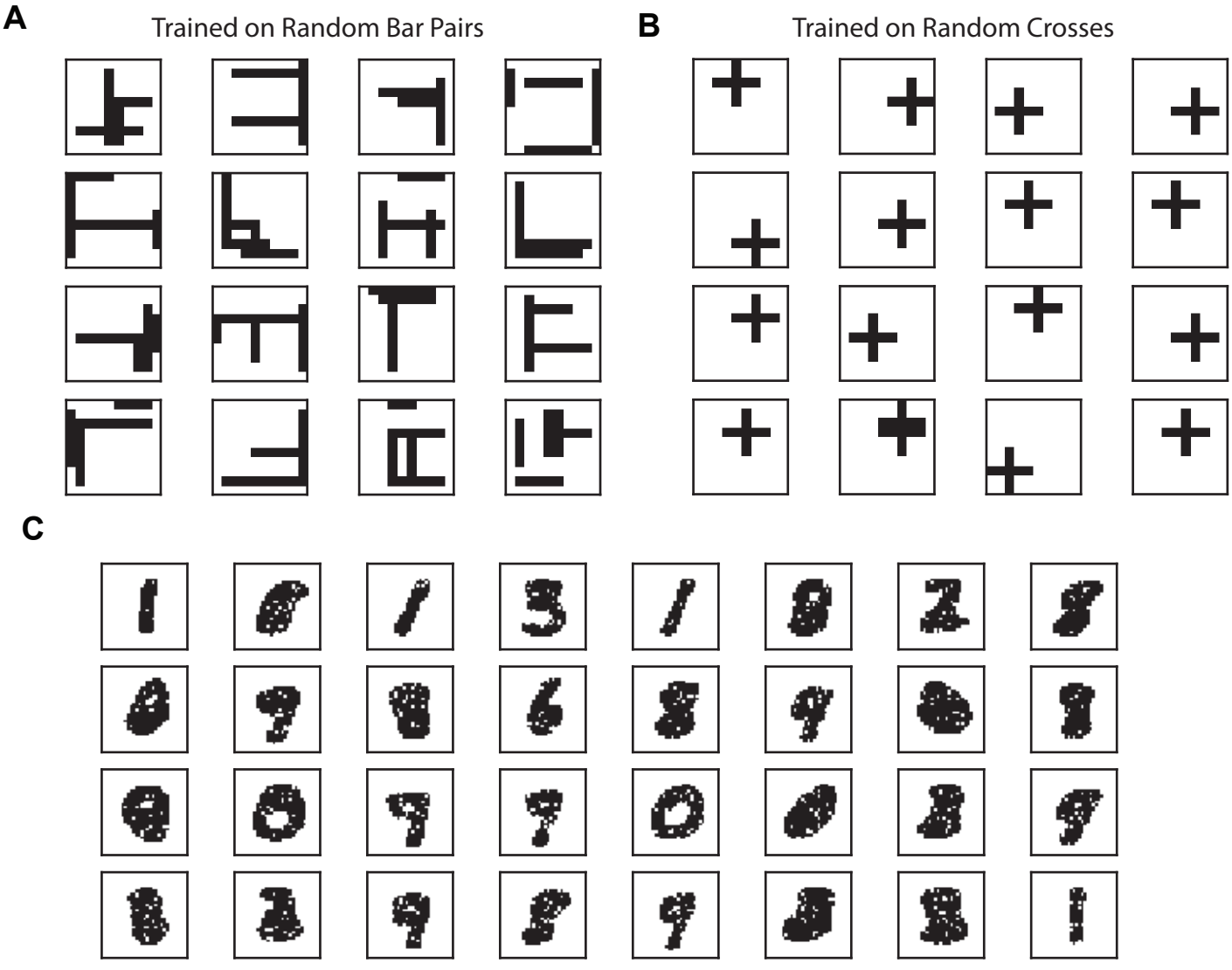

C
More
Excitable
Target Level
Less Excitable
Excitability
Activation Frequency
0
1
2
3
0
0.02
0.04
0.06
0.08
0.1

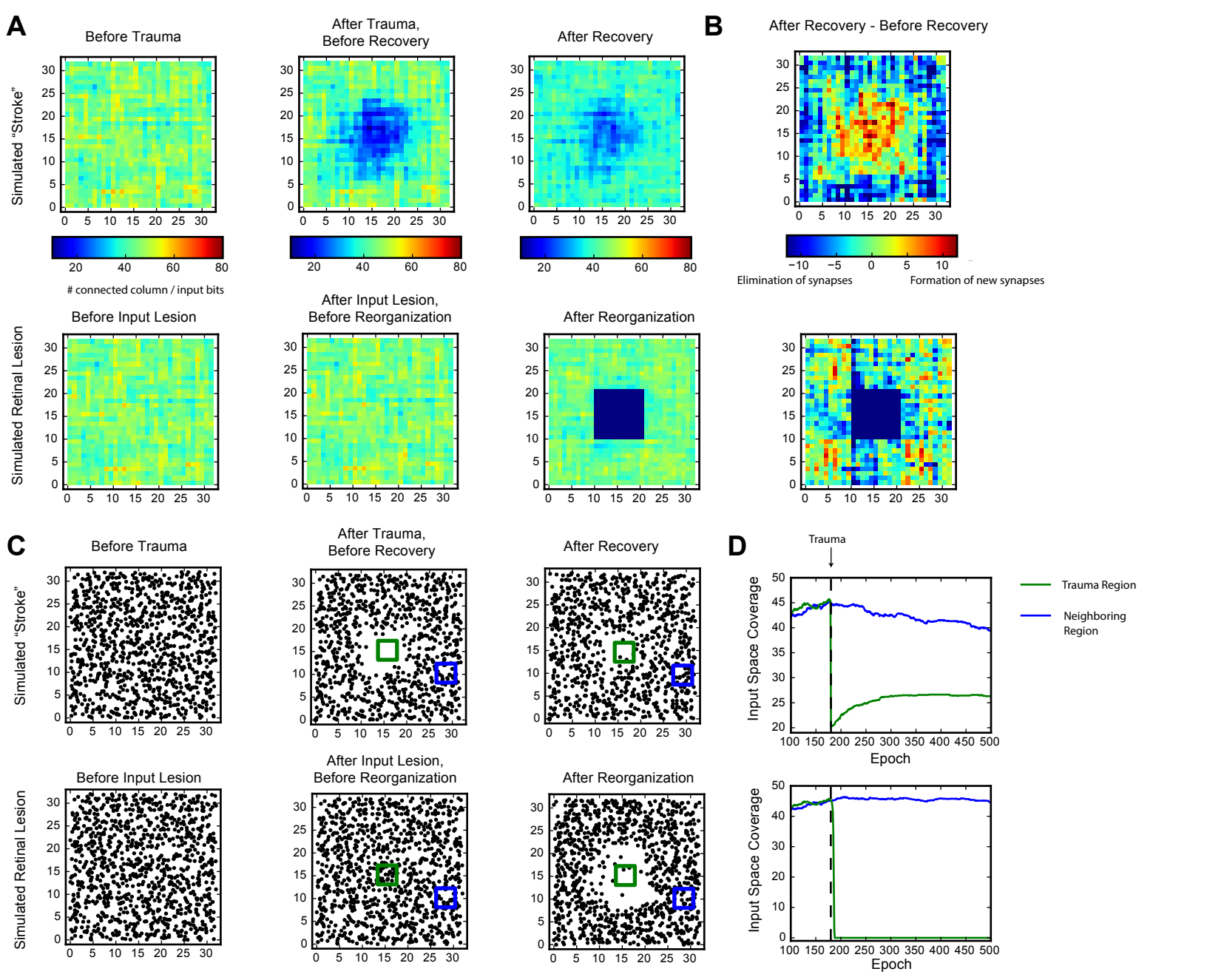

A
Input Vectors
input #
0
20
40
60
80
0
50
100
150
Sparsity
0.00
0.10
0.20
SP outputs
input #
0
20
40
60
80
0
50
100
150
Sparsity
0.00
0.05

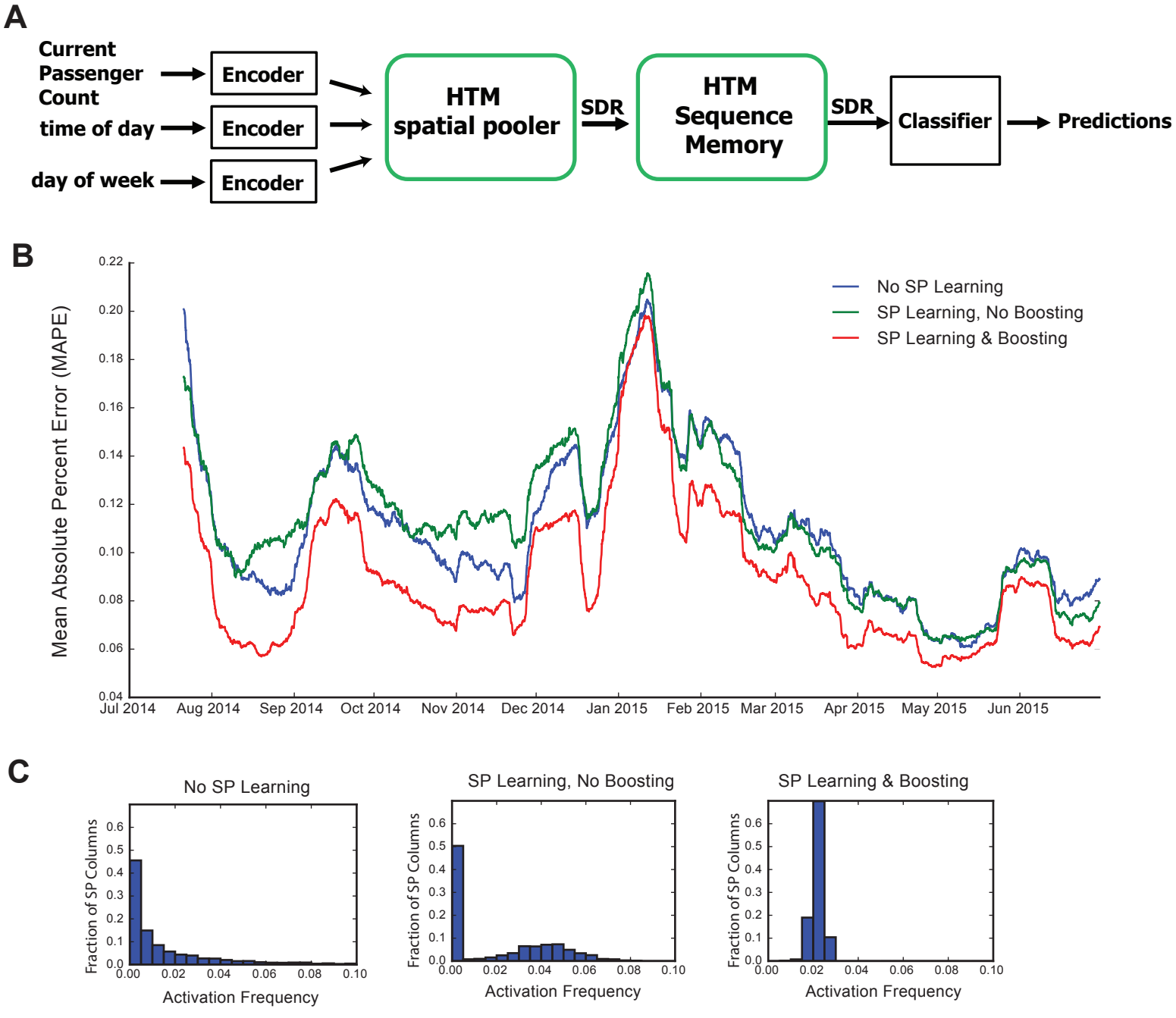

B
Epoch=0, entropy=10.66 bits
Fraction of SP Columns
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.00
0.02
0.04
0.06
0.08
0.10
Activation Frequency
Epoch=49, entropy=11.43 bits
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.00
0.02
0.04
0.06
0.08
0.10
Activation Frequency

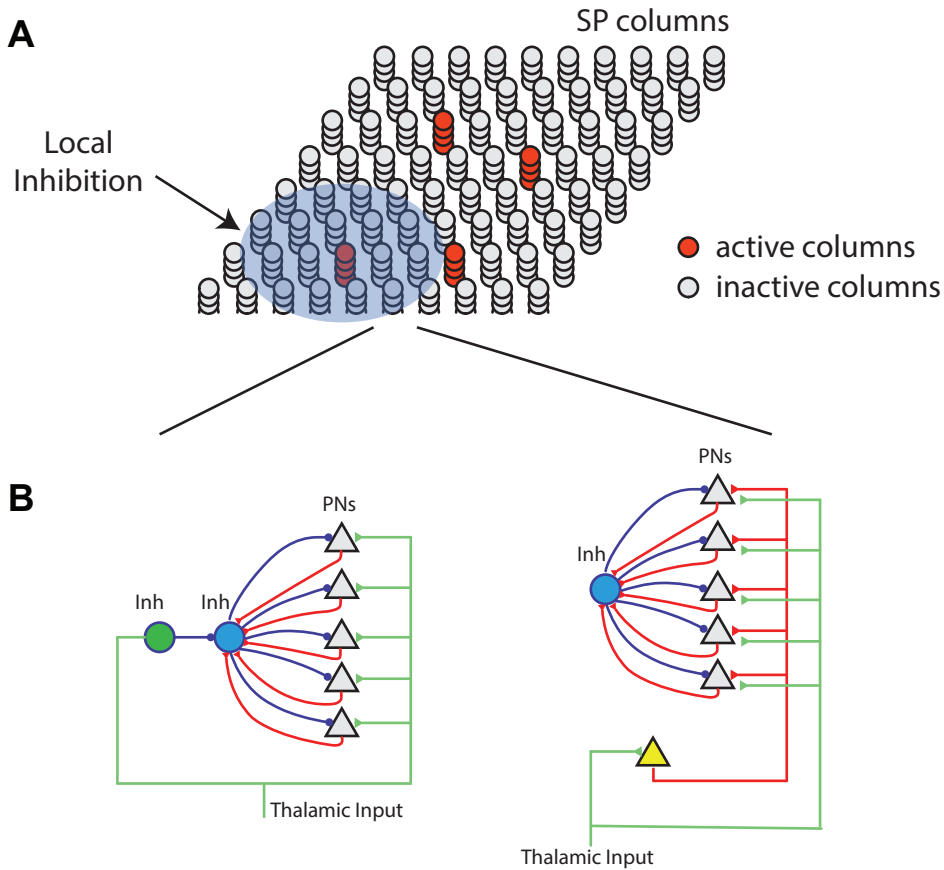

C
Change of SP Output
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Noise Level
epoch 0
epoch 5
epoch 10
epoch 20
epoch 40


A
Stability
Entropy (bits)
Noise Robustness
Synapses Formation
Synapse Removal
Epochs
B
Change of SP output
Noise Level
Before change (epoch 49)
After change (epoch 50)
After Recovery (epoch=120)
C
After change (epoch 49)
After Recovery (epoch=120)
Fraction of SP Columns
Activation Frequency


A
Trained on Random Bar Pairs
B
Trained on Random Crosses
C


A
Before Trauma
After Trauma, Before Recovery
After Recovery
Simulated "Stroke"
connected column / input bits
Before Input Lesion
After Input Lesion, Before Reorganization
After Reorganization
Simulated Retinal Lesion
B
After Recovery - Before Recovery
Elimination of synapses
Formation of new synapses
C
D
Trauma
Trauma Region
Neighboring Region
Input Space Coverage
Epoch


A
Current Passenger Count
time of day
day of week
Encoder
Encoder
Encoder
HTM spatial pooler
SDR
HTM Sequence Memory
SDR
Classifier
Predictions
B
Mean Absolute Percent Error (MAPE)
0.22
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
Jul 2014
Aug 2014
Sep 2014
Oct 2014
Nov 2014
Dec 2014
Jan 2015
Feb 2015
Mar 2015
Apr 2015
May 2015
Jun 2015
No SP Learning
SP Learning, No Boosting
SP Learning & Boosting
C
No SP Learning
SP Learning, No Boosting
SP Learning & Boosting
Fraction of SP Columns
Activation Frequency
0.00
0.02
0.04
0.06
0.08
0.10


A
SP columns
Local
Inhibition
active columns
inactive columns
B
PNs
Inh
Inh
Thalamic Input
PNs
Inh
Thalamic Input