Predictive Attractor Models

Ramy Mounir   **Sudeep Sarkar**
Department of Computer Science and Engineering, University of South Florida, Tampa
`{ramy, sarkar}@usf.edu`

Abstract

Sequential memory, the ability to form and accurately recall a sequence of events or stimuli in the correct order, is a fundamental prerequisite for biological and artificial intelligence as it underpins numerous cognitive functions (e.g., language comprehension, planning, episodic memory formation, etc.) However, existing methods of sequential memory suffer from catastrophic forgetting, limited capacity, slow iterative learning procedures, low-order Markov memory, and, most importantly, the inability to represent and generate multiple valid future possibilities stemming from the same context. Inspired by biologically plausible neuroscience theories of cognition, we propose *Predictive Attractor Models (PAM)*, a novel sequence memory architecture with desirable generative properties. PAM is a streaming model that learns a sequence in an online, continuous manner by observing each input *only once*. Additionally, we find that PAM avoids catastrophic forgetting by uniquely representing past context through lateral inhibition in cortical minicolumns, which prevents new memories from overwriting previously learned knowledge. PAM generates future predictions by sampling from a union set of predicted possibilities; this generative ability is realized through an attractor model trained alongside the predictor. We show that PAM is trained with local computations through Hebbian plasticity rules in a biologically plausible framework. Other desirable traits (e.g., noise tolerance, CPU-based learning, capacity scaling) are discussed throughout the paper. Our findings suggest that PAM represents a significant step forward in the pursuit of biologically plausible and computationally efficient sequential memory models, with broad implications for cognitive science and artificial intelligence research. Illustration videos and code are available on our project page: https://ramymounir.com/publications/pam.

1   Introduction

Modeling the temporal associations between consecutive inputs in a sequence (i.e., sequential memory) enables biological agents to perform various cognitive functions, such as episodic memory formation [53, 64, 15], complex action planning [22] and translating between languages [3]. For example, playing a musical instrument requires remembering the sequence of notes in a piece of music; similarly, playing a game of chess requires simulating, planning, and executing a sequence of moves in a specific order. While the ability to form such memories of static, unrelated events has been extensively studied [56, 55, 63, 69, 66], the ability of biologically-plausible artificial networks to learn and recall temporally-dependent patterns has not been sufficiently explored in literature [62]. The task of sequential memory is considered challenging for models operating under biological constraints (i.e., local synaptic computations) for many reasons, including catastrophic forgetting, ambiguous context representations, multiple future possibilities, etc.

In addition to the biological constraint, we impose the following set of desirable characteristics as learning constraints on sequence memory models.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

- The learning of one sequence does not overwrite the previously learned sequences. This property is defined under the continual learning framework as avoiding catastrophic forgetting and evaluated with the Backward Transfer (BWT) metric [38].

- The model should uniquely encode inputs based on their context in a sequence. Consider the sequence of letters "EVER"; the representation of "E" at position 1 should be different from "E" at position 3, thus resulting in different predictions: "V" and "R". Moreover, the representation of "E" at position 3 in "EVER" should be different from "E" at position 3 in "CLEVER". Therefore, positional encoding is not a valid solution.

- When presented with multiple valid, future possibilities, the model should learn to represent each possibility separately yet stochastically sample a single valid possibility. Consider the two sequences "THAT" and "THEY"; after seeing "TH", the model should learn to generate either "A" or "E", but not an average [37] or a union of both [25].

- The model should be capable of incrementally learning each transition without seeing the whole sequence or revisiting older sequence transitions that are previously learned. This property falls under online learning constraints, also called stream learning [45, 25].

- The learning algorithm should be tolerant and robust to significant input noise. A model should continuously clean the noisy inputs using learned priors and beliefs, thus performing future predictions based on the noise-free observations [25].

We propose *Predictive Attractor Models (PAM)*, which consists of a state prediction model and a generative attractor model. The predictor in PAM is inspired by the Hierarchical Temporal Memory (HTM) [25] learning algorithm, where a group of neurons in the same cortical minicolumn share the same receptive feedforward connection from the sensory input on their proximal dendrites. The depolarization of the voltage of any neuron in a single minicolumn (i.e., on distal dendrites) primes this depolarized neuron to fire first while inhibiting all the other neurons in the same minicolumn from firing (i.e., competitive learning). The choice of which neurons fire within the same minicolumn is based on the previously active neurons and their trainable synaptic weights to the depolarized neurons, which gives rise to a unique context representation for every input [20]. The sparsity of representations (discussed later in Section 3.2) allows for multiple possible predictions to be represented as a union of individual cell assemblies. The Attractor Model learns to disentangle possibilities by strengthening the synaptic weights between active neurons of input representations and inhibiting the other predicted possibilities from firing, effectively forming fixed point attractors during online learning. During recall, the model uses these learned conditional attractors to sample one of the valid predicted possibilities or uses the attractors as prior beliefs for removing noise from sensory observations.

PAM satisfies the above-listed constraints for a sequential memory model, whereas the current state-of-the-art models fail in all or many of the constraints, as shown in the experiments. Our contributions can be summarized as follows: (1) Present the novel *PAM* learning algorithm that can explicitly represent context in memory without backpropagation, avoid catastrophic forgetting, and perform stochastic generation of multiple future possibilities. (2) Perform extensive evaluation of PAM on multiple tasks (e.g., sequence capacity, sequence generation, catastrophic forgetting, noise robustness, etc.) and different data types (e.g., protein sequences, text, vision). (3) Formulate PAM and its learning rules as a State Space Model grounded in variational inference and the Free Energy Principle [17].

2 Background and Related Works

2.1 Predictive Coding

Predictive coding proposes a framework for the hierarchical processing of information. It was initially formulated as a time series compression algorithm to create a more efficient coding system [16, 46]. A few decades later, PC was used to model visual processing in the Retina [59, 27] as an inference model. In the seminal work of Rao and Ballard [52], PC was reformulated as a general computational model of the cortex. The main intuition is that the brain continuously predicts all perceptual inputs, resulting in a quantity of prediction error, which can be minimized by adjusting its neural activities and synaptic strengths. In-depth variational free energy derivation is provided in Appendix A.1.

2

PC defines two subgroups of neurons: value z and error. Each neuron contains a value node sending its prediction to the lower level $\hat{z}_l = f_{l+1}(z_{l+1})$ through learnable function f, and error node propagating its computed error to the higher level. The total prediction error is computed as $\epsilon = \sum_l ||(z_l - \hat{z}_l)||_2^2$, which is minimized by first running the network value nodes to equilibrium through optimizing the value nodes $\{z_l\}_{l=0}^L$. At equilibrium, the value nodes are fixed, and inferential optimization is performed by optimizing the functions $\{f_l\}_{l=1}^L$. Both optimizations aim to minimize the same prediction error over all neurons. This propagation of error to equilibrium is shown to be equivalent to backpropagation but using only local computations [66]. The PC formulation has shown success in training on static and i.i.d data [66, 69, 57, 23]. More recently [62], Temporal Predictive Coding (tPC) has also shown some success in sequential memory tasks by modifying error formulation to account for a one-step synaptic delay through interneurons, thus modeling temporal associations between sequence inputs. In the experiments, we compare our model to tPC and its 2-layer variant [62]. A few PC-based models [47, 48] have shown promise for class-incremental continual learning tasks, other PC-inspired models, such as [39, 23, 24], have diverged from the biologically plausible constraints by training through backpropagation through time.

2.2 Fixed-Point Attractor Dynamics

Attractor dynamics refer to mathematical models that describe the behavior of dynamical systems. In our review, we focus on fixed point attractors, specifically Hopfield Networks [26] (HN), which is an instance of associative memory models [32, 29, 30]. Consider, an ordered sequence of $T + 1$ consecutive patterns $\boldsymbol{x} = [\boldsymbol{x_t}]_{t=1}^{T+1}$, where $\boldsymbol{x_t} \in \{-1, 1\}^N$. We refer to the Universal Hopfield Networks (UHN) framework [41] to describe all variants of HN architecture using a similarity (sim) function and a separation (sep) function, as shown in equation 1. This family of models can be viewed as a memory recall function, where a query ξ (i.e., noisy or incomplete pattern) is compared to the existing patterns to compute similarity scores using the "sim" function. These scores are then used to weight the projection patterns after applying a "sep" function to increase the separation between similarity scores. The classical HN uses symmetric weights to store the patterns; therefore, it cannot be used to model temporal associations in a sequence. The asymmetric Hopfield Network (AHN) [58] uses asymmetric weights to recall the next pattern in the sequence for a given query ξ.

$$\hat{\xi} = \underbrace{P}_{\text{Projection}} \cdot \underbrace{\text{sep}}_{\text{Separation}} \underbrace{(\text{sim}(M, \xi))}_{\text{Similarity}} = \begin{cases} \sum_{t=1}^{T} \mathbf{x}_{t+1} \text{sep}\left(\text{sim}(\mathbf{x}_t, \xi)\right) & \text{Asymmetric Weights} \\ \sum_{t=1}^{T+1} \mathbf{x}_t \text{sep}\left(\text{sim}(\mathbf{x}_t, \xi)\right) & \text{Symmetric Weights} \end{cases} \quad (1)$$

When a dot product "sim" function and identity "sep" function are used, we get the classical HN [26] and AHN [58]. A few variants have been proposed to increase the capacity of the model. Recently, [12] has extended AHN by using a polynomial (with degree d) or a softmax function (with temperature β) as the "sep" function. HN can also be applied to continuous dense patterns [34, 51, 12], and extended to sparse modern Hopfield models [28, 67].

2.3 Predictive Learning

Predictive learning takes a more general form of minimizing the prediction error between two views of the same input to improve representations. Many backpropagation-based approaches to predictive learning have been proposed; most recently, JEPA [37] and its variants [8, 10, 9, 70], learn useful dense representations from images and videos using the predictive objective. Other models, such as [11, 21, 13] - to list a few, use a similar methodology of predicting distorted versions of the same input to learn good feature representations. Prediction-based approaches have also been used to segment videos into events temporally [2, 42] and spatially [44, 43, 1]. More recently, STREAMER [45] used a predictive learning approach to achieve hierarchical segmentation and representation learning from streaming egocentric videos, where a high prediction error is used as an event boundary. While these biologically implausible approaches show impressive results on their respective tasks, they still suffer from deep learning known limitations, such as catastrophic forgetting and the inability to generate multiple possibilities in regression-based predictive tasks. *Hierarchical Temporal Memory* (HTM) [25] is a predictive approach that is heavily inspired by neurobiology. HTM relies on lateral inhibition between neurons of the same minicolumn and sparsity of input representations (i.e., SDR) to learn temporal context and associations using only local Hebbian rules.

3

HTM can be applied to online tasks, such as anomaly detection [6] and object recognition [36], but it is currently incapable of generating future predictions in auto-regressive prediction tasks.

3 Predictive Attractor Models

3.1 State Space Model (SSM) formulation
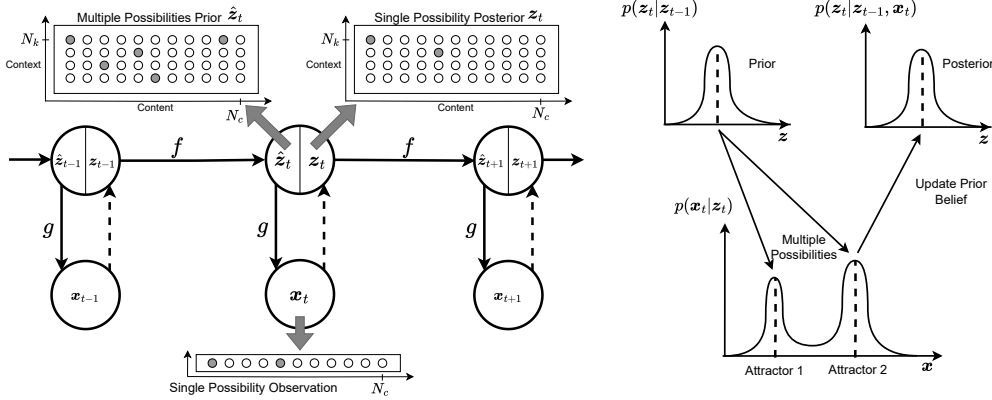
Figure 1: State Space Model. (Left): Dynamical system represented by first-order Markov chain of latent states z with transition function f and an emission function g which projects to the observation states x. (Right): Gaussian form assumptions for the prior $\hat{z}$ and posterior z latent states, and the Mixture of Gaussian model representing the conditional probability of multiple possibilities $p(x|z)$

PAM can be represented as a dynamical system with its structure depicted by a Bayesian probabilistic graphical [35, 31] model, more specifically, a State Space Model, where we can perform Bayesian inference on the latent variables and derive learning rules using Variational Inference. Formally, we define a state space model as a tuple $(\mathcal{Z}, \mathcal{X}, f, g)$, where $\mathcal{Z}$ is the latent state space, $\mathcal{X}$ is the observation space, and f and g are the transition and emission functions respectively (similar to HMM [50]). We consider a Gaussian form with white Gaussian noise covariance Σ_z for the latent states. However, we assume a latent state z can generate multiple valid possibilities. Therefore, we model the conditional probability $p(x_t|z_t)$ as a Multivariate Gaussian Mixture Model (GMM), where each mode is considered a possibility or a fixed-point attractor in an associative memory model. The GMM has C components with means $g_c(z_t)$, covariances Σ_c and component weights w_c. The SSM dynamics can be formally represented with the following equations:

$$z_t|z_{t-1} \sim \mathcal{N}(f(z_{t-1}), \Sigma_z), \quad \text{and} \quad x_t|z_t \sim \sum_{c=1}^{C} w_c \cdot \mathcal{N}(g_c(z_t), \Sigma_c), \tag{2}$$

where $z_t \in \mathcal{Z}$ and $x_t \in \mathcal{X}$. From the Bayesian inference viewpoint, we are interested in the posterior $p(z_t|x_t, z_{t-1})$. Since the functions f and g are non-linear, the computation of this posterior is intractable (unlike an LG-SSM, such as Kalman Filter [65]). Therefore, we utilize variational inference to approximate the posterior by assuming the surrogate posterior $q(z)$ has a Gaussian form, and minimize the Variational Free Energy (VFE) [18]. The minimization of VFE (in equation 3) minimizes the KL-divergence between the approximate posterior $q(z)$ and the true posterior $p(z_t|x_t, z_{t-1})$. Derivation 2 of Variational Free Energy is provided in appendix A.1

$$\underbrace{\sum_{t=1}^{T} \mathbb{E}_q \left[\log \left(\frac{q(z_t)}{p(x_t, z_t|z_{t-1})} \right) \right]}_{\text{Variational Free Energy}} = \underbrace{\sum_{t=1}^{T} \mathbb{E}_q \left[\frac{1}{\log(p(z_t|z_{t-1}))} \right]}_{\text{Latent State Error}} + \underbrace{\sum_{t=1}^{T} \mathbb{E}_q \left[\frac{1}{\log(p(x_t|z_t))} \right]}_{\text{Observation Error}} - \mathcal{H}_q,$$

$$\tag{3}$$

where $\mathbb{E}_q \equiv \mathbb{E}_{z_t \sim q(z_t)}$ and $\mathcal{H}_q$ is the Entropy of the approximate posterior $q(z)$. The assumption of Gaussian forms for the latent and observable states can further simplify the negative log-likelihood

4

terms (i.e., Latent State Accuracy and Observation Accuracy) to prediction errors. This learning objective encourages the approximate posterior $q(z)$ to assign a high probability to states that explain the observations well and follow the latent dynamics of the system. We minimize the prediction errors (i.e., learn the transition and emission functions) through Hebbian rules as shown in equations 7 and 8.

Theorem 1 *Assume the likelihood $p(\boldsymbol{x}_t|\boldsymbol{z}_t)$ in eqn 3 represents multiple possibilities using a Gaussian Mixture Model (GMM) conditioned on the latent state $\boldsymbol{z}_t$, as shown in eqn 2. The maximization of such log-likelihood function (i.e., $\frac{\partial}{\partial \boldsymbol{x}} \log p(\boldsymbol{x}_t|\boldsymbol{z}_t)$) w.r.t a query observation state $\boldsymbol{x}$ is equivalent to the Hopfield recall function (i.e., eqn 1) with the means of the GMM representing the attractors of a Hopfield model. Formally, the weighted average of the GMM means (i.e., $\{\boldsymbol{\mu}_c\}_{c=1}^{C}$), with the weights being a similarity measure, maximizes the log-likelihood of $\boldsymbol{x}$ under the mixture model.*

$$\boldsymbol{x} = \sum_{c=1}^{C} \underbrace{\frac{w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}}}_{\text{similarity score}} \cdot \underbrace{\boldsymbol{\mu}_c}_{\text{projection}} \tag{4}$$

Proof: See derivation 3 in appendix A.2 for the full proof.

3.2 Preliminaries and Notations

Sparse Distributed Representation (SDR)   Inspired by the sparse coding principles observed in the brain, SDRs encode information using a small set of active neurons in high dimensional binary representation. We adopt SDRs as a more biologically plausible cell assembly representation [30]. SDRs have desirable characteristics, such as a low chance of false positives and collisions between multiple SDRs and high representational capacity [5] (More on SDRs in Appendix G). An SDR is parameterized by the total number of neurons N and the number of active neurons W. The ratio $S = W/N$ denotes the SDR sparsity. A 1-dimensional SDR $\boldsymbol{x}$ can be indexed as $\boldsymbol{x}^i \in \{0, 1\}$, whereas a 2-dimensional SDR $\boldsymbol{z}$ can be indexed as $\boldsymbol{z}^{ij} \in \{0, 1\}$. To identify the active neurons, we define the function $I : \{0, 1\}^N \mapsto \mathbb{N}^W$ to represent the indices of the active neurons in an SDR $\boldsymbol{x}$ as $I(\boldsymbol{x}) = \{i | \boldsymbol{x}^i = 1\}$.

Context as Orthogonal Dimension   We transform the high-order Markov dependencies between observation states into a first-order Markov chain of latent states by storing context information in those latent states. The latent states SDRs, $\boldsymbol{z} \in \{0, 1\}^{N_c \times N_k}$, are represented with two orthogonal dimensions, where content information about the input is stored in one dimension with size N_c, while context related information is stored in an orthogonal dimension with size N_k. Therefore, the projection of the latent state $\boldsymbol{z}$ on the first dimension (i.e., $\downarrow \boldsymbol{z}$) removes all context information from the state. In contrast, adding context information to an observation state $\boldsymbol{x}$ expands the dimensionality of the state (i.e., $\uparrow \boldsymbol{x}$) such that context can be encoded without affecting its content. Competitive learning is enforced on the context dimension through lateral inhibition, effectively minimizing the collisions between contexts of multiple SDRs. We define a projection operator $\downarrow$: $\{0, 1\}^{N_c \times N_k} \mapsto \{0, 1\}^{N_c}$. Additionally, we define a projection operator $\uparrow$: $\{0, 1\}^{N_c} \mapsto \{0, 1\}^{N_c \times N_k}$ for 1-dimensional SDRs (i.e., $\boldsymbol{x}$) as shown in equation 5. A detailed list of notations is provided in the supplementary material (Table 1).

$$(\downarrow \boldsymbol{z})^i = \begin{cases} 1 & \text{if } \exists j \text{ s.t. } \boldsymbol{z}^{ij} = 1, \\ 0 & \text{otherwise,} \end{cases} , \quad (\uparrow \boldsymbol{x})^{ij} = \begin{cases} 1 & \text{if } \boldsymbol{x}^i = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

3.3 Sequence Learning

Given a sequence of $T + 1$ SDR patterns $[\boldsymbol{x}_t]_{t=1}^{T+1}$, where $\boldsymbol{x}_t \in \{0, 1\}^{N_c}$, the sequence can be learned by modeling the context-dependent transitions between consecutive inputs within the sequence. We define learnable weight parameters for transition and emission functions, $\boldsymbol{A} \in \mathbb{R}^{N_c N_k \times N_c N_k}, \boldsymbol{B} \in \mathbb{R}^{N_c \times N_c}$ respectively. A single latent state transition is defined as $\hat{\boldsymbol{z}}_t = \delta(\boldsymbol{A} \cdot \boldsymbol{z}_{t-1}) = \delta(\boldsymbol{a}_t)$, where δ is a threshold function transforming the logits $\boldsymbol{a}_t$ to the predicted SDR state $\hat{\boldsymbol{z}}_t$. The full sequence learning algorithm is provided in algorithm 1.

5

Algorithm 1 : Sequence Learning. Given a sequence $\boldsymbol{x}$ of T+1 patterns, this algorithm learns the Transition and Emission synaptic weights ($\boldsymbol{A}$ and $\boldsymbol{B}$). Fixed start context $m(\boldsymbol{a}_0)$ is initialized for all learned sequences.

1: **procedure** TRAIN($\boldsymbol{x}$)
2:   $\boldsymbol{z}_0 = (\uparrow \boldsymbol{x}_0) \cap m(\boldsymbol{a}_0)$ ▷ Eqn. 6
3:   **for** $t = 1$ to $T + 1$ **do**
4:     $\boldsymbol{a}_t = \boldsymbol{A} \cdot \boldsymbol{z}_{t-1}$
5:     $\boldsymbol{z}_t = (\uparrow \boldsymbol{x}_t) \cap m(\boldsymbol{a}_t)$
6:     $\boldsymbol{A} = \boldsymbol{A} + \Delta \boldsymbol{A}$ ▷ Update $\boldsymbol{A}$ via Eqn. 7
7:     $\hat{\boldsymbol{z}}_t = \delta(\boldsymbol{A} \cdot \boldsymbol{z}_{t-1})$
8:     $\boldsymbol{B} = \boldsymbol{B} + \Delta \boldsymbol{B}$ ▷ Update $\boldsymbol{B}$ via Eqn. 8
9:   **end for**
10: **end procedure**

Algorithm 2 : Sequence Generation. Given a noisy sequence (i.e., online), or the first input in a sequence (i.e., offline). The model uses the learned functions $\boldsymbol{A}$ and $\boldsymbol{B}$ to generate the full sequence. $\sim$ denotes sampled from (eqn 9).

1: **procedure** GENERATE($\boldsymbol{x}_0$ or $\boldsymbol{x}$)
2:   $\boldsymbol{z}_0 = (\uparrow \boldsymbol{x}_0) \cap m(\boldsymbol{a}_0)$ ▷ Eqn. 6
3:   **for** $t = 1$ to $T + 1$ **do**
4:     $\boldsymbol{a}_t = \boldsymbol{A} \cdot \boldsymbol{z}_{t-1}$
5:     $\tilde{\boldsymbol{z}}_t = \delta(\boldsymbol{a}_t)$
6:     $\tilde{\boldsymbol{x}}_t = \begin{cases} \boldsymbol{x}_t & \text{online} \\ \sim (\downarrow \hat{\boldsymbol{z}}_t) & \text{offline} \end{cases}$ ▷ Eqn. 9
7:     **for** i=1 to iters **do**   ▷ Attractors iterations
8:       $\tilde{\boldsymbol{x}}_t = \delta(\boldsymbol{B} \cdot \tilde{\boldsymbol{x}}_t) \cap (\downarrow \hat{\boldsymbol{z}}_t)$
9:     **end for**
10:    $\boldsymbol{z}_t = (\uparrow \tilde{\boldsymbol{x}}_t) \cap m(\boldsymbol{a}_t)$
11: **end for**
12: **end procedure**

Context Encoding through Competitive Learning Every observation $\boldsymbol{x}_t$ contains only content information about the input; we embed the observation with context by expanding the state with an orthogonal dimension (i.e., $\uparrow \boldsymbol{x}_t$), which activates all neurons in the minicolumns at the indices $I(\boldsymbol{x}_t)$. Then, for each active minicolumn, the neuron in a predictive state (i.e., higher than the prediction threshold) fires and inhibits all the other neurons in the same minicolumn from firing (i.e., lateral inhibition), as shown in Equation 6. If none - or more than one - of the neurons are in a predictive state, random Gaussian noise (ϵ) acts as a tiebreaker to select a context neuron. We *do not* allow multiple neurons to fire in the same minicolumn, which is different from HTM [25], where multiple cells can fire in any minicolumn (e.g., bursting).

$$m(\boldsymbol{a}_t)^{ij} = \begin{cases} 1 & \text{if } \boldsymbol{a}_t^{ij} = \max(\{\delta(\boldsymbol{a}_t^{ij}) + \epsilon\}_{j=0}^{N_k}), \\ 0 & \text{otherwise,} \end{cases}, \quad \boldsymbol{z}_t = (\uparrow \boldsymbol{x}_t) \cap m(\boldsymbol{a}_t) \qquad (6)$$

State Transition Learning The transition between latent states is learned through local computations with Hebbian-based rules. We modify the synaptic weights $\boldsymbol{A}$ to model the transition between pre-synaptic neurons $\boldsymbol{z}_{t-1}$ and post-synaptic neurons $\boldsymbol{z}_t$. Only the synapses with active pre-synaptic neurons are modified. The weights operate on context-dependant latent states (i.e., $\boldsymbol{z}_{t-1} \rightarrow \boldsymbol{z}_t$); thus, the learning of one transition does not overwrite previously learned transition of different contexts, regardless of the input contents (i.e., $\boldsymbol{x}_{t-1}$). We use the learning constant coefficients η_A^+ and η_A^- to independently control learning and forgetting behavior, as shown in equation 7. A lower η_A^- encourages learning multiple possibilities by slowing down the forgetting behavior.

$$\Delta \boldsymbol{A} = \underbrace{\eta_A^+ \cdot \boldsymbol{z}_{t-1} \cdot \boldsymbol{z}_t^T}_{\Delta \boldsymbol{A}_{\text{increase}}} + \underbrace{\eta_A^- \cdot \boldsymbol{z}_{t-1} \cdot (1 - \boldsymbol{z}_t)^T}_{\Delta \boldsymbol{A}_{\text{decrease}}} \qquad (7)$$

Contrastive Attractors Formation The attractors are formed in an online manner by contrasting the input observation $\boldsymbol{x}_t$ to the predicted union set of possibilities $\downarrow \hat{\boldsymbol{z}}_t$. The goal is to learn excitatory synapses between active neurons of $\boldsymbol{x}_t$, and bidirectional inhibitory synapses between $\boldsymbol{x}_t$ and the union set of predicted possibilities *excluding* the $\boldsymbol{x}_t$ possibility (i.e., $(\downarrow \hat{\boldsymbol{z}}_t) \setminus \boldsymbol{x}_t)$), as shown in equation 8.

$$\Delta \boldsymbol{B} = \underbrace{\eta_B^+ \cdot \boldsymbol{x}_t \cdot \boldsymbol{x}_t^T}_{\Delta \boldsymbol{B}_{\text{increase}}} + \underbrace{\eta_B^- \cdot [\boldsymbol{x}_t \cdot ((\downarrow \hat{\boldsymbol{z}}_t) \setminus \boldsymbol{x}_t)^T + ((\downarrow \hat{\boldsymbol{z}}_t) \setminus \boldsymbol{x}_t) \cdot \boldsymbol{x}_t^T]}_{\Delta \boldsymbol{B}_{\text{decrease}}} \qquad (8)$$

3.4 Sequence Generation

After learning one or multiple sequences using algorithm 1, we use algorithm 2 to generate sequences. First, we define two generative tasks: online and offline. In online sequence generation, a noisy

6
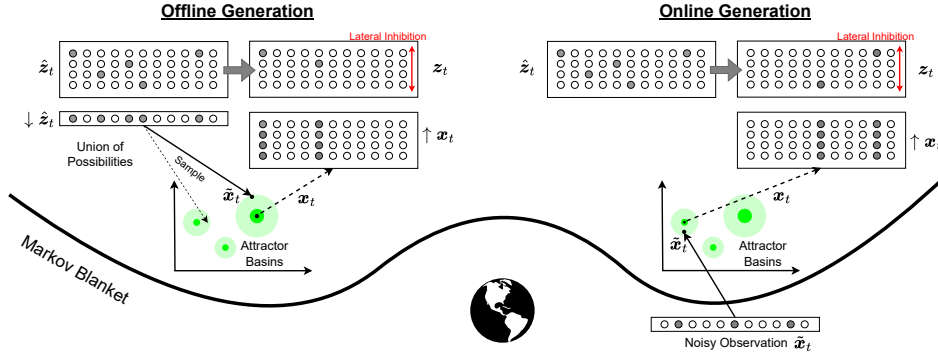
Figure 2: Sequence Generation. (Left): Offline generation by sampling a single possibility (i.e., attractor point) from a union of predicted possibilities. (Right): Online generation by removing noise from an observation using the prior beliefs about the observed state. Markov Blanket separates the agent's latent variables from the world's observable states.

version of the sequence is provided as input, and the model is expected to generate the original learned sequence. In offline sequence generation, the model is only provided with the first input, and it is expected to generate the entire sequence auto-regressively. For cases with equally valid future predictions (e.g., "a" and "e" after "TH" in "THAT" and "THEY"), the model is expected to stochastically generate either one of the possibilities (i.e., "THAT" or "THEY"). The online generation task is a more challenging extension of the online recall task in [62], where the noise-free inputs are provided, and the model only makes a 1-step prediction into the future. During offline sequence generation, the model randomly samples from the union set of predictions $\downarrow \hat{z}$ a single SDR with W active neurons (equation 9) to initialize the iterative attractor procedure. π denotes a random permutation function. This random permutation function allows the model to randomly generate a different sequence with every generation.

$$\tilde{x}^i = \begin{cases} 1 & \text{if } i \in \{\pi(I(\downarrow \hat{z}_t))^w\}_{w=0}^W, \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

4 Experiments

4.1 Evaluation and Metrics

Metrics To evaluate the similarity of two SDRs, we use the Jaccard Index (i.e., IoU)[1], which focuses on the active bits in sparse binary representations. Since the sparsity S of the representations can change across experiments and methods, we normalize the IoU by the expected IoU (Derived in Theorem 2 in Appendix A.3) of two random SDRs at their specified sparsities. The normalized IoU is computed as $\frac{\text{IoU}-\mathbb{E}[\text{IoU}]}{1-\mathbb{E}[\text{IoU}]}$. We use the Backward Transfer [38] metric in evaluating catastrophic forgetting. Mean Squared Error (MSE) is used to compare images for vision datasets.

Datasets We perform evaluations on synthetic and real datasets. The synthetic datasets allow us to manually control variables (e.g., sequence length, correlation, noise, input size) to better understand the models' behavior across various settings. Additionally, we evaluate on real datasets of various types (e.g., protein sequences, text, vision) to benchmark PAM's performance relative to other models on more challenging and real sequences. For all vision experiments, we use an SDR autoencoder to learn a mapping between images and SDRs (Details on the autoencoder are provided in Appendix C). We run all experiments for 10 different seeds and report the mean and standard deviation in all the figures. More experimental details and results are provided in Appendices C and E.

---

[1]In the context of SDRs, IoU refers to the number of active bits in the intersection SDR divided by the number of active bits in the union SDR

7
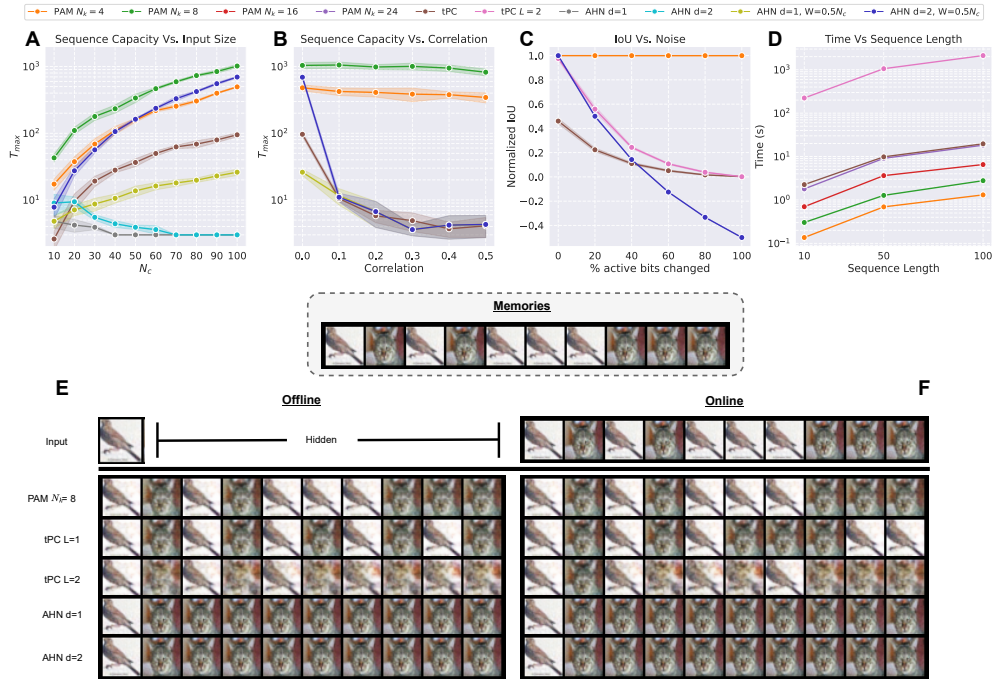
Figure 3: Quantitative results on (**A-B**) Offline sequence capacity, (**C**) Noise robustness, and (**D**) Time of sequence learning and recall. Qualitative results on highly correlated CIFAR sequence in (**E**) offline and (**F**) online settings. The mean and standard deviation of 10 trials are reported for all plots.

4.2 Results

We align our evaluation tasks with the desired characteristics of a biologically plausible sequence memory model, as listed in the introduction. We show that PAM outperforms current predictive coding and associative memory SoTA approaches on all tasks. Most importantly, PAM is capable of long context encoding, multiple possibilities generation, and learning continually and efficiently while avoiding catastrophic forgetting. These tasks pose numerous significant challenges to other methods.

Offline Sequence Capacity We evaluate the models' capacity to learn long sequences by varying the input size N_c, model parameters (e.g., N_k), and sequence correlation. The correlation is increased by reducing the number of unique patterns (i.e., vocab) used to create a sequence of length T. Correlation is computed as $1.0 - \frac{\text{vocab}}{T}$. In Figure 3 **A**, we vary the input size N_c and ablate the models to find the maximum sequence length to be encoded and retrieved, in an *offline* manner, with a Normalized IoU higher than 0.9. We set the number of active bits W to be 5 unless otherwise specified. Results show that Hopfield Networks (HN) fail to learn with sparse representations; therefore, we use W of $0.5N_c$ only for HN and normalize the IoU metric accordingly. PAM's capacity significantly increases with context neurons N_k, as expected. HN's capacity also increases with the polynomial degree d of its separation function; however, as shown in Figure 3 **B**, the capacity sharply drops as correlation increases. PAM retains its capacity with increasing correlation, reflecting its ability to encode context in long sequences (i.e., high-order Markov memory). This context encoding property is also demonstrated in the qualitative CIFAR [33] results in Figure 3 **E** and **F**, where a short sequence of images with high correlation is used. The model must uniquely encode the context to correctly predict at every step in the sequence. While PAM correctly predicts the full context, single-layer tPC learns to indefinitely alternate between the patterns, while two-layered tPC attempts to average its predictions. AHN shows similar low performance and failure mode as in [62].

Catastrophic Forgetting To assess the model's performance in a continual learning setup, we sequentially train each model on multiple sequences and compute the Backward Transfer (BWT) [38]

8
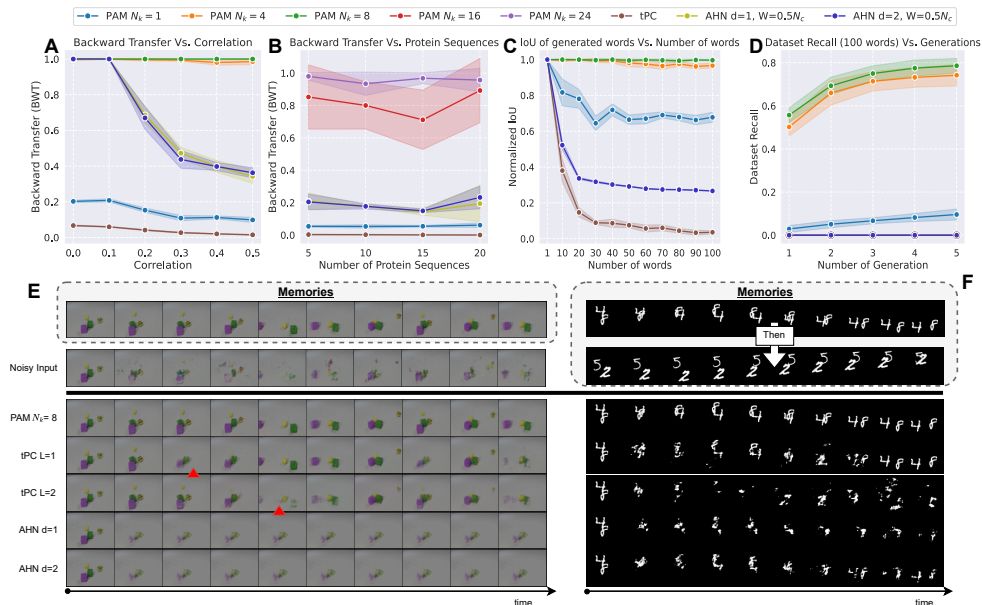
Figure 4: Qualitative results on (**A**) synthetic and (**B**) protein sequences backward transfer, and (**C-D**) multiple possibilities generation on text datasets. Qualitative results on (**E**) noise robustness on CLEVRER sequence, and (**F**) catastrophic forgetting on Moving MNIST dataset. ▲ highlights the first frame with significant error. The mean and standard deviation of 10 trials are reported for all plots.

metric by reporting the average normalized IoU on previously learned sequences after learning a new one. In Figure 4 **A**, we report BWT for 50 synthetically generated sequences with varying correlations. AHN can avoid catastrophic forgetting when the patterns are not correlated, whereas tPC fails to retain previous knowledge regardless of the correlation value. PAM, with high enough context N_k, does not overwrite or forget previously learned sequences after learning new ones but performs poorly when $N_k = 1$, as expected. We repeat the experiment on more challenging sequences from ProteinNet [7], which contains highly correlated (> 0.9) and long sequences (details in appendix). The results in Figure 4 **B** show a similar trend with PAM requiring more context neurons N_k to handle the more challenging data. Qualitative results on moving-MNIST [60] in Figure 4 **F** further demonstrate the catastrophic forgetting challenge where the learning of the second sequence overwrites the learned sequence. PAM successfully retrieves the previously learned sequence while the other models fail.

Multiple Possibilities Generation   In addition to accurately encoding input contexts, PAM is designed to represent multiple valid possibilities and sample a single possibility. We perform evaluation on a dataset of four-letter English words (details in appendix), which includes many possible future completions (e.g., "th[is]", "th[at]", "th[em]", etc.) We train PAM on the list of letters sequentially (i.e., one word at a time); the other methods are trained in a simpler batched setup as in [62] because they suffer from catastrophic forgetting. This puts PAM at a disadvantage, but as shown in Figure 4 **C**, PAM still significantly outperforms the other methods in accurately generating valid words (high average IoU) in an offline manner. Both tPC and AHN fail to generate meaningful words when trained on sequences with multiple future possibilities. Figure 4 **D** further demonstrates the stochastic generative property of PAM. We show PAM's ability to recall more of the dataset as it repeats the generation process, whereas PC and AHN fail in the dataset recall task.

Online Noise Robustness   The online generation ability of PAM shown in Figure 2 allows the model to use the learned attractors to clean the noisy observations before using them as inputs to the predictor. This step allows the model to use its prior belief about future observations to modify the noisy inputs. In Figure 3 **C**, we evaluate the models' performances by changing a percentage of the active bits during online generation. PAM is able to completely replace the noisy input with its prior belief if it does not exist in the predicted set of possibilities $\hat{z}$. In contrast, the other methods use the noisy inputs, thus hindering their performances. We provide qualitative results on CLEVRER [68]

9

in Figure 4 **E**; PAM retrieves the original sequence despite getting noisy inputs (40% noise), and outperforms the other models. Interestingly, tPC performs reasonably well on this task despite the added noise.

Efficiency We report, in Figure 3 **D**, the time each model requires to learn and recall a sequence. For this study, we use input size $N_c = 100$ and vary the sequence length. PAM operates entirely on *CPU*. The results show that a single-layer tPC model requires more time than all PAM variants ($N_k \leq 24$). Additionally, a two-layered tPC requires two to three orders of magnitude more time than PAM or single-layered tPC, significantly limiting its scalability and practicality when applied to real data with long sequences.

5   Limitations

PAM requires that its representations be sparse and binary (i.e., SDRs) in order to represent multiple possibilities as a union of SDRs with minimal overlap. Therefore, PAM cannot be directly applied to images in the input space like Dense Hopfield Networks. We argue that the neocortex encodes sensory input into SDRs for processing and does not operate directly on the input in its dense representation. Methods that operate directly on dense representations (e.g., images) are arguably less biologically plausible as the neocortex uses sparse binary representations (i.e., cell assemblies with a small fraction of firing neurons) to store and manipulate information. This paper focuses on learning multiple sequences without catastrophic forgetting and stochastically generating multiple possibilities efficiently, and we assume the sequence is provided in SDR format. Additionally, methods that operate on the input directly face challenges when the input is naturally sparse (see Figure 3 **A**). Therefore, it is useful to encode the input into a representation with fixed sparsity before applying sequential memory learning algorithms. In future work, we plan to investigate how to encode high dimensional complex inputs (e.g., images) in a compositional part-whole structure of SDRs where we can apply PAM at different levels of abstraction.

6   Conclusion

We proposed *PAM*, a biologically plausible generative model inspired by neuroscience findings and theories of cognition. We demonstrate that PAM is capable of encoding unique contexts with tremendous scalable capacity that is not affected by sequence correlation or noise. PAM is a generative model; it can represent multiple possibilities as a union of SDRs (a property of sparsity) and sample single possibilities, thus predicting multiple steps in the future despite multiple possible continuations. We also show that PAM does not suffer from catastrophic forgetting as it learns multiple sequences. PAM is trained using local computations through Hebbian rules and runs efficiently on CPUs. Future directions include hierarchical sensory processing and higher-order sparse predictive models.

10

Acknowledgements

This research was supported by the US National Science Foundation Grants CNS 1513126 and IIS 1956050.

References

[1] Sathyanarayanan Aakur and Sudeep Sarkar. Action localization through continual predictive learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 300–317. Springer, 2020.

[2] Sathyanarayanan N Aakur and Sudeep Sarkar. A perceptual prediction framework for self supervised event segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1197–1206, 2019.

[3] Valentin Afraimovich, Xue Gong, and Mikhail Rabinovich. Sequential memory: Binding dynamics. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(10), 2015.

[4] Subutai Ahmad and Jeff Hawkins. Properties of sparse distributed representations and their application to hierarchical temporal memory. *arXiv preprint arXiv:1503.07469*, 2015.

[5] Subutai Ahmad and Jeff Hawkins. How do neurons operate on sparse distributed representations? a mathematical theory of sparsity, neurons and active dendrites. *arXiv preprint arXiv:1601.00720*, 10, 2016.

[6] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[7] Mohammed AlQuraishi. Proteinnet: a standardized data set for machine learning of protein structure. *BMC bioinformatics*, 20:1–10, 2019.

[8] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023.

[9] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

[10] Adrien Bardes, Jean Ponce, and Yann LeCun. Mc-jepa: A joint-embedding predictive architecture for self-supervised learning of motion and content features. *arXiv preprint arXiv:2307.12698*, 2023.

[11] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[12] Hamza Chaudhry, Jacob Zavatone-Veth, Dmitry Krotov, and Cengiz Pehlevan. Long sequence hopfield memory. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.

[14] Nikolaos Chrysanthidis, Florian Fiebig, and Anders Lansner. Introducing double bouquet cells into a modular cortical associative memory model. *Journal of computational neuroscience*, 47(2):223–230, 2019.

[15] Martin A Conway. Episodic memories. *Neuropsychologia*, 47(11):2305–2313, 2009.

[16] Peter Elias. Predictive coding–i. *IRE transactions on information theory*, 1(1):16–24, 1955.

[17] Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127–138, 2010.

[18] Karl Friston, Jérémie Mattout, Nelson Trujillo-Barreto, John Ashburner, and Will Penny. Variational free energy and the laplace approximation. *Neuroimage*, 34(1):220–234, 2007.

[19] Karl J Friston, N Trujillo-Barreto, and Jean Daunizeau. Dem: a variational treatment of dynamic systems. *Neuroimage*, 41(3):849–885, 2008.

11

[20] Dileep George, Rajeev V Rikhye, Nishad Gothoskar, J Swaroop Guntupalli, Antoine Dedieu, and Miguel Lázaro-Gredilla. Clone-structured graph representations enable flexible learning and vicarious evaluation of cognitive maps. *Nature communications*, 12(1):2392, 2021.

[21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[22] Patrick Haggard. Planning of action sequences. *Acta Psychologica*, 99(2):201–215, 1998.

[23] Kuan Han, Haiguang Wen, Yizhen Zhang, Di Fu, Eugenio Culurciello, and Zhongming Liu. Deep predictive coding network with local recurrent processing for object recognition. *Advances in neural information processing systems*, 31, 2018.

[24] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[25] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, 10:174222, 2016.

[26] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[27] Toshihiko Hosoya, Stephen A Baccus, and Markus Meister. Dynamic predictive coding by the retina. *Nature*, 436(7047):71–77, 2005.

[28] Jerry Yao-Chieh Hu, Donglin Yang, Dennis Wu, Chenwei Xu, Bo-Yu Chen, and Han Liu. On sparse modern hopfield model. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] Hiroshi Kage. Implementing associative memories by echo state network for the applications of natural language processing. *Machine Learning with Applications*, 11:100449, 2023.

[30] Pentti Kanerva. *Sparse distributed memory*. MIT press, 1988.

[31] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[32] Bart Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, man, and Cybernetics*, 18(1):49–60, 1988.

[33] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[34] Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

[35] Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

[36] Niels Leadholm, Marcus Lewis, and Subutai Ahmad. Grid cell path integration for movement-based visual object recognition. *arXiv preprint arXiv:2102.09076*, 2021.

[37] Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. 2022.

[38] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[39] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

[40] Henry Markram, Maria Toledo-Rodriguez, Yun Wang, Anirudh Gupta, Gilad Silberberg, and Caizhi Wu. Interneurons of the neocortical inhibitory system. *Nature reviews neuroscience*, 5(10):793–807, 2004.

[41] Beren Millidge, Tommaso Salvatori, Yuhang Song, Thomas Lukasiewicz, and Rafal Bogacz. Universal hopfield networks: A general framework for single-shot associative memory models. In *International Conference on Machine Learning*, pages 15561–15583. PMLR, 2022.

[42] Ramy Mounir, Sathyanarayanan Aakur, and Sudeep Sarkar. Self-supervised temporal event segmentation inspired by cognitive theories. In *Advanced Methods and Deep Learning in Computer Vision*, pages 405–448. Elsevier, 2022.

12

[43] Ramy Mounir, Roman Gula, Jörn Theuerkauf, and Sudeep Sarkar. Spatio-temporal event segmentation for wildlife extended videos. In *International Conference on Computer Vision and Image Processing*, pages 48–59. Springer, 2021.

[44] Ramy Mounir, Ahmed Shahabaz, Roman Gula, Jörn Theuerkauf, and Sudeep Sarkar. Towards automated ethogramming: Cognitively-inspired event segmentation for streaming wildlife video monitoring. *International journal of computer vision*, 131(9):2267–2297, 2023.

[45] Ramy Mounir, Sujal Vijayaraghavan, and Sudeep Sarkar. Streamer: Streaming representation learning and event segmentation in a hierarchical manner. *Advances in Neural Information Processing Systems*, 36, 2024.

[46] J O'Neal. Entropy coding in speech and television differential pcm systems (corresp.). *IEEE Transactions on Information Theory*, 17(6):758–761, 1971.

[47] Alex Ororbia, Ankur Mali, C Lee Giles, and Daniel Kifer. Lifelong neural predictive coding: Learning cumulatively online without forgetting. *Advances in Neural Information Processing Systems*, 35:5867–5881, 2022.

[48] Alexander Ororbia, Ankur Mali, C Lee Giles, and Daniel Kifer. Continual learning of recurrent neural networks by locally aligning distributed representations. *IEEE transactions on neural networks and learning systems*, 31(10):4267–4278, 2020.

[49] A Peters and C Sethares. The organization of double bouquet cells in monkey striate cortex. *Journal of neurocytology*, 26(12):779–797, 1997.

[50] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[51] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, et al. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2020.

[52] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[53] Edmund T Rolls. A computational theory of episodic memory formation in the hippocampus. *Behavioural brain research*, 215(2):180–196, 2010.

[54] Tommaso Salvatori, Ankur Mali, Christopher L Buckley, Thomas Lukasiewicz, Rajesh PN Rao, Karl Friston, and Alexander Ororbia. Brain-inspired computational intelligence via predictive coding. *arXiv preprint arXiv:2308.07870*, 2023.

[55] Tommaso Salvatori, Luca Pinchetti, Beren Millidge, Yuhang Song, Tianyi Bao, Rafal Bogacz, and Thomas Lukasiewicz. Learning on arbitrary graph topologies via predictive coding. *Advances in neural information processing systems*, 35:38232–38244, 2022.

[56] Tommaso Salvatori, Yuhang Song, Yujian Hong, Lei Sha, Simon Frieder, Zhenghua Xu, Rafal Bogacz, and Thomas Lukasiewicz. Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, 34:3874–3886, 2021.

[57] Tommaso Salvatori, Yuhang Song, Beren Millidge, Zhenghua Xu, Lei Sha, Cornelius Emde, Rafal Bogacz, and Thomas Lukasiewicz. Incremental predictive coding: A parallel and fully automatic learning algorithm. *arXiv preprint arXiv:2212.00720*, 2022.

[58] Haim Sompolinsky and Ido Kanter. Temporal association in asymmetric neural networks. *Physical review letters*, 57(22):2861, 1986.

[59] Mandyam Veerambudi Srinivasan, Simon Barry Laughlin, and Andreas Dubs. Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 216(1205):427–459, 1982.

[60] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852. PMLR, 2015.

[61] Dan D Stettler, Aniruddha Das, Jean Bennett, and Charles D Gilbert. Lateral connectivity and contextual interactions in macaque primary visual cortex. *Neuron*, 36(4):739–750, 2002.

13

[62] Mufeng Tang, Helen Barron, and Rafal Bogacz. Sequential memory with temporal predictive coding. *Advances in Neural Information Processing Systems*, 36, 2024.

[63] Mufeng Tang, Tommaso Salvatori, Beren Millidge, Yuhang Song, Thomas Lukasiewicz, and Rafal Bogacz. Recurrent predictive coding models for associative memory employing covariance learning. *PLoS computational biology*, 19(4):e1010719, 2023.

[64] Endel Tulving et al. Episodic and semantic memory. *Organization of memory*, 1(381-403):1, 1972.

[65] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[66] James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.

[67] Dennis Wu, Jerry Yao-Chieh Hu, Weijian Li, Bo-Yu Chen, and Han Liu. Stanhop: Sparse tandem hopfield model for memory-enhanced time series prediction. *arXiv preprint arXiv:2312.17346*, 2023.

[68] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations*, 2019.

[69] Jinsoo Yoo and Frank Wood. Bayespcn: A continually learnable predictive coding associative memory. *Advances in Neural Information Processing Systems*, 35:29903–29914, 2022.

[70] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.

14

Supplementary Material

A   Theorems and Derivations

A.1   Variational Free Energy

Predictive Coding   Consider a hierarchical generative model with hidden states $\{z\}_{l=0}^L$, where $l \leq L$ denotes the level in the hierarchy. The conditional probability $p(z_l|z_{l+1})$ is assumed to be a multivariate Gaussian distribution with its mean calculated as a function f_{l+1} of the higher-level hidden representation z_{l+1} and covariance Σ_l as shown in equation 10.

$$p(z_l|z_{l+1}) = \mathcal{N}(f_{l+1}(z_{l+1}), \Sigma_l) \tag{10}$$

The goal is to calculate the posterior of hidden states given an observation x, (i.e., $P(\{z\}_{l=0}^L|x)$). Since the prediction function f contains a non-linear activation, we cannot analytically compute the posterior and we have to approximate it with a surrogate posterior (i.e., $q(\{z\}_{l=0}^L)$ by maximizing the Evidence Lower Bound (ELBO). We apply the mean field approximation to factorize this joint posterior probability into conditionally independent posteriors $\{q(z_l)\}_{l=0}^L$, and apply the Laplace approximation to use Gaussian forms for the approximate distribution [19, 18, 54]. Through these approximations, we can maximize the ELBO, or equivalently minimize the Variational Free Energy, in equation 11.

$$\underbrace{\mathbb{E}_{z \sim q(z)}[\log(\frac{q(z)}{p(x,z)})]}_{\text{Variational Free Energy}} = \underbrace{\mathbb{E}_{z \sim q(z)}[\log(\frac{q(z)}{p(z)})]}_{D_{KL}(q||p)} + \underbrace{\mathbb{E}_{z \sim q(z)}[\log(\frac{1}{p(x|z)})]}_{\text{Accuracy}} \tag{11}$$

The variational free energy can be reduced to minimizing the negative log-likelihood (Accuracy term), which is simply the prediction error when the likelihood is assumed to take a Gaussian Form. Therefore, minimizing the prediction error reduces to the sum of the squared prediction error of every neuron.

Derivation 1 *Variational Free Energy derivation for the predictive coding objective function in equation 11. We approximate the true posterior $p(z|x)$ with a surrogate posterior $q(z)$. The objective is to minimize the reverse KL divergence $D_{KL}(q(z)||p(z|x))$.*

$$
\begin{aligned}
D_{KL}(q(z)||p(z|x)) &= \mathbb{E}_{z \sim q(z)}[\log \frac{q(z)}{p(z|x)}] &&\textit{(KL Divergence definition)} \\[6pt]
&= \mathbb{E}_{z \sim q(z)}[\log \frac{q(z)p(x)}{p(x|z)p(z)}] &&\textit{(Bayes Theorem)} \\[6pt]
&= \mathbb{E}_{z \sim q(z)}[\log \frac{q(z)}{p(x|z)p(z)}] + \mathbb{E}_{z \sim q(z)}[\log p(x)] &&\textit{(Linearity of expectations)} \\[6pt]
D_{KL}(q(z)||p(z|x)) &= \underbrace{\mathbb{E}_{z \sim q(z)}[\log \frac{q(z)}{p(x|z)p(z)}]}_{\textit{Variational Free Energy}} + \log p(x) &&\textit{(Evidence does not depend on } q(x)\textit{)}
\end{aligned}
$$

To minimize the KL divergence, we can minimize the Variational Free energy instead because the Evidence term ($\log p(x)$) is a constant negative term. The Variational Free Energy can be further simplified as follows:

15

$$\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log \frac{q(\boldsymbol{z})}{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}] = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log \frac{1}{p(\boldsymbol{x}|\boldsymbol{z})}] + \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z})}] \quad \textit{(Linearity of Expectations)}$$

$$\underbrace{\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log \frac{q(\boldsymbol{z})}{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}]}_{\textit{Variational Free Energy}} = \underbrace{\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z})}[\log \frac{1}{p(\boldsymbol{x}|\boldsymbol{z})}]}_{\textit{Error}} + D_{KL}(q(\boldsymbol{z})||p(\boldsymbol{z})) \quad \textit{(KL Divergence definition)}$$

We arrive at equation 11. Minimizing the error term (i.e., negative log-likelihood) is equivalent to minimizing the Sum of Squared Error (SSE) when a Gaussian form is assumed for the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$.

Derivation 2 *Variational Free Energy derivation for a State Space Model (SSM) in equation 3. Latent states are denoted with $\boldsymbol{z}$, whereas observations are denoted with $\boldsymbol{x}$. We assume non-linear transition and emission function (i.e., f and g); therefore a variational approximation is needed to approximate the true posterior $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_t)$ with a surrogate posterior $q(\boldsymbol{z}_t)$. As in derivation 1, the goal is to minimize the divergence between the true posterior and the approximate posterior (i.e., $D_{KL}(q(\boldsymbol{z}_t)||p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_t)))$. Note that, for notation brevity, $\mathbb{E}_q \equiv \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}_t)}$.*

$$D_{KL}(q(\boldsymbol{z}_t)||p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_t)) = \mathbb{E}_q[\log \frac{q(\boldsymbol{z}_t)}{p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_t)}] \quad \textit{(KL Divergence definition)}$$

$$= \mathbb{E}_q[\log \frac{q(\boldsymbol{z}_t)p(\boldsymbol{x}_t|\boldsymbol{z}_{t-1})}{p(\boldsymbol{x}_t|\boldsymbol{z}_t, \boldsymbol{z}_{t-1})p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}] \quad \textit{(Bayes Theorem)}$$

$$= \mathbb{E}_q[\log \frac{q(\boldsymbol{z}_t)p(\boldsymbol{x}_t|\boldsymbol{z}_{t-1})}{p(\boldsymbol{x}_t|\boldsymbol{z}_t, \cancel{\boldsymbol{z}_{t-1}})p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}] \quad \textit{(Conditional Independence)}$$

$$D_{KL}(q(\boldsymbol{z}_t)||p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_t)) = \underbrace{\mathbb{E}_q[\log \frac{q(\boldsymbol{z}_t)}{p(\boldsymbol{x}_t|\boldsymbol{z}_t)p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}]}_{\textit{Variational Free Energy}} + \mathbb{E}_q[\log p(\boldsymbol{x}_t|\boldsymbol{z}_{t-1})] \quad \textit{(Linearity of expectations)}$$

We can minimize the Variational Free Energy term, which reduces to the log-likelihood of two prediction error terms and the negative entropy of the approximate posterior $q(\boldsymbol{z}_t)$ as shown below.

$$\mathbb{E}_q[\log \frac{q(\boldsymbol{z}_t)}{p(\boldsymbol{x}_t|\boldsymbol{z}_t)p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}] = \underbrace{\mathbb{E}_q[\log \frac{1}{p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})}]}_{\textit{Latent State Error}} + \underbrace{\mathbb{E}_q[\log \frac{1}{p(\boldsymbol{x}_t|\boldsymbol{z}_t)}]}_{\textit{Observation Error}} - \underbrace{\mathbb{E}_q[\log \frac{1}{q(\boldsymbol{z}_t)}]}_{\textit{entropy } \mathcal{H}_q}$$

Minimizing the Variational Free Energy above forces $q(\boldsymbol{z}_t)$ to better approximate the true posterior.

A.2 Gaussian Mixture Model and Hopfield Recall

Derivation 3 *We derive theorem 1 which states that the maximization of the log-likelihood of $p(\boldsymbol{x}_t|\boldsymbol{z}_t)$ in the form of a Gaussian Mixture Model is equivalent to the recall function in Hopfield networks (i.e., eqn 1), where the means of the GMM (i.e., $\{\boldsymbol{\mu}_c\}_{c=1}^{C}$) represents the attractors of a Hopfield model. To maximize the log-likelihood, we compute its partial derivative with respect to $\boldsymbol{x}$.*

16

$$\frac{\partial}{\partial \boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{z}) = \frac{\partial}{\partial \boldsymbol{x}} [\log \sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)]$$

$$= \frac{1}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)} \cdot \frac{\partial}{\partial \boldsymbol{x}} \sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

$$= \frac{1}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)} \cdot \sum_{c=1}^{C} w_c \frac{\partial}{\partial \boldsymbol{x}} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

$$= \frac{1}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)} \cdot \sum_{c=1}^{C} w_c \cdot [-\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)] \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

$$= \frac{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot -\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}$$

By setting the partial derivative of the log-likelihood to 0, we can estimate the value of $\boldsymbol{x}$, which maximizes the function $\log p(\boldsymbol{x}|\boldsymbol{z})$.

$$\frac{\partial}{\partial \boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{z}) = 0$$

$$\frac{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot -\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)} = 0$$

$$\frac{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}\boldsymbol{x}}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)} = \frac{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}$$

Finally, we can rearrange the equation in terms of $\boldsymbol{x}$ and show that it is equivalent to the Hopfield recall function where the recall value $\boldsymbol{x}$ equals a weighted average of the attractors (i.e., means of GMM $\{\boldsymbol{\mu}_c\}_{c=1}^{C}$), with the weights being a similarity score function.

$$\boldsymbol{x} = \frac{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}}$$

$$\boldsymbol{x} = \sum_{c=1}^{C} \underbrace{\frac{w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}}{\sum_{c=1}^{C} w_c \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \cdot \boldsymbol{\Sigma}_c^{-1}}}_{\text{similarity score}} \cdot \underbrace{\boldsymbol{\mu}_c}_{\text{projection}}$$

A.3 Expected IoU of Random SDRs

Theorem 2 *Consider two SDRs with sparsity defined as random variables $p \sim \mathcal{U}(0, 1)$ and $q \sim \mathcal{U}(0, 1)$, the expected Jaccard Index (i.e., IoU) of the two random SDRs follows:*

$$\frac{pq}{p + q - pq}$$

Proof: Given the sparsity random variables of both SDRs (i.e., p and q) and the SDR size n, the number of active bits at the same location in both SDRs is equal to the joint probability of both SDRs being active multiplied by the SDR size (i.e., npq). The union of both SDRs is the total number of active bits minus the active bits in both SDRs, which is equal to $np + nq - npq$. Therefore, the expected intersection over union is $\frac{npq}{np+nq-npq} = \frac{pq}{p+q-pq}$

17
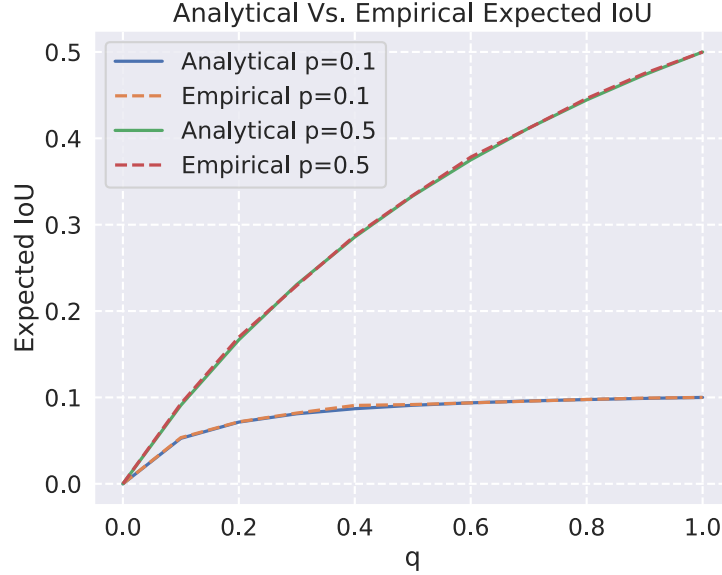
Figure 5: Empirical Validation of Theorem 2

Empirical Validation: We perform empirical validation of the above theorem as shown in figure 5. The sparsity of the first SDR is fixed at 0.1 and 0.5. We vary the sparsity of the second SDR between 0.0 and 1.0 in steps of 0.1 and calculate the average IoU over a population of 1000 pairs of SDR for every setting. Empirical results agree with the derived formulation in theorem 2.

B   Notations

The notations used in our paper are summarized in Table 1.

C   Datasets

Synthetic   For synthetic experiments, we generate SDRs with the specified size N_c and uniformly initialized active bits W to match the required sparsity S. In many of the experiments, N_c is set to 100 with 5 active bits unless otherwise specified. For Hopfield experiments, we set the sparsity to 50% to improve its performance.

Protein Sequences   We use the dataset ProteinNet 7 [7] to extract protein sequences. Each sequence consists of a chain of Amino Acids. In the dataset, there are only 20 different types of Amino Acids (i.e., vocabulary), creating long protein sequences with hundreds of Amino Acids. The dataset is reported in the fasta format, where each Amino Acid is represented with a single-letter code. We create a dictionary mapping from the Amino Acid types to random SDRs with $N_c = 100$ and $W = 5$ to train the models. When choosing the sequences, we ensure that the starting Amino Acid is unique for all the dataset sequences to avoid ambiguous predictions in the continual learning evaluation. A sample of the protein sequence is provided below:

MGAAASIQTTVNTLSERISSKLEQEANASAQTKCDIEIGNFYIRQNHGCN
LTVKNMCSADADAQLDAVLSAATETYSGLTPEQKAYVPAMFTAALNIQTS
VNTVVRDFENYVKQTCNSSAVVDNKLKIQNVIIDECYGAPGSPTNLEFIN
TGSSKGNCAIKALMQLTTKATTQIAPKQVAGTGVQFYMIVIGVIILAALF
MYYAKRMLFTSTNDKIKLILANKENVHWTTYMDTFFRTSPMVIATTDMQN

Text   To evaluate the generative ability of PAM, we use a dataset of the most frequently used English words. For preprocessing, we extract one hundred 4-letter words from the dataset and create a mapping dictionary from all the unique letters in the dataset to random SDRs with $N_c = 100$ and

18

Table 1: Table of Notations

Symbol	Description
A	Learnable transition weight matrix
B	Learnable emission weight matrix
x_t	Observation at time t
$\tilde{x}_t$	Noisy observation at time t during recall
z_t	Posterior Latent state after observing x_t (i.e., single possibility)
$\hat{z}_t$	Prior Latent state before observing x_t (i.e., multiple possibilities)
a_t	Predicted latent logits at time t (i.e., $A \cdot z_{t-1}$) before applying threshold
δ	Threshold function: $\mathbb{R}^{N_c \times N_k} \mapsto \{0,1\}^{N_c \times N_k}$ or $\mathbb{R}^{N_c} \mapsto \{0,1\}^{N_c}$
η	Hebbian learning strength for adjusting the synaptic weights
$\uparrow$	Projection operator adds context to a 1d observation state
$\downarrow$	Projection operator removes context from a 2d latent state
I	Function for computing the indices of active bits in an SDR
π	Random permutation function
N_c	Input size of a pattern
N_k	Number of neurons per minicolumn for context encoding
W	Number of active bits in a Sparse Distributed Representation (SDR)
S	Sparsity of SDR, calculated as W/N
T	Number of patterns in one sequence (i.e., sequence length)
d	Degree of polynomial in Hopfield separation function
L	Number of Layers used in temporal Predictive Coding (tPC)

$W = 5$ (except for AHN; $W = 0.5 N_c$). The dataset contains many words with ambiguous future possibilities. The selected words are provided below:

```
that  with  they  have  this  from  word  what  some  were
when  your  said  each  time  will  many  then  them  like
long  make  look  more  come  most  over  know  than  call
down  side  been  find  work  part  take  made  live  back
only  year  came  show  good  give  name  very  just  form
help  line  turn  much  mean  move  same  tell  does  want
well  also  play  home  read  hand  port  even  land  here
must  high  such  went  kind  need  near  self  head  page
grow  food  four  keep  last  city  tree  farm  hard  draw
left  late  real  life  open  seem  next  walk  ease  both
```

Vision In our experiment, we evaluate on sequences extracted from Moving MNIST [60], CLEVRER [68] as well as synthetically generated sequences of CIFAR [33] images. In order to convert images to SDRs and SDRs back to images while encoding semantics into the SDRs, we design an SDR AutoEncoder. The goal is to force the bottleneck representation of the autoencoder to become a sparse binary representation, where two visually similar images would result in two SDRs with a high overlap of active neurons. We simply design a CNN autoencoder with a 3-layer CNN encoder and 3-layer CNN decoder and apply top K binarization operation on the bottleneck embedding during training. The full architecture of the SDR autoencoder is shown in Figure 6.

In practice, we use a weighted average of the SDR and Dense representation to allow gradients of the reconstruction loss to freely propagate into the encoder. The weight of the SDR (i.e., α) is gradually and linearly increased (from 0.0 to 1.0) with the number of training epochs. This gradual increase is fundamental to the training of the SDR Autoencoder as it smooths the loss landscape and allows the model to distribute the active bits on the full SDR. The total mse loss becomes $\mathcal{L}_{enc} + \mathcal{L}_{recon}$. We use Adam optimizer with a learning rate of 1×10^{-4}. For Moving MNIST, we use a bottleneck embedding

19
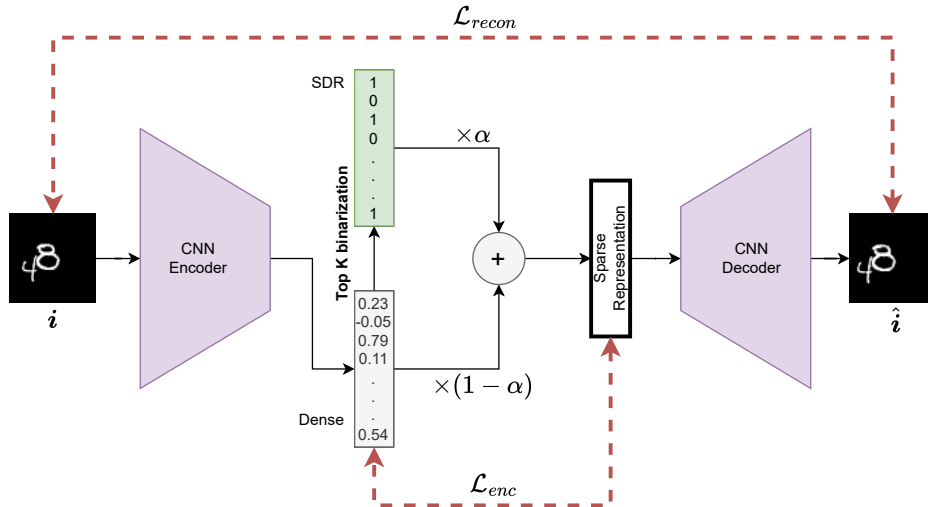
Figure 6: Architecture of the SDR Autoencoder

(i.e., N_c) of size 100 with 5 active bits, whereas, for more complex datasets (i.e., CLEVRER, CIFAR), we use an SDR of size 200 with 10 active bits. We show examples of the autoencoder reconstruction with full binary SDR (i.e., $\alpha = 1$) for all three datasets in Figure 7.
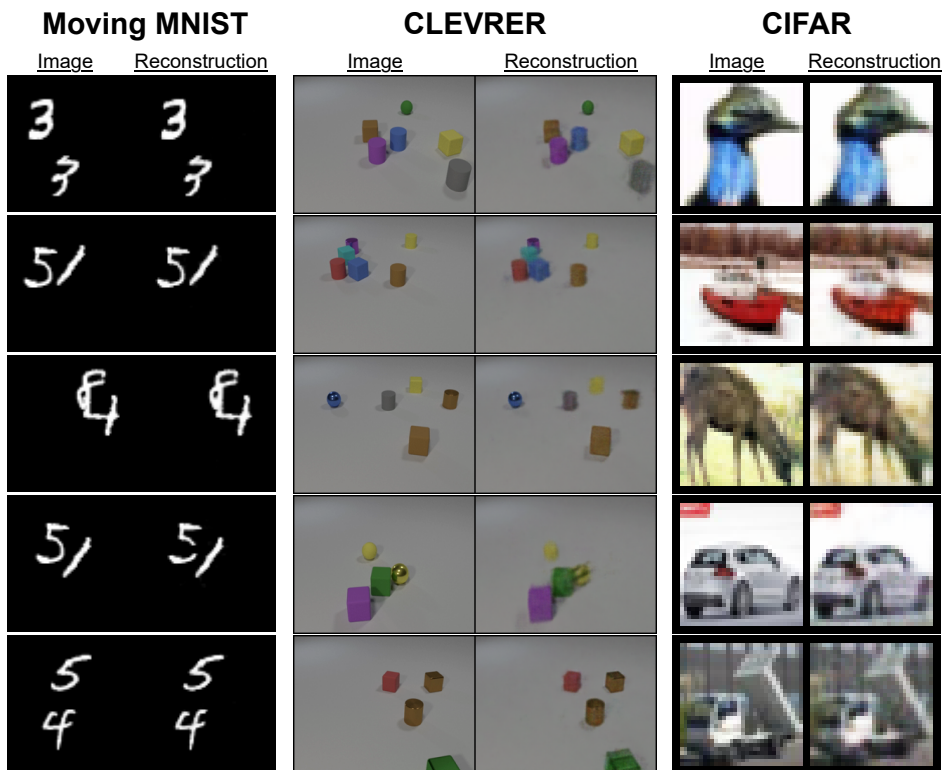
Figure 7: Examples of Autoencoder reconstructions from SDRs for all three datasets

20

D Implementation Details

In this section, we describe the implementation details and hyperparameters of each method. For each model, we optimize a single set of hyperparameters for all the experiments.

PAM   The neurons in both, transition and emission, functions are fully connected. We *do not* assume any of the weight matrices are symmetric. All synaptic weights are initialized by sampling from a normal distribution with a mean of 0.0 and a standard deviation of 0.1. All η^+ values in Equations 7 & 8 are set to 0.1. η_B^- is set to -0.1, while η_A^- is set to 0.0 to avoid forgetting previous possibilities when learning new transitions; PAM learns a union of possibilities. The threshold for the δ function is set as a function of the SDR sparsity. For the transition function, we use a threshold of $0.8W$, where W is the active number of bits in the latent state ($\boldsymbol{z}$) SDR. For the emission function, we use a threshold of $0.1W$, where W is the active number of bits in the observation state ($\boldsymbol{x}$) SDR. During offline generation we sample an initial $\tilde{\boldsymbol{x}}$ from $\downarrow \hat{\boldsymbol{z}}$ with $W = 1$ active neurons. During generation, we set the maximum number of attractor iterations to 100 but stop iterating when the energy of the state converges to a local minimum. During sequence learning, we update $\boldsymbol{A}$ and $\boldsymbol{B}$ iteratively until the transition is learned before learning the next transition. This iterative weight update makes the model insensitive to the hyperparameter values η. Both $\boldsymbol{A}$ and $\boldsymbol{B}$ are always clamped in the range $[-1, 1]$. The states $\boldsymbol{z}$ are flattened into a single dimension before applying the learning rule in Equation 7. Binary representations (i.e., $\{0, 1\}$) are used as inputs.

Temporal Predictive Coding   For the tPC architecture, we use learning rate of 1e-4 for 800 learning iterations. When a 2-layer tPC model is used, the inference learning rate is set to 1e-2 for 400 inference iterations. Also, the hidden size is set to twice the input size. We found that these parameters work best for all of the experiments and allow the model to fully converge. Bipolar representations (i.e., $\{-1, 1\}$) are used as inputs.

Asymmetric Hopfield Network   The Hopfield model does not require hyperparameters other than the ablated separation function. In many experiments, we use a polynomial separation function with degree d set to 1 or 2. Bipolar representations (i.e., $\{-1, 1\}$) are used as inputs.

E Experiments

In this section, we describe the setup of each figure in the main paper and provide additional quantitative and qualitative results for each task. *All experiments are run for 10 different seeds/trials. We report the mean and standard deviation in all the figures and tables.*

E.1 Sequence Capacity

Figure 3 A   This experiment plots the maximum offline sequence length (i.e., sequence capacity, T_{max}) at different input sizes. The input size N_c is varied from 10 to 100 while the number of active bits W is fixed to 5. We compare variants of our model with N_k set to 4 and 8 to temporal predictive coding (tPC) and Asymmetric Hopfield Network (AHN). We observe that AHN completely fails as the sparsity S of the pattern decreases. Therefore, we also compare to AHN with the sparsity set to 50% (i.e., $W = 0.5N_c$). For AHN models, we experiment with a polynomial exponential function with degrees 1 and 2, as recently proposed [12] and used for evaluation in recent papers [62]. All models in this experiment are set to recall/generate in an offline manner, where only the first input is provided. PAM outperforms all other methods and has the potential to improve further by expanding the context neurons N_k. The patterns in this experiment are uncorrelated, such that each pattern has active bits that are uniformly initialized.

Figure 3 B   This experiment plots the effect of sequence correlation on the maximum offline capacity. The higher the correlation value, the more exact repetitions of patterns are available in the sequence. We enforce correlation by limiting the number of unique patterns (i.e., vocab) used to create the sequence. All patterns in this experiment are set to a size of $N_c = 100$ and $W = 5$ (except for AHN, which is set at $W = 0.5N_c$). Results show that the capacity of all other methods sharply drops when correlation is introduced. PAM retains most of its original capacity.

21

Figure 3 E & F   In this experiment, we provide a qualitative example of a short sequence ($T = 10$) with high correlation (0.8). The sequence is learned by all the methods; then we perform offline (**E**) and online (**F**) recall on the sequence. We use the SDR autoencoder to create SDRs from these CIFAR images for training and recall. The SDRs have a size N_c of 200 and $W = 10$. In the offline recall, only the first input is provided and the model auto-regressively generates the full sequence using its own predictions at every time step. In online recall, the models perform a single-step prediction and always use the ground truth input at every time step to perform predictions. Results show that only PAM can retain a context of correlated sequence and accurately predict into the future based on this context.

Figure 8 A   In this experiment, we show the effect of scaling the model context memory beyond a simple $N_k = 4$ and $N_k = 8$. We show that when using $N_k = 16$ and $N_k = 24$, PAM can model much longer sequences. We vary the input size N_c from 10 to 50 and report the offline sequence capacity of the model as ablations.
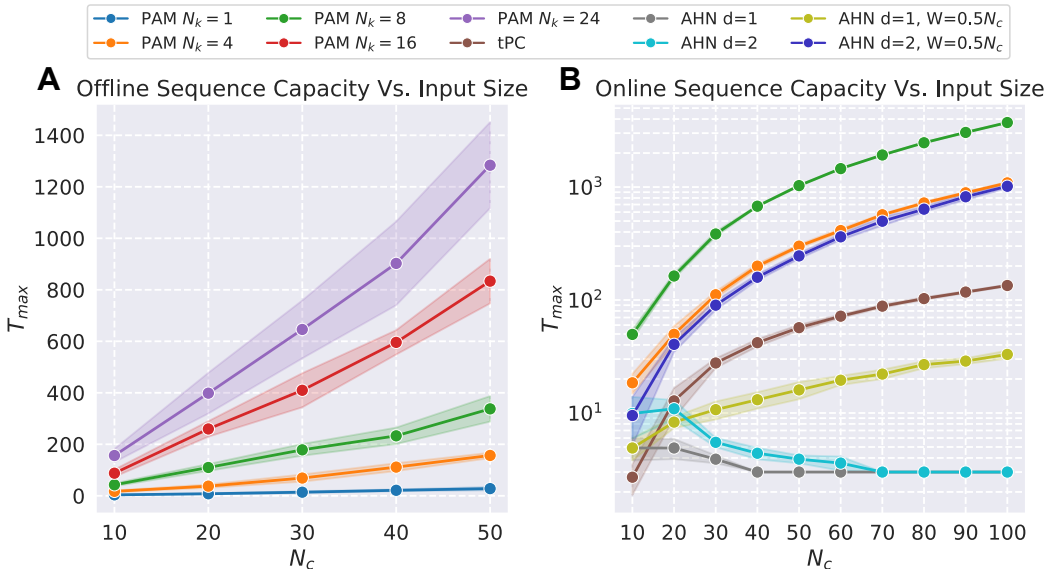
Figure 8: Additional sequence capacity experiments. **A**: scaling of the offline sequence capacity with context memory size N_k and input size N_c. **B**: Online sequence capacity of various methods.

Figure 8 B   Similar to the experiment plotted in Figure 3 **A**, we report the sequence capacity with input size N_c. However, this experiment evaluates the online generation capacity, where the model uses the correct pattern at every prediction time step instead of using its own prediction from the previous time step. Results show that PAM significantly increased in capacity (three times in some cases), whereas the other methods have not increased as much in modeling longer sequences.

Figure 9   We provide additional qualitative example on a different highly correlated sequence. The result shows a different failure mode for AHN, whereas PAM still performs well.

E.2   Catastrophic Forgetting

Figure 4 A   We benchmark the performance of difference models in the challenging continual learning setup. The models are expected to avoid catastrophic forgetting by not overwriting previously learned sequences. In this experiment, we use 50 sequences, each with size $N_c = 100$ and length $T = 10$. We vary the correlation of the sequences from 0.0 to 0.5 and compute the backward transfer metric with the normalized IoU as the measure of similarity. Results show that AHN can avoid catastrophic forgetting when the sequence is uncorrelated but quickly drops in performance with correlation. tPC fails to retain learned sequences regardless of correlation. PAM performs well with more context neurons N_k. When setting N_k to 1, the model fails to retain its knowledge due to the decreased context modeling capability with a single context neuron.

22

Figure 9: Additional qualitative example of correlated sequential memory with CIFAR images.

Figure 4 B In this experiment, we report the performance of the models on protein sequences. This is a more challenging setup due to the long sequence (a few hundred on average) with high correlation (only 20 unique Amino Acids). We show a similar trend where the other methods fail due to high correlation or sequence lengths. PAM outperforms the other methods when using context memory N_k of 16 or 24. All Amino Acid types are converted to fixed and randomly initialized SDRs with $N_c = 100$. The sparsity is set similar to sequence capacity experiments (i.e., $W = 5$ and $W = 0.5N_c$).

Figure 4 F We provide qualitative results on a simple experiment with 2 sequences from moving MNIST. The models learn the first sequence and then learn the second sequence. The models are not allowed to train on the first sequence after they have trained on the second sequence. We then perform online generation on the first sequence with all models. We use the SDR autoencoder to generate SDRs for all images in the sequences; the SDRs have $N_c = 100$ with $W = 5$ (for all methods except AHN). The results show that PAM can recall the full sequence even after being trained on another sequence. Other methods fail in this simple task, even in online recall setup.

Figure 10 We provide continual learning results similar to Figures 4 **A** & **B**; however, instead of offline generation, we perform the evaluation in online manner.

Figure 11 We provide additional qualitative results on Moving MNIST, as well as quantitative results averaged over 10 trials of MNIST sequence pairs. These quantitative results are reported as the Mean Squared Error of the reconstructed image. Results show that PAM reports the lowest error with a much smaller variance.

Tables 2, 3, 4, & 5 The Backward Transfer metric (BWT) to evaluate catastrophic forgetting is computed by taking the average of the performance on previously learned sequences after training on a new sequence. In addition to plotting this average in previous experiments, we provide the full tables for one of the experiments as an example. All tables show results on 10 sequences and $N_c = 100$. The BWT metric is calculated as the average of the similarity metric reported in these tables.

23
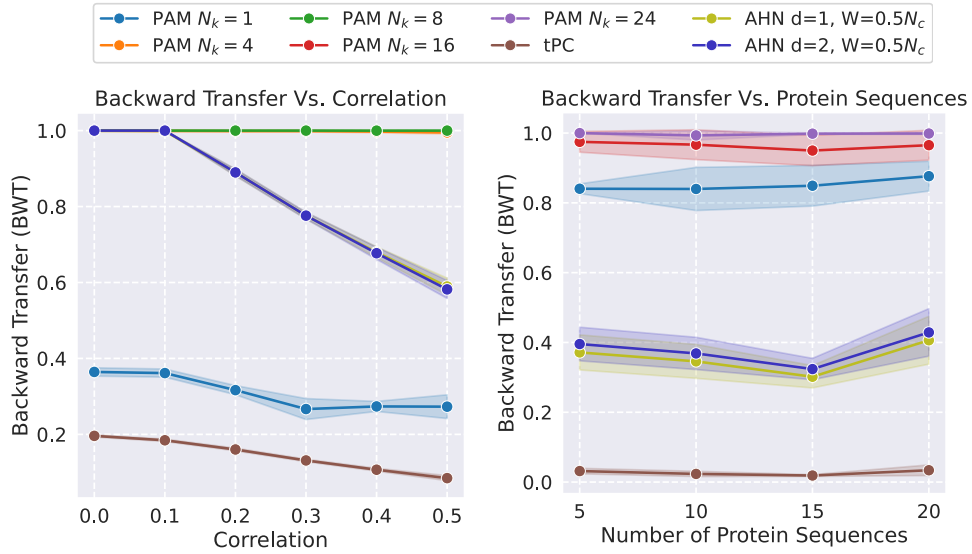
Figure 10: Catastrophic forgetting experiments in an online manner. Results are shown on a synthetic dataset of SDR sequences with different correlations and on protein sequences.
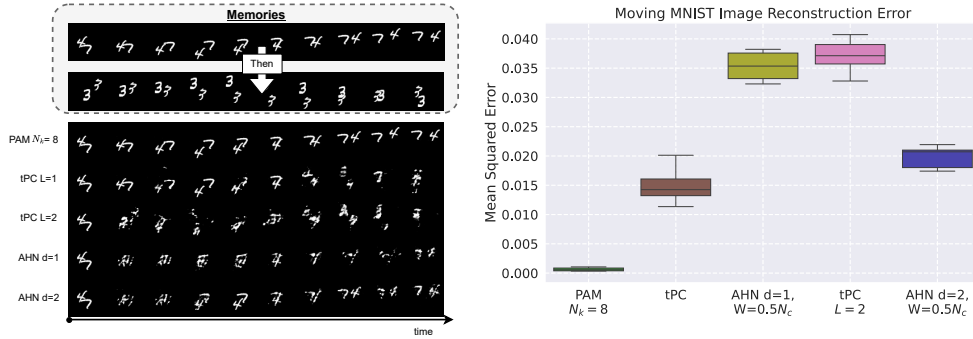
Figure 11: Additional qualitative catastrophic forgetting visualization on Moving MNIST and quantitative results on the reconstruction error of 10 Moving MNIST random examples

Table 2: Catastrophic forgetting experiment results on 10 sequences for PAM with $N_k = 1$. The table shows the mean normalized IoU and standard deviation of previous learned sequences after training on new sequences. The Backward Transfer metric is the average of all the shown numbers. Results are averaged over 10 trials.

	Test	Sequence ID									
Train		1	2	3	4	5	6	7	8	9	10
Sequence ID	1	-	-	-	-	-	-	-	-	-	-
	2	0.692 ± 0.220	-	-	-	-	-	-	-	-	-
	3	0.654 ± 0.252	0.586 ± 0.314	-	-	-	-	-	-	-	-
	4	0.581 ± 0.200	0.619 ± 0.411	0.759 ± 0.229	-	-	-	-	-	-	-
	5	0.634 ± 0.236	0.553 ± 0.348	0.817 ± 0.200	0.823 ± 0.215	-	-	-	-	-	-
	6	0.617 ± 0.224	0.501 ± 0.322	0.695 ± 0.289	0.546 ± 0.329	0.637 ± 0.299	-	-	-	-	-
	7	0.660 ± 0.246	0.545 ± 0.343	0.706 ± 0.265	0.490 ± 0.239	0.604 ± 0.325	0.716 ± 0.293	-	-	-	-
	8	0.631 ± 0.231	0.657 ± 0.317	0.693 ± 0.292	0.527 ± 0.236	0.633 ± 0.300	0.571 ± 0.297	0.748 ± 0.268	-	-	-
	9	0.700 ± 0.237	0.531 ± 0.357	0.723 ± 0.294	0.645 ± 0.286	0.584 ± 0.277	0.605 ± 0.284	0.613 ± 0.265	0.539 ± 0.267	-	-
	10	0.659 ± 0.235	0.488 ± 0.317	0.602 ± 0.326	0.498 ± 0.260	0.532 ± 0.246	0.630 ± 0.257	0.664 ± 0.327	0.660 ± 0.330	0.575 ± 0.313	-

24

Table 3: Catastophic forgetting experiment results on 10 sequences for PAM with $N_k = 4$. The table shows the mean normalized IoU and standard deviation of previous learned sequences after training on new sequences. The Backward Transfer metric is the average of all the shown numbers. Results are averaged over 10 trials.

Train \ Test (Sequence ID)	1	2	3	4	5	6	7	8	9	10
1	-	-	-	-	-	-	-	-	-	-
2	1.000 ± 0.000	-	-	-	-	-	-	-	-	-
3	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-	-	-	-	-
4	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-	-	-	-
5	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-	-	-
6	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-	-
7	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-	-
8	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-	-
9	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-	-
10	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	-

Table 4: Catastophic forgetting experiment results on 10 sequences for tPC. The table shows the mean normalized IoU and standard deviation of previous learned sequences after training on new sequences. The Backward Transfer metric is the average of all the shown numbers. Results are averaged over 10 trials.

Train \ Test (Sequence ID)	1	2	3	4	5	6	7	8	9	10
1	-	-	-	-	-	-	-	-	-	-
2	0.452 ± 0.230	-	-	-	-	-	-	-	-	-
3	0.180 ± 0.135	0.360 ± 0.270	-	-	-	-	-	-	-	-
4	0.148 ± 0.102	0.326 ± 0.257	0.462 ± 0.311	-	-	-	-	-	-	-
5	0.095 ± 0.047	0.148 ± 0.040	0.253 ± 0.166	0.393 ± 0.339	-	-	-	-	-	-
6	0.055 ± 0.038	0.102 ± 0.052	0.213 ± 0.245	0.211 ± 0.199	0.344 ± 0.256	-	-	-	-	-
7	0.068 ± 0.037	0.062 ± 0.050	0.102 ± 0.067	0.103 ± 0.087	0.215 ± 0.197	0.383 ± 0.189	-	-	-	-
8	0.038 ± 0.035	0.041 ± 0.045	0.052 ± 0.034	0.080 ± 0.074	0.073 ± 0.062	0.232 ± 0.138	0.461 ± 0.326	-	-	-
9	0.021 ± 0.022	0.032 ± 0.022	0.056 ± 0.042	0.057 ± 0.027	0.079 ± 0.086	0.170 ± 0.129	0.290 ± 0.159	0.284 ± 0.169	-	-
10	0.016 ± 0.019	0.023 ± 0.028	0.039 ± 0.028	0.032 ± 0.039	0.058 ± 0.058	0.133 ± 0.059	0.187 ± 0.134	0.288 ± 0.302	0.261 ± 0.237	-

Table 5: Catastophic forgetting experiment results on 10 sequences for AHN with $d = 2$ and $W = 0.5N_c$. The table shows the mean normalized IoU and standard deviation of previous learned sequences after training on new sequences. The Backward Transfer metric is the average of all the shown numbers. Results are averaged over 10 trials.

Train \ Test (Sequence ID)	1	2	3	4	5	6	7	8	9	10
1	-	-	-	-	-	-	-	-	-	-
2	0.689 ± 0.192	-	-	-	-	-	-	-	-	-
3	0.689 ± 0.192	0.595 ± 0.334	-	-	-	-	-	-	-	-
4	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	-	-	-	-	-	-	-
5	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	-	-	-	-	-	-
6	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	-	-	-	-	-
7	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	0.667 ± 0.226	-	-	-	-
8	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	0.667 ± 0.226	0.734 ± 0.267	-	-	-
9	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	0.667 ± 0.226	0.734 ± 0.267	0.558 ± 0.257	-	-
10	0.689 ± 0.192	0.595 ± 0.334	0.765 ± 0.238	0.780 ± 0.226	0.692 ± 0.268	0.667 ± 0.226	0.734 ± 0.267	0.558 ± 0.257	0.659 ± 0.293	-

E.3 Multiple Possibilities Generation

In this task, we evaluate the models' ability to generate meaningful sequences and recall the full dataset despite being presented with multiple valid possibilities. Ideally, the models are expected to sample a single possibility if trained on sequences with ambiguous future continuations (equally valid possibilities). This is a challenging task for most biologically plausible (e.g., tPC, AHN, etc.) and implausible (e.g., transformers, RNNs, etc) models. Most approaches assume the existence of a full set of possibilities and transform the task from regression to classification (e.g., LLM). For vision tasks, some methods (VQ-VAE and its variants) cluster the dense representations to create this set of possibilities and perform classification. We *do not* assume the existence of a full set of possibilities but instead perform a true generative evaluation as a regression task.

Figure 4 C   In this experiment, we compute the average normalized IoU of the generated words. The model's ability to generate a full sequence with high IoU means it can produce sharp single predictions despite being trained on multiple equally valid future predictions. As the number of words

25

increases, the performance of other models decreases as they struggle to model ambiguous future predictions; however, PAM outperforms the other approaches by sampling from these possibilities.

Figure 4 D This experiment evaluates the ability of the models to recall the dataset words. We compute the recall as the number of valid unique words generated divided by the total number of words in the dataset. Since PAM is a generative stochastic model, the recall increases with every generation. The other methods are deterministic and, therefore, do not report an increase in dataset recall with more generations. The other methods completely fail to generate any meaningful words. We use an average IoU threshold of 0.9 to classify a generated word as correct, similar to sequence capacity experiments.

Figure 12 We provide qualitative results by showing the unique generated words by different models after 5 dataset generations. PAM $N_k = 4$ generates some of the dataset words but also generates many wrong words. By increasing the context memory neurons to $N_k = 8$, the model generates many more correct words and reduces the false positives. The other methods cannot generate meaningful words.

Full Dataset	part - call - live - side - both - play - from - move - open - city - left - over - name - form - were - port - more - make - came - hard - most - take - line - some - life - self - like - made - same - near - than - many - late - land - does - this - down - will - farm - hand - turn - work - very - then - been - tell - year - good - must - draw - show - when - said - also - grow - last - know - them - only - read - want - with - what - back - just - such - need - ease - keep - well - tree - look - head - kind - give - page - even - food - home - four - they - long - here - time - that - much - come - have - seem - next - mean - your - walk - went - help - word - find - each - real - high
PAM-4	meem - next - bany - evbe - page - keep - word - came - open - read - kind - went - some - late - seem - tell - city - turn - more - will - bada - move - grow - were - here - keee - near - draw - mean - tree - left - real - such - does - meee - hand - four - very - much - come - back - head - looo - lina - show - with - most - land - from - higa - port - ward - food - whee - good - even - ease - your - only - play - both - name - walk - what - meel - evbr - give - down - year - find - life - want - home - farm - just - form - alea - call - meea - must - hard - alei - each - high - meei - been
PAM-8	name - most - when - turn - more - will - from - land - next - move - side - grow - port - walk - evaf - were - what - here - look - give - near - year - food - find - life - mean - draw - want - page - line - home - farm - just - keep - tree - word - left - came - real - have - call - open - form - good - such - time - help - does - kind - went - even - must - ease - hard - four - hand - also - some - very - high - your - play - come - back - late - both - seem - tell - head - well - city - been - show - with
tPC	hbaa - oaaa - naaa - keaa - laaa - gaaa - thaa - eaaa - mbaa - caaa - raaa - saaa - paaa - veaa - faaa - jaaa - yaaa - waaa - aaaa - daaa - baaa
AHN	oaaa - dcaa - naaa - keaa - laaa - maaa - eaaa - raaa - caaa - saaa - yeaa - paaa - veaa - faaa - haaa - taaa - juca - gcaa - waaa - aaaa - baaa

Figure 12: Qualitative results showing the generated words from PAM, tPC [62] and AHN [12] Words highlighted in green are available in the dataset (i.e., True positives). Words highlighted in red are not available in the dataset (i.e., False positives).

E.4 Noise Robustness

Figure 3 C To evaluate for noise robustness, we plot each model's performance (Normalized IoU) with varying levels of noise added in an online generation setting. The noisy inputs are created by changing a percentage of the active neurons to different uniformly chosen neurons in the SDR. The noise is computed as a percentage for a fair comparison across different SDR sparsities (e.g., tPC vs. AHN). We show the results for sequences of lengths $T = 200$ and no correlation. PAM has the ability to compare the noisy input representation to the learned attractors to recover the correct clean input. Therefore, even when the SDR is completely changed, PAM relies on its predictions and completely ignores the noisy input. During generation, PAM always generates (or corrects a noisy input) from within the predicted set of possibilities. The other approaches use the noisy inputs during recall, which affects their performance.

Figure 4 E We provide qualitative results on the CLEVRER dataset. The memories sequence is learned by all the models, then a noisy sequence is used during generation. We only add noise starting from the second pattern in the input sequence. The results show that tPC models are performing relatively well, yet they are still outperformed by PAM $N_k = 8$. We set $N_c = 200$ in the SDR autoencoder to learn the SDRs used in this experiment. We use 40% noise in this experiment.

Figure 13 We perform additional experiments on varying the sequence lengths and the correlation in the sequence, all the other settings remain the same as in the experiment of Figure 3 C. The results

26

show that with shorter sequences (≤ 200), no noise, and no correlation, all the models recall the learned sequence well. When higher correlation is used, 2-layered tPC performs relatively well with short sequences (i.e., $T = 10$) but fails with longer sequences (i.e., $T = 100$). The Hopfield model fails more with correlation than sequence length. The added noise affects all reported methods except for PAM due to its ability to rely on its predictions and attractors to clean the noisy signal.

Figure 14 We provide an additional qualitative example with similar trend to Figure 4 **E**. We also provide quantitative results of CLEVRER averaged over 10 experiments. The mean squared error of the generated sequence for multiple models at different noise levels is reported. It is clear that PAM outperforms all methods, and a 2-layered tPC is the second best.
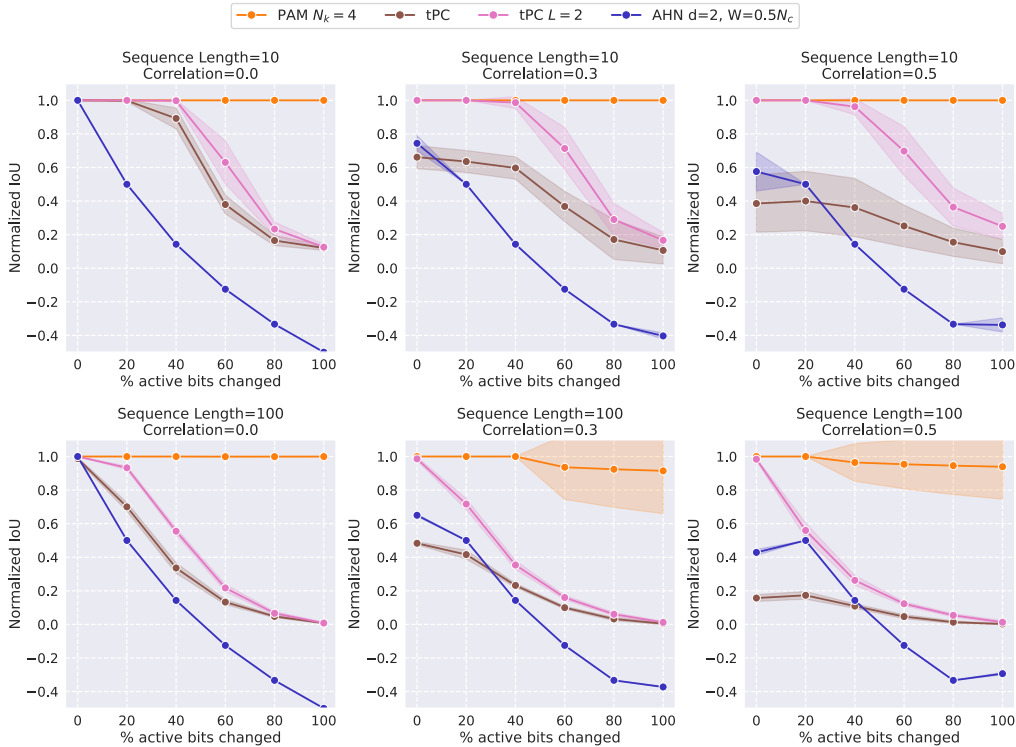
Figure 13: The effect of noise on online generation with varying sequence lengths and sequence correlations.
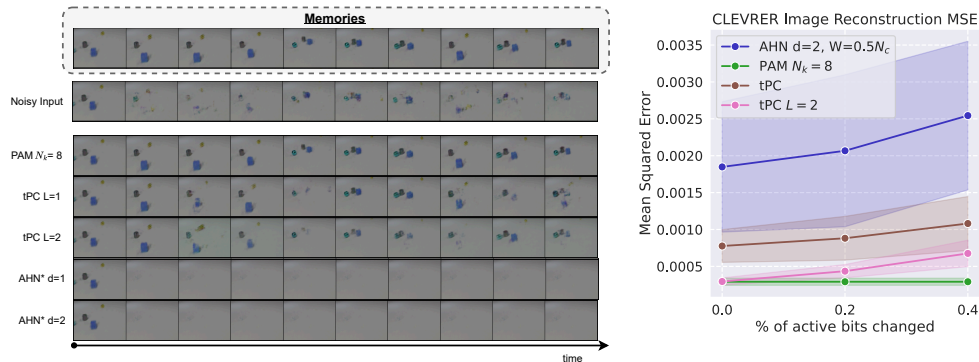
Figure 14: Additional qualitative example of online generation with noise on CLEVRER dataset, and quantitative results of reconstruction error over 10 CLEVRER examples at different noise levels.

27

E.5   Efficiency

Figure 3 D   We compare the efficiency of the models and show that PAM is at least two orders of magnitude more efficient than tPC with 2 layers. A single layer tPC is almost equivalent to PAM with high context memory of $N_k = 24$. AHN is highly efficient as the model is not usually trained, but the recall equation is used instead. Therefore, we exclude AHN from the comparison.

E.6   Additional Results
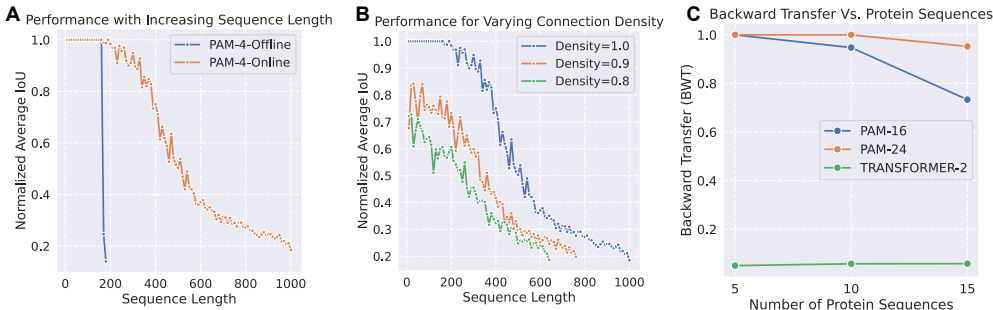
Figure 15: (**A**) Performance of online and offline PAM ($N_c = 50, N_k = 4$) generation with sequence length. (**B**) Performance of PAM ($N_c = 50, N_k = 4$) generation for different connection densities. (**C**) Backward transfer performance for PAM and 2-layer transformer on the protein sequence task.

Figure 15 A   We show the performance of PAM in an online and offline generation task as the sequence length increases. In offline generation, incorrect predictions accumulate quickly, leading to a sharp drop in performance. However, online generation attempts to correct inaccurate predictions; therefore, performance smoothly degrades with sequence length. Note that both experiments (i.e., online and offline) use identical seeds/sequences for a fair comparison.

Figure 15 B   We provide additional results on testing the effect of connection density. While neurons in PAM are currently densely connected (similar to Hopfield Network), we show that PAM can still perform relatively well in a sparser setup.

Figure 15 C   We provide additional results comparing PAM to a 2-layer transformer on continually learning protein sequences. Backward transfer results show low performance of transformers in remembering older sequences after training on new sequences. Table 6 supplements these results by showing detailed sequence-by-sequence performance values. Transformers excel at learning the current task (diagonal values) but suffer in remembering previously learned tasks (lower triangle values).

Table 6: Catastrophic forgetting experiment results on 10 protein sequences for a 2-layer Transformer. The table shows the mean normalized IoU of previous learned sequences after training on new sequences. The Backward Transfer metric is the average of all the numbers in the lower triangle.

Test / Train	1	2	3	4	5	6	7	8	9	10
1	0.749	-	-	-	-	-	-	-	-	-
2	0.000	0.903	-	-	-	-	-	-	-	-
3	0.054	0.051	0.954	-	-	-	-	-	-	-
4	0.066	0.0458	0.002	0.931	-	-	-	-	-	-
5	0.066	0.079	0.0482	0.09	0.863	-	-	-	-	-
6	0.056	0.054	0.035	0.043	0.076	0.945	-	-	-	-
7	0.061	0.042	0.104	0.099	0.041	0.071	0.727	-	-	-
8	0.042	0.068	0.129	0.049	0.078	0.079	0.033	0.719	-	-
9	0.039	0.039	0.078	0.019	0.052	0.026	0.077	0.078	0.723	-
10	0.062	0.077	0.135	0.042	0.029	0.008	0.032	0.065	0.073	0.955

28

F Discussions

F.1 Biological Plausibility

PAM inherits assumptions from Hierarchical Temporal Memory (HTM) [25] that the sequence modeling takes place in Layer IV of the cortical column, where it receives both thalamocortical sensory driving input and contextual information. Sensory input arrives on the proximal dendrites of pyramidal cells, while contextual information reaches the basal dendrites, priming specific contexts by depolarizing neurons. Depolarized neurons then fire first and inhibit other neurons within the same minicolumn via lateral inhibition, a process assumed to be mediated by interneurons (possibly Double Bouquet Cells (DBCs)) [49]. HTM assumes interneurons [40] perform lateral inhibition during prediction but does not learn the synapses of the interneurons. Interneurons are also assumed to perform the inhibitory tasks of the attractor model [14].

Biological plausibility in PAM mainly refers to excluding the obvious biologically implausible learning rules, such as backpropagation or copying of weights. This allows models like PC and HN to be referred to as biologically plausible despite clear contradictions with anatomy [54, 14]. Moreover, PAM also constrains its learning and representations to sparse and binary cell assemblies, which improves the plausibility of the model. Therefore, a biologically plausible model here refers to the ability to continually learn using only local learning rules (e.g., Hebbian, anti-Hebbian, STDP) and sparse binary representations. However, we do not intend to accurately map every detail in PAM to the anatomy of the neocortex.

F.2 Future Directions

In future work, we plan to implement a hierarchy of PAM blocks that supports compositional, part-whole predictions. Unlike simply stacking layers as in Transformer or CNN architectures, our aim is to establish a hierarchy where higher-level blocks make predictions based on broader, contextual patterns. For instance, in vision, higher levels could predict entire objects based on multiple visual cues, while in NLP, they could predict phrases or sentences rather than individual tokens. To achieve this, each PAM level would send a summary representation of the current sequence (its latent state) to the next level for recursive, higher-order processing.

Additionally, we plan to explore top-down feedback mechanisms between PAM blocks - inspired by the hierarchical feedback observed in the neocortex. In the visual system, for example, top-down signals travel from higher regions (e.g., V2) to lower ones (e.g., V1), reaching the apical dendrites of pyramidal neurons. These connections originate in layer VI(a) of the higher region and form en-passant synapses in layer VI(a) of the lower region in the hierarchy (e.g., V1). These connections reach Layer I and provide feedback on the apical dendrites of other layers [61]. In PAM, a similar feedback pathway could enable higher-level blocks to modulate lower-level processing based on partial observations, refining predictions by ruling out certain lower-level possibilities that conflict with higher-level context. For example, the attractor in the higher-level block can reach a decision on the object being seen (e.g., human) based on partial observation (V1: layer III $\rightarrow$ V2: layer IV), and therefore, the feedback inhibitory connections on the apical dendrites can rule out some lower-level possibilities (e.g., paws).

G Sparse Distributed Representations

The neocortex stores and represents information using sparse activity patterns, as demonstrated by empirical evidence [4]. Inspired by HTM [25] and neuroscience-based theories of cortical function, we use Sparse Distributed Representations (SDRs) as the main representation format of PAM. An SDR is a sparse binary representation of a cell assembly where only a small fraction of the neurons in the SDR are active at any time. The location of these active neurons encodes the information that is represented by this SDR. In this section, we describe some useful properties of SDRs and discuss their robustness to noise as opposed to dense representations.

29

G.1 SDR Properties

SDRs are used to represent rich sensory information in the neocortex as a sparse activity pattern. Therefore, from a mathematical viewpoint, an SDR must have the ability to represent many patterns and easily distinguish between them. The capacity of an SDR can be calculated as the possible combinations of locations where neurons can be active. Consider an SDR with size N and number of active neurons W. The total capacity of this SDR is computed as shown in Equation 12.

$$\binom{N}{W} = \frac{N!}{W!(N-W)!} \tag{12}$$

Based on the above Binomial coefficient equation, it may seem that sparsity is not optimal for capacity as the capacity will be the highest when W is exactly half of N. While capacity is important, we aim to represent multiple possibilities as a union of SDRs and, therefore, minimize the overlap between them. From an information-theoretic viewpoint, the goal is to minimize *mutual information* between SDRs to ensure that each SDR carries unique information and the union represents a more comprehensive and diverse set of features. We can minimize the expected IoU by using lower sparsities as shown in Theorem 2. In our experiments, we use $N = 100$ and $W = 5$, which results in a capacity of $\approx 75 \times 10^6$ and an expected IoU of ≈ 0.02. However, when scaled up to more typical values of SDR sizes and sparsities in the neocortex [25, 4] (i.e., $N = 2048$, $W = 40$), we get capacity of $\approx 2.37 \times 10^{84}$ (more than the estimated number of atoms in the observable universe $\approx 10^{80}$) and expected IoU of ≈ 0.01. A sparsity of 0.5 maximizes the mutual information and results in an expected IoU of 0.33, which cannot be used to represent multiple possibilities as a union of SDRs in spite of the optimal capacity. In practice, the size N of the SDR is increased to increase the capacity, and the sparsity W/N is decreased to minimize the expected overlap.

G.2 The robustness of SDRs

Sparse representations naturally minimize the overlap between random SDRs; therefore, they are very tolerant to noise. To visualize this robustness property of SDRs, we design an experiment (Figure 16) where we train an SDR autoencoder with different sparsities and then decode SDRs at various levels of noise added. When the sparsity is increased to 50%, there is a high chance of overlap between SDRs. Therefore, a small amount of noise can cause collisions between SDRs. However, a 5% sparsity can tolerate much more noise without overlapping with other SDRs.

30
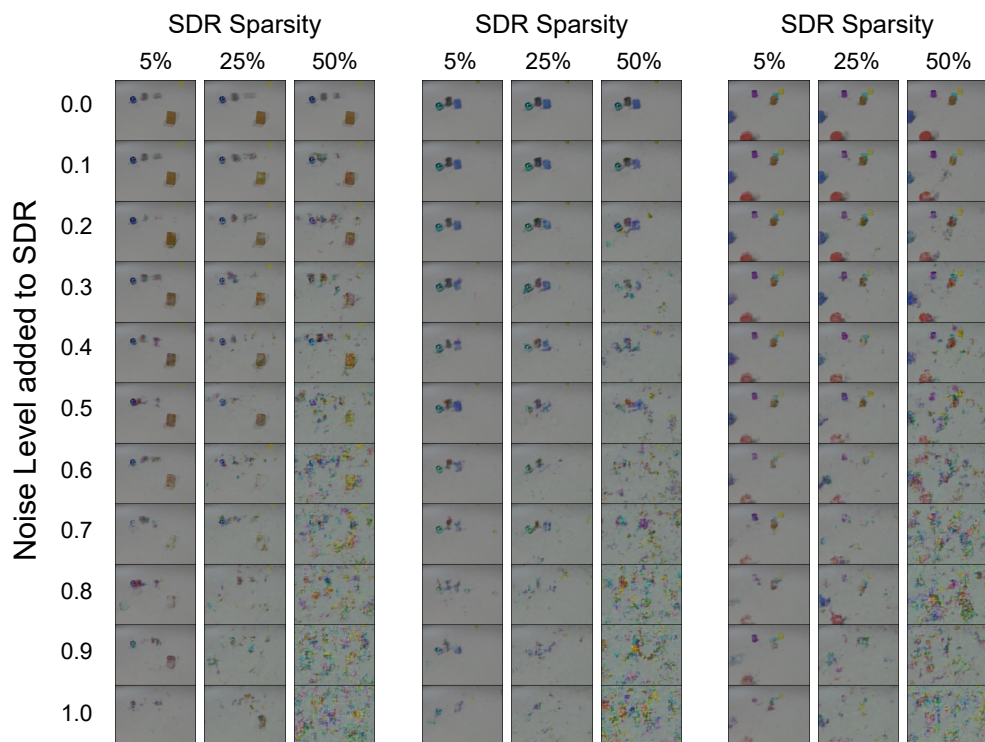
Figure 16: Three examples of decoding an SDR with different noise levels. The results are shown for SDRs with different sparsities trained in an SDR autoencoder on the CLEVRER dataset.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our claims are supported by experimental results and theoretical proofs where applicable. Derivations are provided in the supplemental material.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, the limitations section has been moved from the supplemental material to the main paper for the camera ready version.

   Guidelines:

31

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We provide proofs and derivations in the appendix, where applicable.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide general implementation details for all the methods used. Additionally, we describe the experimental setup for every plot/table in the main paper and in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

32

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our codebase *(now released)* includes a script for every single experiment to accurately reproduce the plots in the paper. We fix the seeds for exact reproducibility.

   Guidelines:
   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

33

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide implementation details for all methods and the datasets' details in the appendix. The code also contains a separate configuration file with all the needed parameters.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All experiments are repeated 10 times at different seeds. The means and standard deviations are provided in all plots and tables.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We specify the compute resources required for PAM (i.e., CPU) and plot a comparison of the time required by each method at different parameters in Figure 3 **D**.

Guidelines:

34

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: The paper conforms with NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: PAM is a powerful sequence modeling algorithm. We do not see direct positive or negative societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

35

Answer: [NA]

Justification: No such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Authors of comparison methods and datasets are properly cited.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No new assets are released.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

36

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

37