BayesPCN: A Continually Learnable Predictive Coding Associative Memory

Anonymous Author(s)
Affiliation
Address
email

Abstract

Associative memory plays an important role in human intelligence and its mechanisms have been linked to attention in machine learning. While the machine learning community's interest in associative memories has recently been rekindled, most work has focused on memory recall ($read$) over memory learning ($write$). In this paper, we present BayesPCN, a hierarchical associative memory capable of performing continual one-shot memory writes without meta-learning. Moreover, BayesPCN is able to gradually forget past observations ($forget$) to free its memory. Experiments show that BayesPCN can recall corrupted i.i.d. high-dimensional data observed hundreds of "timesteps" ago without a significant drop in recall ability compared to the state-of-the-art offline-learned associative memory models.

1 Introduction

One of the hallmarks of biological intelligence is the ability to robustly recall learned associations. Biological agents are able to associate stimuli of various modalities and rely on their memories to perform everyday tasks from navigation to entity recognition. Yet these agents are not only good at recall but can quickly and continually form new memories from as little as a single experience. How do biological agents achieve this feat while mitigating catastrophic forgetting [French, 1999]? And how can we build neural networks that demonstrate these properties? This paper addresses these questions, focusing on fast, continual learning of neural artifacts that perform robust recall of i.i.d. high-dimensional observations.

Associative memory models have been of interest in machine learning for decades [Steinbuch, 1961, Hopfield, 1982]. They have been used to solve a wide range of problems from sequence processing [Schmidhuber, 1992, Graves et al., 2014, Schlag et al., 2021] to pattern detection [Widrich et al., 2020, 2021, Seidl et al., 2021]. Their study has allowed us to build models that replicate aspects of biological intelligence [Kanerva, 1988, Hawkins and Blakeslee, 2004] and provided novel perspectives on machine learning toolsets like dot-product attention, transformers, and linear layers of deep neural networks [Ramsauer et al., 2020, Bricken and Pehlevan, 2021, Irie et al., 2022].

Our main aim is to recall exact memories from a neural network that can rapidly and continually learn from a stream of data. Towards this end we introduce BayesPCN, a Bayesian take on the generative predictive coding network (GPCN) [Salvatori et al., 2021] that combines predictive coding and locally conjugate Bayesian updates. There are currently a number of associative memory systems that are continually learnable [Hopfield, 1982, Kanerva, 1988, Hawkins and George, 2007, Ramsauer et al., 2020, Sharma et al., 2022]. However, to our knowledge, BayesPCN is the first parametric associative memory model to continually learn hundreds of very high-dimensional observations (> 10,000 dimensions) while maintaining good recall performance. In addition, it is also the first hierarchical memory which after repeated forgetting recovers its original memory state.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

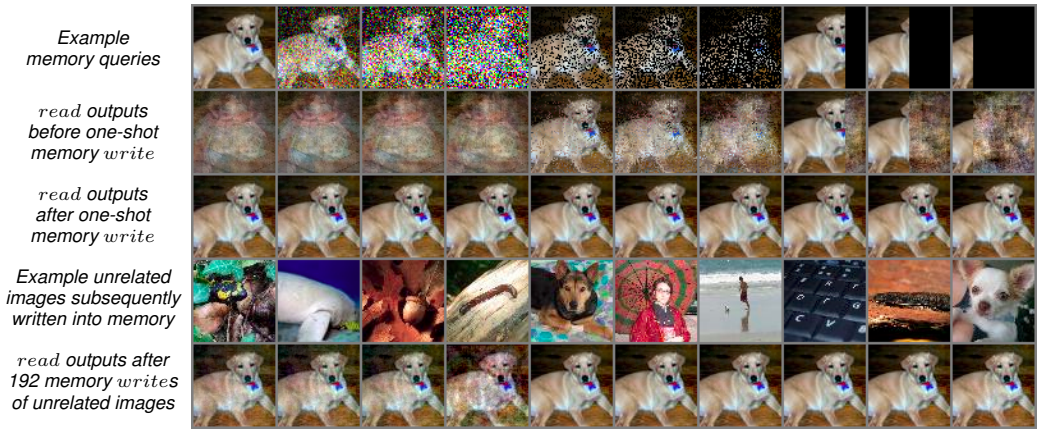

Figure 1: BayesPCN recall before and after the one-shot $write$ of the top left image into memory. The first row contains example memory queries, the second row contains the memory's $read$ outputs before $write$, the third row contains the memory's $read$ outputs immediately after $write$, the fourth row contains sample images written into memory after the top left image, and the last row contains the memory's $read$ outputs after 192 additional datapoints have been sequentially written into memory. The model has 3 hidden layers of width 1024. The four leftmost columns illustrate auto-associative recall while the remaining columns illustrate hetero-associative recall.

2 Background

2.1 Auto-Associative and Hetero-Associative Memory

Before we begin let us review two useful concepts: auto-associative memory and hetero-associative memory. The former, auto-associative memory, refers to a memory system that is content addressable but for which the query may be a corrupted version of the original data item. One can think about auto-associative memory systems as a kind of approximate nearest neighbor lookup mechanism. Hetero-associative memory systems are key-value stores, meaning that the recalled "value" is potentially of a different type to the "key." Crucially, hetero-associative memory systems can, in large part, be implemented by auto-associative memory systems by forming a "joint" object to memorize by combining the key and value into one object. Hetero-associative queries can then be processed simply by looking up the joint object "corrupted" by discarding the value.

2.2 Generative Predictive Coding Network

A generative predictive coding network (GPCN) [Salvatori et al., 2021] is an energy-based associative memory model. As illustrated in Figure 2, GPCN takes the form of a fully connected neural network whose bottom layer activations characterize the sensory inputs and higher layer activations characterize the internal representations of the sensory inputs. GPCN's $read$ operation (recall) initializes the bottom layer activations to the input query vector and iteratively denoises the query until a stored value is retrieved by minimizing its energy function w.r.t. some or all of its neuron activations. GPCN's $write$ operation (learning) adapts the network parameters to shape the energy function such that the stored datapoints become the local minima of the energy function.

Before describing GPCN's $read$ and $write$ mechanisms in more detail, we introduce the notation we use in this paper. Let M be a GPCN with $L+1$ layers of neurons where 0 and L denotes the bottom and the top layer index respectively. Then, let d_l be the width of the l-th layer, $x^l \in \mathbb{R}^{d_l}$ be the row vector denoting the l-th layer's pre-activation function neuron values, and $f^l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$ be the l-th layer's activation function, for instance ReLU. In addition, let $W^l \in \mathbb{R}^{d_{l+1} \times d_l}$ where $0 \leq l < L$ be the synaptic weight matrix between l+1-th and l-th layers and $W^L \in \mathbb{R}^{d_L}$ be the "bias" row vector that influences the dynamics of the top layer neuron activities x^L. One can intuitively think of the model parameters $W^{0:L}$ as the memory content and the activations $x^{0:L}$ as the read state of the memory.

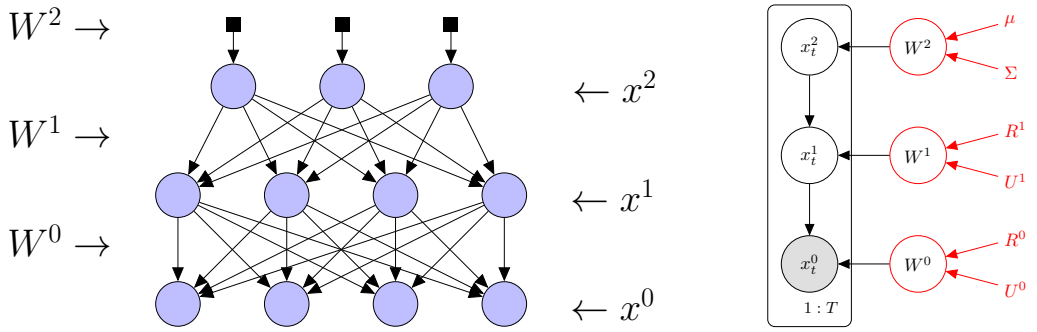

Figure 2: **Left**: A sample 3-layer GPCN/BayesPCN architecture. Circular nodes denote neurons, arrows denote synaptic weights, x^0 denotes the sensory neuron activations, x^1, x^2 denote the upper layer hidden neurons activations, W^0, W^1 denote the synaptic weights between network layers, and W^2 denotes the bias term that influences the top layer neuron activities. **Right**: graphical model for the memory model to the left with T i.i.d. sensory observations $x^0_{1:T}$. GPCN's graphical model is black while BayesPCN's graphical model adds the red components. GPCN treats the synaptic weights as constants while BayesPCN treats them as normally distributed random variables.

GPCN's $read$ and $write$ are facilitated by predictive coding (PC) [Rao and Ballard, 1999], a biologically plausible learning rule that can approximate backpropagation [Millidge et al., 2020]. PC's energy function is the sum of prediction errors across all network layers. For GPCN, the choice of prediction error is the squared Euclidean distance between the actual layer activations and the prediction of the same layer activations made by their immediate upper layer activations. This leads to the following energy function.

$$E(x^{0:L}, W^{0:L}) = \frac{1}{2}||x^L - W^L||^2 + \frac{1}{2}\sum_{l=0}^{L-1} ||x^l - f^{l+1}(x^{l+1})W^l||^2 \tag{1}$$

GPCN's $read$ iteratively performs gradient descent on the energy w.r.t. $x^{0:L}$ after setting the initial values of x^0 to the memory query $\bar{x}^0$ and fixing the values of $W^{0:L}$. GPCN's $write$ obtains the $read$ output $x^{0:L}_{1:T}$ for each of T i.i.d. datapoints to store $x^0_{1:T}$, takes a gradient step on the energy w.r.t. $W^{0:L}$ while fixing $x^{0:L}_{1:T}$, and repeats until convergence.

We now take a probabilistic perspective on GPCN. One can view GPCN as a hierarchical generative model where all neuron activations are modeled by normally distributed random variables and the link function between the subsequent layers is a layer-specific nonlinearity followed by a linear transform. Then, GPCN's $read$ - the minimization of Equation 1 - is equivalent to maximizing the log density in Equation 2 w.r.t. x^0 and $x^{1:L}$.

$$\log p(x^{0:L}|W^{0:L}) = \log \mathcal{N}(x^L; W^L, I) + \sum_{l=0}^{L-1} \log \mathcal{N}(x^l; f^{l+1}(x^{l+1})W^l, I) \tag{2}$$

In other words, GPCN's $read$ performs iterated conditional modes [Besag, 1986] on the hierarchical generative model in the neuron activation space (refer to Algorithms 3 and 4). In addition, GPCN's $write$ can be viewed as an approximate maximization of the lower bound on $\log p(x^0_{1:T}|W^{0:L})$ w.r.t. $W^{0:L}$ [Millidge et al., 2021]. This follows from the fact that GPCN learns via PC.

While GPCN possesses impressive associative recall capabilities, the learning rule proposed in Salvatori et al. [2021] is not amenable to continual learning. This means $write$ operations can only occur offline and incorporating new datapoints requires costly model refitting as well as the need to retain all observed datapoints. BayesPCN's foremost aim is to address this limitation while preserving GPCN's $read$ capabilities as much as possible.

3 Memory Model

BayesPCN is structurally identical to GPCN, but unlike GPCN the network parameters $W^{0:L}$ are treated as latent random variables. This allows BayesPCN's $write$ operation to be cast as

(approximate) posterior inference over $W^{0:L}$ given new observations in a way similar to [Wu et al., 2018a]. In other words, storing T i.i.d. datapoints $x^0_{1:T}$ into memory is equivalent to estimating $p(W^{0:L}|x^0_{1:T})$. Figure 2 illustrates the graphical model for BayesPCN.

We now review the probabilistic structure of BayesPCN. We assume that, prior to any observations, the random variables $x^{0:L}_t$ and $W^{0:L}$ (where t is the datapoint index) are distributed as follows

$$p(W^l) = \begin{cases} \mathcal{MN}(W^l; R^l, \sigma^2_W I, I) & \text{if } 0 \leq l < L \\ \mathcal{N}(W^L; \mu, \sigma^2_W I) & \text{if } l = L \end{cases} \tag{3}$$

$$p(x^l_t|x^{l+1}_t, W^l) = \begin{cases} \mathcal{N}(x^l_t; f^{l+1}(x^{l+1}_t)W^l, \sigma^2_x I) & \text{if } 0 \leq l < L \\ \mathcal{N}(x^L_t; W^L, \sigma^2_x I) & \text{if } l = L \end{cases} \tag{4}$$

where $\mathcal{N}(\mu, \Sigma)$ and $\mathcal{MN}(R, U, V)$ denotes normal and matrix normal distribution [Gupta and Nagar, 2018] respectively. We remind the readers that the matrix normal distribution is a special kind of normal distribution. The joint distribution over all random variables when there are T sensory observations is, due to the conditional independence of $x^{0:L}_t, t \in \{1, ..., T\}$ given $W^{0:L}$,

$$p(x^{0:L}_{1:T}, W^{0:L}) = \left[\prod_{t=1}^{T} p(x^{0:L}_t|W^{0:L})\right] p(W^{0:L}) = \left[\prod_{t=1}^{T}\prod_{l=0}^{L} p(x^l|x^{l+1}, W^l)\right]\prod_{l=0}^{L} p(W^l) \tag{5}$$

We note that BayesPCN's structure leads to two desirable properties that become useful when approximating $p(W^{0:L}|x^0_{1:T})$ during the $write$ operation. First, by exploiting the linear Gaussian marginalization formula and conditional independence we can analytically compute the marginal distribution over all activation vectors $x^{0:L}_t$.

$$p(x^{0:L}_t) = \int \prod_{l=0}^{L} p(x^l_t|x^{l+1}_t, W^l)p(W^l)dW^{0:L} = \prod_{l=0}^{L} p(x^l_t|x^{l+1}_t) \tag{6}$$

$$p(x^l_t|x^{l+1}_t) = \begin{cases} \mathcal{N}(x^l_t; f^{l+1}(x^{l+1}_t)R^l, (\sigma^2_x + f^{l+1}(x^{l+1}_t)U^l f^{l+1}(x^{l+1}_t)^\top)I) & \text{if } 0 \leq l < L \\ \mathcal{N}(x^L_t; \mu, (\sigma^2_x + \sigma^2_W)I) & \text{if } l = L \end{cases} \tag{7}$$

We stress that neither $p(x^0_t)$ nor $p(x^0_t|W^{0:L})$ has an analytic form due to the presence of nonlinearities. In addition, if all activation vectors $x^{0:L}_t$ are observed, the posterior over the model parameters becomes normally distributed due to the linear Gaussian posterior formula and the fact that $W^{0:L}$ are conditionally independent given $x^{0:L}_t$ (the analytical posterior formulae for Equation 9 are in Appendix C).

$$p(W^{0:L}|x^{0:L}_t) = \frac{1}{Z}\prod_{l=0}^{L} p(x^l_t|x^{l+1}_t, W^l)p(W^l) = \prod_{l=0}^{L} p(W^l|x^l_t, x^{l+1}_t) \tag{8}$$

$$p(W^l|x^l_t, x^{l+1}_t) = \begin{cases} \mathcal{MN}(W^l; R^{l*}, U^{l*}, I) & \text{if } 0 \leq l < L \\ \mathcal{N}(W^L; \mu^*, \Sigma^*) & \text{if } l = L \end{cases} \tag{9}$$

$R^{l*}, U^{l*}, \mu^*, \Sigma^*$ denote the sufficient statistics of the layer-wise posterior distributions given x^l_t, x^{l+1}_t.

4 Memory Operations

This section contains the memory operations that BayesPCN supports: $read$, $write$, and $forget$. We use the shorthand $\mathbf{h} = x^{1:L}$ to denote all hidden layer neuron activations and $\mathbf{W} = W^{0:L}$ to denote all network parameters.

4.1 Read

The $read$ operation takes the corrupted query vector $\bar{x}^0$ and retrieves the memory's best guess of the associated original datapoint x^{0*}. As in GPCN, this is done by climbing the memory's log density $\log p(x^0, \mathbf{h}|x^0_{1:T})$ w.r.t. the neuron activations $x^0, \mathbf{h}$. To clarify, $x^0, \mathbf{h}$ is the current memory read state and is not directly related to the past observations $x^0_{1:T}$. BayesPCN's auto-associative recall performs iterated conditional modes on $\log p(x^0, \mathbf{h}|x^0_{1:T})$ w.r.t. $x^0, \mathbf{h}$ and BayesPCN's hetero-associative recall performs gradient ascent on $\log p(x^0, \mathbf{h}|x^0_{1:T})$ w.r.t. $x^0, \mathbf{h}$ while keeping the known values of x^0 fixed. Algorithm 3 and Algorithm 4 in Appendix A, the recall procedures taken from Salvatori et al. [2021] and rewritten from the probabilistic perspective, describe this in more detail.

4.2 Write

The $write$ operations aims to take the previous timestep posterior over the model parameters $p(\mathbf{W}|x^0_{1:t-1})$ and the current timestep observation x^0_t to return the current timestep posterior over the model parameters $p(\mathbf{W}|x^0_{1:t})$. Unfortunately, the analytical form of $p(\mathbf{W}|x^0_{1:t})$ is unavailable due to the presence of nonlinearities in BayesPCN so one must resort to approximate posterior inference.

We propose a sequential importance sampling posterior inference algorithm for estimating $p(\mathbf{W}|x^0_{1:t})$ where the particles are high-dimensional Gaussian distributions. Let $N \in \mathbb{N}$ be the number of sequential importance sampling particles and $n \in \{1, ..., N\}$ be the particle index. The t-th timestep posterior estimate $\hat{p}(\mathbf{W}|x^0_{1:t})$ is recursively represented as a weighted Gaussian mixture of the form $\hat{p}(\mathbf{W}|x^0_{1:t}) = \sum_{n=1}^{N} \omega^{(n)} p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n)}_{1:t})$ where $\omega^{(n)}$ is the n-th particle's importance weight and $p^{(n)}(\mathbf{W}|x^0_{1:t-1}, \mathbf{h}^{(n)}_{1:t-1})$ is the n-th particle's activation conditioned posterior described in Equation 8. We note that each $p^{(n)}(\mathbf{W}|x^0_{1:t-1}, \mathbf{h}^{(n)}_{1:t-1})$ is efficiently characterised by the sufficient statistics $\mu^{(n)}, \Sigma^{(n)}, R^{0:L-1,(n)}, U^{0:L-1,(n)}$. Algorithm 1 describes the $write$ operation in detail (see Appendix B for its justification).

Algorithm 1 Memory Write

Input: Previous network weight distribution $\hat{p}(\mathbf{W}|x^0_{1:t-1})$, current timestep observation x^0_t
Output: Updated network weight distribution $\hat{p}(\mathbf{W}|x^0_{1:t})$

for each model $p^{(n)}(\mathbf{W}|x^0_{1:t-1}, \mathbf{h}^{(n)}_{1:t-1})$ and weight $\omega^{(n)}_{t-1}$ in $\hat{p}(\mathbf{W}|x^0_{1:t-1})$ **do**
  Fit the parameters of the current timestep variational distribution:
    $\phi = \text{argmin}_\phi \, D_{KL}\left(q^{(n)}_\phi(\mathbf{h}_t) \,||\, \hat{p}^{(n)}(x^0_t, \mathbf{h}_t|x^0_{1:t-1})\right)$
  Sample $\mathbf{h}^{(n)}_t$: $\mathbf{h}^{(n)}_t \sim q^{(n)}_\phi(\mathbf{h}_t)$
  Update the model via linear Gaussian update conditioned on x^0_t and $\mathbf{h}^{(n)}_t$:
    $p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n)}_{1:t}) \propto p^{(n)}(\mathbf{W}|x^0_{1:t-1}, \mathbf{h}^{(n)}_{1:t-1}) p^{(n)}(x^0_t, \mathbf{h}^{(n)}_t|\mathbf{W})$
  Compute the unnormalized particle weight: $\hat{\omega}^{(n)}_t = \omega^{(n)}_{t-1} \frac{p^{(n)}(x^0_t, \mathbf{h}^{(n)}_t|x^0_{1:t-1})}{q^{(n)}_\phi(\mathbf{h}^{(n)}_t)}$
end for
Set $\hat{p}(\mathbf{W}|x^0_{1:t}) = \sum_{n=1}^{N} \frac{\hat{\omega}^{(n)}_t}{\sum_{n'=1}^{N} \hat{\omega}^{(n')}_t} p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n)}_{1:t})$

While $q^{(n)}(\mathbf{h}_t)$ in principle can be any distribution, BayesPCN's $write$ sets it to $q^{(n)}(\mathbf{h}_t) = \delta\left(\text{argmax}_{\mathbf{h}_t} \log p(x^0_t, \mathbf{h}_t|x^{0:L}_t)\right)$ as is standard in predictive coding [Millidge et al., 2021]. In addition, we define the BayesPCN parameter prior (of an "empty" memory) as $\hat{p}(\mathbf{W}) = \sum_{n=1}^{N} \frac{1}{N} p^{(n)}(\mathbf{W})$ where $\mu^{(n)}, R^{0:L-1,(n)}$ are initialized using Kaiming initialization. We chose this prior over a non-mixture prior to obtain a multi-particle posterior approximation despite using a Dirac variational distribution.

We now direct our attention to obtaining $\log p^{(n)}(x^0_t, \mathbf{h}|x^0_{1:t-1})$ used in the $read$ operations from $\hat{p}(\mathbf{W}|x^0_{1:t})$. Using Equation 6, we obtain the approximate posterior predictive density

$$\log p(x^0_t, \mathbf{h}|x^0_{1:t-1}) = \log \sum_{n=1}^{N} \omega^{(n)} \int p(x^0_t, \mathbf{h}_t|\mathbf{W}) \hat{p}^{(n)}(\mathbf{W}|x^0_{1:t-1}) d\mathbf{W} \tag{10}$$

$$= \log \sum_{n=1}^{N} \omega^{(n)} p^{(n)}(x^0_t, \mathbf{h}_t|x^0_{1:t-1}) \tag{11}$$

BayesPCN's $write$ can be seen as a special case of sequential imputation [Kong et al., 1994], an online parameter inference algorithm that represents the parameter posterior conditioned on partial observations as a mixture of parameter posteriors conditioned on complete observations. In addition, whereas the parameter update proposed by Salvatori et al. [2021] is doing local gradient descent for each W^l, BayesPCN's parameter update is doing local conjugate normal-normal Bayesian update for each W^l. Hence, like GPCN, BayesPCN's synaptic weight update only utilizes local information.

Lastly, while exact Bayesian updates are invariant to the order of observations, because we are using predictive coding to fit the variational distribution BayesPCN's $write$ is technically not invariant to the order of observations.

4.3 Forget

We also propose a diffusion-based forgetting mechanism for BayesPCN. As more datapoints are written into a BayesPCN model, the model's belief over the true parameter location $R^{1:L-1}, \mu$ shifts and the uncertainty over its belief $U^{1:L-1}, \Sigma$ decreases. This has the effect of making the model's parameters less adaptive to new datapoints and magnifying $R^{1:L-1}, \mu$'s norms as the prior's contribution weakens. One can interpret this as the memory being overloaded. The $forget$ operation counteracts this phenomenon by nudging the model's belief over $R^{1:L-1}, \mu$ toward the prior (empty memory state) and increasing its uncertainty $U^{1:L-1}, \Sigma$ proportional to the forget strength parameter β. Algorithm 2 describes this in detail (the analytical diffusion formulae are in Appendix D).

Algorithm 2 Memory Forget

Input: Previous network weight distribution $\hat{p}(\mathbf{W}|x^0_{1:t})$, forget strength $\beta \in [0, 1]$
Output: Updated network weight distribution $\hat{p}(\mathbf{W}'|x^0_{1:t})$

for each model $p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n)}_{1:t})$ in $\hat{p}(\mathbf{W}|x^0_{1:t})$ **do**
  for each layer weight distribution $p^{(n)}(W^l|x^{(n),l}_{1:t}, x^{(n),l+1}_{1:t})$ in $p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n)}_{1:t})$ **do**
    Obtain the one-step diffused layer weight distribution $p^{(n)}(W^{l'}|x^{(n),l}_{1:t}, x^{(n),l+1}_{1:t})$:
      $p^{(n)}(W^{l'}|x^{(n),l}_{1:t}, x^{(n),l+1}_{1:t}) = \int p(W^{l'}|W^l, \beta) p^{(n)}(W^l|x^{(n),l}_{1:t}, x^{(n),l+1}_{1:t}) dW^l$
  end for
  Set $p^{(n)}(\mathbf{W}'|x^0_{1:t}, \mathbf{h}^{(n)}_{1:t}) = \prod_{l=0}^{L} p^{(n)}(W^{l'}|x^{(n),l}_{1:t}, x^{(n),l+1}_{1:t})$
end for
Set $\hat{p}(\mathbf{W}'|x^0_{1:t}) = \sum_{n=1}^{N} \omega^{(n)} p^{(n)}(\mathbf{W}'|x^0_{1:t}, \mathbf{h}^{(n)}_{1:t})$

If applied an infinite number of times, $forget$ reverts the posterior over the parameters $\hat{p}(\mathbf{W}|x^0_{1:t})$ to the prior $\hat{p}(\mathbf{W})$ due to the property of diffusion if the particle weights are uniform. In practice, one would apply $forget$ periodically or when the memory starts to become overloaded from too much data and stops being performant. Because $forget$ erases stored information for all of datapoints somewhat evenly, the datapoints from the distant past that $forget$ was applied to more would be, as the name suggests, more forgotten than the recent datapoints that $forget$ was applied to less.

5 Related Work

We briefly review the most relevant work to BayesPCN, starting with the predictive coding associative memories. As covered in Section 2, GPCN [Salvatori et al., 2021] is the model that BayesPCN is built upon. Self-taught associative memory [Dovrolis, 2018, Smith et al., 2021] is a hierarchical predictive coding memory that performs unsupervised continual learning using online clustering modules as building blocks, unlike BayesPCN's building blocks of artificial neurons. The sequential neural coding network [Ororbia et al., 2019, 2018], while not explicitly used as an associative memory, is also a continually learned predictive coding network similar in structure to GPCN that combats catastrophic forgetting via task-dependent activation sparsity. BayesPCN on the other hand combats catastrophic forgetting by incorporating uncertainty over the synaptic weights.

Moving on to general attractor-based memory models, Hopfield networks [Hopfield, 1982] and their modern variants [Krotov and Hopfield, 2016, Ramsauer et al., 2020, Krotov and Hopfield, 2020, Krotov, 2021, Millidge et al., 2022] are biologically motivated auto-associative memories that define the neuron activation dynamics via Lagrangian functions. Out of these models only Krotov [2021]'s model is hierarchical, but theirs is not continually learned. BayesPCN generalizes the modern Hopfield network and fits into the universal Hopfield network's similarity-separation-projection paradigm for associative memories (refer to Appendix E). Sparse Distributed Memory (SDM) [Kanerva, 1988] is a non-hierarchical memory that exploits high-dimensionality to perform auto-associative and hetero-associative recall. Hierarchical Temporal Memory (HTM) [Hawkins and George, 2007, Cui et al., 2016] is a biologically constrained hierarchical memory that continually

learns time-based patterns with sparse representations. HTM does not employ predictive coding and to our knowledge has not yet been tested on high-dimensional data.

Relative to generative modeling and memory research, our work has connections to the Kanerva machine's memory $write$ as inference approach [Wu et al., 2018a,b] and the one-shot memory write from Bartunov et al. [2019], Pham et al. [2022]. These papers' models employ meta-learning unlike aforementioned work. We note that the Kanerva machine also applies conjugate normal-normal Bayesian update to adapt its top level memory matrix once the model is meta-learned via end-to-end gradient descent. On the other hand, BayesPCN applies conjugate normal-normal Bayesian update to all of its weight matrices and this as of itself constitutes learning.

6 Experiments

This section assesses BayesPCN's recall capabilities on image recovery tasks and examines the effect of $forget$ on BayesPCN's behaviour.

We used CIFAR10 [Krizhevsky et al., 2009] and Tiny ImageNet [Le and Yang, 2015] datasets for all experiments, whose images are of size $3 \times 32 \times 32$ and $3 \times 64 \times 64$ respectively. During the $write$ phase of the experiments, BayesPCN models were given one image to store into their memories per timestep for up to 1024 images. Then, during the $read$ phase of the experiments, we recorded the pixel-wise mean squared error (MSE) between the $write$ phase images and the models' $read$ outputs given the corrupted $write$ phase images as inputs. For reference, the pixel-wise MSE of 0.01 is distinguishable to human eyes on close inspection of the images.

BayesPCN models were compared against identity models (models that return their inputs as outputs), modern Hopfield networks (MHN) [Ramsauer et al., 2020], and vanilla GPCNs [Salvatori et al., 2021] all of which were offline learned. BayesPCN models were also compared against online GPCNs, GPCNs that naively applied their offline memory $write$ to sequentially arriving observations and memorized one image per timestep for up to 1024 images. We used our own autodiff implementation of MHNs based on Appendix E. See Appendix F for additional experiment details.

6.1 Auto-Associative and Hetero-Associative Read

For the auto-associative recall experiments, the memory inputs at the $read$ phase were the $write$ phase images corrupted by pixel-wise white noise with standard deviation 0.2. For the hetero-associative recall experiments, we tested two types of training image corruption schemes. The first type, dropout, randomly blacked out 25% of the image pixels. The second type, mask, blacked out 25% of the rightmost image pixels. Image recovery under these settings is hetero-associative recall because one can view the non-blacked out pixels as the "key" and the blacked out pixels as the "value". We experimented with 4-layer BayesPCN models that have hidden layer sizes of 256, 512, and 1024, particle counts of 1 and 4, and activation functions ReLU and GELU. We searched over the same hyperparameters for offline and online 4-layer GPCNs aside from the particle count. Lastly, we evaluated 4-layer BayesPCN models, which we denote as BayesPCN ($forget$), that applied $forget$ with strength $\beta = 0.001$ to their weights once every 64 $write$ operations.

Table 1 shows the average MSE between the ground truth images and the associative memory $read$ outputs for models trained on data sequences of different lengths, where the reported scores come from the best models of each category. The results were averaged across three random seeds. Appendix H contains additional images that demonstrate how BayesPCN's $read$ performance scales with different hyperparameters, how BayesPCN gets better at reconstructing unseen images the more datapoints it observes, and shows images sampled from BayesPCN by using it as a generative model.

We found that for sequences of length ≤ 512, BayesPCN's recall errors were consistently lower than online GPCN's recall errors and were similar to offline GPCN's recall errors. In particular, BayesPCN's hetero-associative recall errors were orders of magnitude lower than online GPCN's errors. As expected, BayesPCN ($forget$) performed worse than BayesPCN in this regime because information is being forgotten before the memory is nearing its storage capacity. However, when the sequence length approached 1024, BayesPCN's recall errors deteriorated more rapidly than online GPCN's errors and on the TinyImageNet white noise task became even worse than the identity model. However, BayesPCN ($forget$)'s recall errors deteriorated more gradually and overall was able to outperform online GPCN's recall errors on all tasks and sequence lengths. We hypothesize that

White Noise CIFAR10 MSE				
Sequence Length	128	256	512	1024
Identity	0.1596	0.1600	0.1600	0.1600
MHN	0.0000	0.0000	0.0000	0.0000
GPCN (Offline)	0.0029	0.0047	0.0074	0.0121
GPCN (Online)	0.0102	0.0150	0.0191	0.0210
BayesPCN	0.0017	0.0079	0.0146	0.0339
BayesPCN ($forget$)	0.0064	0.0101	0.0145	0.0180

White Noise Tiny ImageNet MSE				
Sequence Length	128	256	512	1024
Identity	0.1600	0.1602	0.1601	0.1600
MHN	0.0000	0.0000	0.0000	0.0000
GPCN (Offline)	0.0005	0.0010	0.0019	0.0058
GPCN (Online)	0.0082	0.0109	0.0137	0.0180
BayesPCN	0.0011	0.0033	0.0064	0.6223
BayesPCN ($forget$)	0.0027	0.0059	0.0108	0.0176

Dropout CIFAR10 MSE				
Sequence Length	128	256	512	1024
Identity	1.1140	1.1178	1.1353	1.1481
MHN	0.0000	0.0000	0.0000	0.0000
GPCN (Offline)	0.0000	0.0000	0.0000	0.0003
GPCN (Online)	0.0023	0.0034	0.0055	0.0075
BayesPCN	0.0000	0.0000	0.0000	0.0049
BayesPCN ($forget$)	0.0000	0.0001	0.0004	0.0019

Dropout Tiny ImageNet MSE				
Sequence Length	128	256	512	1024
Identity	1.0629	1.0889	1.1154	1.1072
MHN	0.0000	0.0000	0.0000	0.0000
GPCN (Offline)	0.0000	0.0000	0.0000	0.0001
GPCN (Online)	0.0044	0.0061	0.0077	0.0106
BayesPCN	0.0000	0.0000	0.0000	0.0175
BayesPCN ($forget$)	0.0000	0.0000	0.0002	0.0009

Mask CIFAR10 MSE				
Sequence Length	128	256	512	1024
Identity	1.1272	1.1373	1.1619	1.1653
MHN	0.0000	0.0000	0.0011	0.0000
GPCN (Offline)	0.0002	0.0006	0.0011	0.0028
GPCN (Online)	0.0136	0.0281	0.0547	0.0805
BayesPCN	0.0000	0.0000	0.0007	0.1375
BayesPCN ($forget$)	0.0000	0.0007	0.0101	0.0544

Mask Tiny ImageNet MSE				
Sequence Length	128	256	512	1024
Identity	1.0876	1.0884	1.0982	1.1132
MHN	0.0000	0.0000	0.0000	0.0010
GPCN (Offline)	0.0001	0.0004	0.0005	0.0018
GPCN (Online)	0.0105	0.0357	0.0491	0.0735
BayesPCN	0.0000	0.0000	0.0003	0.0673
BayesPCN ($forget$)	0.0000	0.0003	0.0031	0.0284

Table 1: Average MSE between the training images and the associative memory $read$ outputs for the white noise, dropout, and mask tasks on CIFAR10 and Tiny ImageNet. The reported values are from the best models of each category. Table 2 shows the same result with standard deviations.

BayesPCN's sudden performance drop stems from the fact that it is not "removing" old information unlike online GPCN and BayesPCN ($forget$) due to conjugate Bayesian update's observation order invariance property. This causes BayesPCN to approach the overloaded memory regime, where recall of all data suffers, more abruptly once some amount of information has been stored.

GPCN and BayesPCN models overall found hetero-associative recall easier than auto-associative recall, likely because the former fixes the key bits during memory $read$ and thus discourages the algorithm from wandering to less realistic x^0 values. Counterintuitively, GPCN and BayesPCN models were better at Tiny ImageNet recall over CIFAR10 recall. We attribute this behaviour to the fact that for some models, high-dimensional data recall can ironically be easier than low-dimensional data recall because individual datapoints are further apart from one another in higher dimensions.

A surprising discovery from this experiment was how good MHN's recall can be compared to GPCN and BayesPCN's recall when the memory query has a moderate amount of noise. Because MHN explicitly retains past observations in a data matrix and its softmax inverse temperature was set very high ($\beta = 10,000$), MHN's recall effectively selected the observation that had the highest dot-product similarity to the memory query in our experiments. In Appendix G, we assess how MHN's recall compares to BayesPCN's recall when the memory query corruption is much higher. As expected, because the observation with the highest dot-product similarity to the memory query is often not the right image to recall, BayesPCN significantly outperformed MHN in hetero-associative recall tasks.

6.2 Forgetting

This section investigates the $forget$ operation's effect on BayesPCN's recall behaviour. Specifically, we show that $forget$ causes BayesPCN to better recall recent observations at the expense of older observations and that this effect is more pronounced the higher the forget strength is. We trained six structurally identical BayesPCN models on 1024 CIFAR10 images sequentially and applied $forget$ with strength $\beta = 0, 0.001, 0.002, 0.005, 0.01, 0.02$ to each model once after every 64 $write$ operations.

Figure 3 compares the models' recall MSE on 16 data batches of size 64 that were observed during different points of the training. We see that BayesPCN with no $forget$ ($\beta = 0$) performs similarly on all datapoints regardless of their observation timestep, which is likely the consequence of conjugate

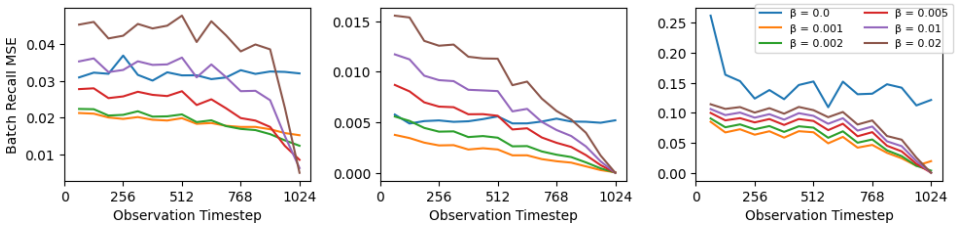

Figure 3: From left to right, a sample BayesPCN model's recall MSE for white noise ($\sigma = 0.2$), pixel dropout (25%), and pixel masking (25%) tasks with different forget strength β after sequentially observing 1024 CIFAR10 images.

Bayesian update's observation order invariance. On the other hand, BayesPCNs with $forget$ ($\beta > 0$) consistently perform better on newer datapoints with this effect being strongly correlated with the value of β. As previously shown, some amount of forgetting during training can improve the *overall* performance of the memory across all datapoints over no forgetting for long training sequences.

7 Discussion

One can intuitively think of BayesPCN $read$ as an approximate nearest neighbour operation with a learned similarity metric and $write$ as the continual learning of that metric. MHN on the other hand uses the dot-product distance as its similarity metric [Millidge et al., 2022]. This explains why MHN is not good at highly masked image recall because from a pixelwise dot-product distance perspective, highly masked image queries are often closer to unrelated images than to the ground truth images.

BayesPCN is closely related to the predictive coding theories of the brain's memory system. For example, Barron et al. [2020] suggests that the hippocampus may correspond to the top layer of a predictive coding network and the neo-cortical areas may correspond to the rest of the predictive coding network. Similarly, Dovrolis [2018] posits that their predictive coding network could serve as a model of the cortical column albeit without experimental results. It is however unclear how BayesPCN's locally conjugate Bayesian weight update can be realized in the brain unlike GPCN's Hebbian weight update. Nevertheless, given that conjugate normal-normal Bayesian update's posterior mean formula (Equation 27) can be seen as a preconditioned Hebbian weight update influenced by the input and error neurons, there may be a biologically plausible way of approximating single particle BayesPCN $write$. In addition, there are similarities between BayesPCN's diffusion-based $forget$ and the synaptic homeostasis hypothesis [Cirelli and Tononi, 2022], where during sleep synaptic weight strengths are decreased on average to facilitate future learning.

Aside from these biological connections, BayesPCN has many potential machine learning applications. One possible line of work is using it in conjunction with a neural controller that learns what information to store into memory [Graves et al., 2014, Schlag et al., 2021]. Using BayesPCN in this way would confer the benefit of high query noise tolerance for neural controllers. Another line of work is using it to solve small data supervised learning problems, leveraging the fact that nearest neighbour methods work well when data availability is sparse [Ramsauer et al., 2020].

To conclude, we introduced BayesPCN, an associative memory model that can robustly retrieve past observations given various corrupted inputs, perform one-shot parameter update of a deep neural network to store new observations as attractors, and forget old observations to better store newer ones. We demonstrated that BayesPCN's recall performance is comparable to its offline-learned counterpart even though it is continually learned. There are several intriguing extensions to BayesPCN that may address some of its current limitations. As of now, BayesPCN $read$ can take up to thousands of activation gradient descent iterations to obtain the best possible recall result when a lot of data has been stored into memory. Supplementing BayesPCN $read$ with amortized inference may greatly reduce this computational requirement [Tschantz et al., 2022]. In addition, BayesPCN $read$ can return a nonsensical output if the memory query is too far away from the stored datapoints or if the memory is overloaded. Investigating BayesPCN's latent landscape may allow us design models with more robust basin of attractions around the stored datapoints. Last but not least, extending BayesPCN beyond memorizing i.i.d. data to memorizing sequences like the human episodic memory would be of great interest.

References

Helen C. Barron, Ryszard Auksztulewicz, and Karl Friston. Prediction and memory: A predictive coding account. *Progress in Neurobiology*, 192:101821, 2020. ISSN 0301-0082. doi: https://doi.org/10.1016/j.pneurobio.2020.101821. URL https://www.sciencedirect.com/science/article/pii/S0301008220300769.

Sergey Bartunov, Jack W Rae, Simon Osindero, and Timothy P Lillicrap. Meta-learning deep energy-based memory models. *arXiv preprint arXiv:1910.02720*, 2019.

Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 48(3):259–279, 1986.

Trenton Bricken and Cengiz Pehlevan. Attention approximates sparse distributed memory. *Advances in Neural Information Processing Systems*, 34, 2021.

Chiara Cirelli and Giulio Tononi. The why and how of sleep-dependent synaptic down-selection. *Seminars in Cell & Developmental Biology*, 125:91–100, 2022. ISSN 1084-9521. doi: https://doi.org/10.1016/j.semcdb.2021.02.007. URL https://www.sciencedirect.com/science/article/pii/S1084952121000318. Special Issue: Interplay between non-canonical and canonical Wnt signalling by Terry Van Raay/ Special Issue: The making of memories by Patricio Opazo and Victor Anggono.

Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, 28(11):2474–2504, nov 2016. doi: 10.1162/neco_a_00893. URL https://doi.org/10.1162%2Fneco_a_00893.

Constantine Dovrolis. A neuro-inspired architecture for unsupervised continual learning based on online clustering and hierarchical predictive coding, 2018. URL https://arxiv.org/abs/1810.09391.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014. URL https://arxiv.org/abs/1410.5401.

A.K. Gupta and D.K. Nagar. *Matrix Variate Distributions*. Monographs and Surveys in Pure and Applied Mathematics. CRC Press, 2018. ISBN 9781351433006. URL https://books.google.ca/books?id=Tl1ZDwAAQBAJ.

Jeff Hawkins and Sandra Blakeslee. *On Intelligence*. Times Books, New York, 2004.

Jeffrey Hawkins and Dileep George. Directed behavior using a hierarchical temporal memory based system, August 16 2007. US Patent App. 11/622,448.

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.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9639–9659. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/irie22a.html.

Pentti Kanerva. *Sparse distributed memory*. MIT press, 1988.

Augustine Kong, Jun S Liu, and Wing Hung Wong. Sequential imputations and bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288, 1994.

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

Dmitry Krotov. Hierarchical associative memory. *arXiv preprint arXiv:2107.06446*, 2021.

Dmitry Krotov and John Hopfield. Large associative memory problem in neurobiology and machine learning. *arXiv preprint arXiv:2008.06996*, 2020.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. 2016. doi: 10.48550/ARXIV.1606.01164. URL https://arxiv.org/abs/1606.01164.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Beren Millidge, Alexander Tschantz, and Christopher L. Buckley. Predictive coding approximates backprop along arbitrary computation graphs, 2020. URL https://arxiv.org/abs/2006.04182.

Beren Millidge, Anil Seth, and Christopher L Buckley. Predictive coding: a theoretical and experimental review, 2021. URL https://arxiv.org/abs/2107.12979.

Beren Millidge, Tommaso Salvatori, Yuhang Song, Thomas Lukasiewicz, and Rafal Bogacz. Universal hopfield networks: A general framework for single-shot associative memory models. *arXiv preprint arXiv:2202.04557*, 2022.

Alexander Ororbia and Daniel Kifer. The neural coding framework for learning generative models. *Nature Communications*, 13(1), April 2022. doi: 10.1038/s41467-022-29632-7. URL https://doi.org/10.1038/s41467-022-29632-7.

Alexander Ororbia, Ankur Mali, Daniel Kifer, and C. Lee Giles. Lifelong neural predictive coding: Learning cumulatively online without forgetting, 2019. URL https://arxiv.org/abs/1905.10696.

Alexander G. Ororbia, Ankur Mali, Daniel Kifer, and C. Lee Giles. Conducting credit assignment by aligning local representations, 2018. URL https://arxiv.org/abs/1803.01834.

Kha Pham, Hung Le, Man Ngo, Truyen Tran, Bao Ho, and Svetha Venkatesh. Generative pseudo-inverse memory. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Harn4_EZBw.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

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.

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, 2021.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR, 2021.

Jürgen Schmidhuber. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Computation*, 4(1):131–139, 01 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.1.131. URL https://doi.org/10.1162/neco.1992.4.1.131.

Philipp Seidl, Philipp Renz, Natalia Dyubankova, Paulo Neves, Jonas Verhoeven, Marwin Segler, Jörg K. Wegner, Sepp Hochreiter, and Günter Klambauer. Modern hopfield networks for few- and zero-shot reaction template prediction, 2021. URL https://arxiv.org/abs/2104.03279.

Sugandha Sharma, Sarthak Chandra, and Ila R. Fiete. Content addressable memory without catastrophic forgetting by heteroassociation with a fixed scaffold, 2022.

James Smith, Cameron Taylor, Seth Baer, and Constantine Dovrolis. Unsupervised progressive learning and the stam architecture. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2979–2987. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/410. URL https://doi.org/10.24963/ijcai.2021/410. Main Track.

K. Steinbuch. Die lernmatrix. *Kybernetik*, 1:36–45, 1961.

Alexander Tschantz, Beren Millidge, Anil K Seth, and Christopher L Buckley. Hybrid predictive coding: Inferring, fast and slow, 2022. URL https://arxiv.org/abs/2204.02169.

Michael Widrich, Bernhard Schäfl, Milena Pavlović, Hubert Ramsauer, Lukas Gruber, Markus Holzleitner, Johannes Brandstetter, Geir Kjetil Sandve, Victor Greiff, Sepp Hochreiter, et al. Modern hopfield networks and attention for immune repertoire classification. *Advances in Neural Information Processing Systems*, 33:18832–18845, 2020.

Michael Widrich, Markus Hofmarcher, Vihang Prakash Patil, Angela Bitto-Nemling, and Sepp Hochreiter. Modern hopfield networks for return decomposition for delayed rewards. In *Deep RL Workshop NeurIPS 2021*, 2021. URL https://openreview.net/forum?id=t0PQSDcqAiy.

Yan Wu, Greg Wayne, Alex Graves, and Timothy Lillicrap. The kanerva machine: A generative distributed memory. *arXiv preprint arXiv:1804.01756*, 2018a.

Yan Wu, Gregory Wayne, Karol Gregor, and Timothy Lillicrap. Learning attractor dynamics for generative memory. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018b. URL https://proceedings.neurips.cc/paper/2018/file/6e4243f5511fd6ef0f03e9f386d54403-Paper.pdf.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We mention that our memory can only store i.i.d. data, questions of its biological plausibility compared to GPCN, and how its performance rapidly declines relative to the baselines for longer sequences.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will release the code under the MIT License.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Refer to Section 6 and Appendix F.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We did in Appendix F.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We did not measure the total compute amount but listed the GPU type and the compute clusters used in Appendix F.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Refer to Section 6.
   (b) Did you mention the license of the assets? [Yes] Refer to Appendix F.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A BayesPCN's Auto-Associative and Hetero-Associative *read*

Algorithm 3 Auto-Associative Memory Read

Input: Memory log density $\log p(x^0, \mathbf{h}|x^0_{1:T})$, query vector $\bar{x}^0$
Output: Recall vector x^{0*}

Set the initial recall vector: $x^{0*} = \bar{x}^0$
repeat
  Find the most probable latent code given the input: $\mathbf{h}^* = \text{argmax}_{\mathbf{h}} \log p(x^{0*}, \mathbf{h}|x^0_{1:T})$
  Find the most probable input given the latent code: $x^{0*} = \text{argmax}_{x^0} \log p(x^0, \mathbf{h}^*|x^0_{1:T})$
until convergence or upper iteration limit

Algorithm 4 Hetero-Associative Memory Read

Input: Memory log density $\log p(x^0, \mathbf{h}|x^0_{1:T})$, query vector $\bar{x}^0 = (k, v)$ where the key k is known but the value v is arbitrarily initialized
Output: Recall vector x^{0*}

Find the most probable value and latent code given the key:
  $v^*, \mathbf{h}^* = \text{argmax}_{v,\mathbf{h}} \log p((k, v), \mathbf{h}|x^0_{1:T})$
Set the final recall vector: $x^{0*} = (k, v^*)$

B Derivation of BayesPCN *write*

$$p(\mathbf{W}|x^0_{1:t}) = \frac{1}{Z} p(\mathbf{W}|x^0_{1:t-1}) p(x^0_t|\mathbf{W}) \tag{12}$$

$$= \frac{1}{Z} p(\mathbf{W}|x^0_{1:t-1}) \mathbb{E}_{q(\mathbf{h}_t)}[\frac{p(x^0_t, \mathbf{h}_t|\mathbf{W})}{q(\mathbf{h}_t)}] \tag{13}$$

$$\approx \frac{1}{Z}[\sum_{n=1}^{N} \omega^{(n)}_{t-1} p^{(n)}(\mathbf{W}|x^0_{1:t-1}, \mathbf{h}^{(n)}_{1:t-1})] \mathbb{E}_{q(\mathbf{h}_t)}[\frac{p(x^0_t, \mathbf{h}_t|\mathbf{W})}{q(\mathbf{h}_t)}] \tag{14}$$

$$= \frac{1}{Z} \sum_{n=1}^{N} \omega^{(n)}_{t-1} p^{(n)}(\mathbf{W}|x^0_{1:t-1}, \mathbf{h}^{(n)}_{1:t-1}) \mathbb{E}_{q^{(n)}(\mathbf{h}_t)}[\frac{p(x^0_t, \mathbf{h}_t|\mathbf{W})}{q^{(n)}(\mathbf{h}_t)}] \tag{15}$$

$$\approx \frac{1}{Z} \sum_{n=1}^{N} \omega^{(n)}_{t-1} \frac{1}{S} \sum_{i=1}^{S} \frac{p^{(n)}(\mathbf{W}|x^0_{1:t-1}, \mathbf{h}^{(n)}_{1:t-1}) p(x^0_t, \mathbf{h}^{(n,i)}_t|\mathbf{W})}{q^{(n)}(\mathbf{h}^{(n,i)}_t)}, \quad \mathbf{h}^{(n,i)}_t \sim q^{(n)}(\mathbf{h}_t) \tag{16}$$

$$= \frac{1}{Z} \frac{1}{S} \sum_{n=1}^{N} \sum_{i=1}^{S} \omega^{(n)}_{t-1} \frac{p^{(n)}(x^0_t, \mathbf{h}^{(n,i)}_t|x^0_{1:t-1})}{q^{(n)}(\mathbf{h}^{(n,i)}_t)} p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n,i)}_{1:t}) \tag{17}$$

$$= \sum_{n=1}^{N} \sum_{i=1}^{S} \frac{\omega^{(n)}_{t-1} \frac{p^{(n)}(x^0_t, \mathbf{h}^{(n,i)}_t|x^0_{1:t-1})}{q^{(n)}(\mathbf{h}^{(n,i)}_t)}}{\sum_{n'=1}^{N} \sum_{i'=1}^{S} \omega^{(n')}_{t-1} \frac{p^{(n')}(x^0_t, \mathbf{h}^{(n',i')}_t|x^0_{1:t-1})}{q^{(n')}(\mathbf{h}^{(n',i')}_t)}} p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n,i)}_{1:t}) \tag{18}$$

$$= \sum_{n=1}^{N} \sum_{i=1}^{S} \omega^{(n,i)}_t p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n,i)}_{1:t}) \tag{19}$$

Equation 17 holds because

$$p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n)}_{1:t}) = \frac{p^{(n)}(\mathbf{W}|x^0_{1:t-1}, \mathbf{h}^{(n)}_{1:t-1}) p(x^0_t, \mathbf{h}^{(n,i)}_t|\mathbf{W})}{p^{(n)}(x^0_t, \mathbf{h}^{(n,i)}_t|x^0_{1:t-1})} \tag{20}$$

and Equation 18 holds because

$$Z = \int p(\mathbf{W}|x^0_{1:t-1})\mathbb{E}_{q^{(n)}(\mathbf{h}_t)}[\frac{p(x^0_t, \mathbf{h}_t|\mathbf{W})}{q^{(n)}(\mathbf{h}_t)}]d\mathbf{W} \tag{21}$$

$$\approx \int \frac{1}{S}\sum_{n=1}^{N}\sum_{i=1}^{S}\omega^{(n)}_{t-1}\frac{p^{(n)}(x^0_t, \mathbf{h}^{(n,i)}_t|x^0_{1:t-1})}{q^{(n)}(\mathbf{h}^{(n,i)}_t)}p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n,i)}_{1:t})d\mathbf{W} \tag{22}$$

$$= \frac{1}{S}\sum_{n=1}^{N}\sum_{i=1}^{S}\omega^{(n)}_{t-1}\frac{p^{(n)}(x^0_t, \mathbf{h}^{(n,i)}_t|x^0_{1:t-1})}{q^{(n)}(\mathbf{h}^{(n,i)}_t)} \tag{23}$$

Finally, if $S = 1$ we get

$$\hat{p}(\mathbf{W}|x^0_{1:t}) = \sum_{n=1}^{N}\frac{\omega^{(n)}_{t-1}\frac{p^{(n)}(x^0_t,\mathbf{h}^{(n)}_t|x^0_{1:t-1})}{q^{(n)}(\mathbf{h}^{(n)}_t)}}{\sum_{n'=1}^{N}\omega^{(n')}_{t-1}\frac{p^{(n')}(x^0_t,\mathbf{h}^{(n')}_t|x^0_{1:t-1})}{q^{(n')}(\mathbf{h}^{(n')}_t)}}p^{(n)}(\mathbf{W}|x^0_{1:t}, \mathbf{h}^{(n)}_{1:t}) \tag{24}$$

■

C Analytical Posterior Formulae for *write*

Let $z = f(x)$. Then, BayesPCN's top layer parameter update given $x^{0:L}$ is

$$\mu^* \leftarrow (\Sigma^{-1} + \frac{1}{\sigma_x^2}I)^{-1}(\Sigma^{-1}\mu + \frac{1}{\sigma_x^2}x^L) \tag{25}$$

$$\Sigma^* \leftarrow (\Sigma^{-1} + \frac{1}{\sigma_x^2}I)^{-1} \tag{26}$$

while the update for all other layers given $x^{0:L}$ is

$$R^{l*} \leftarrow R^l + U^l z^{l+1^\top}(z^{l+1}U^l z^{l+1^\top} + \sigma_x^2 I)^{-1}(x^l - z^{l+1}R^l) \tag{27}$$

$$U^{l*} \leftarrow U^l - U^l z^{l+1^\top}(z^{l+1}U^l z^{l+1^\top} + \sigma_x^2 I)^{-1}z^{l+1}U^l \tag{28}$$

D Analytical Diffusion Formulae for *forget*

Let $R_0^{1:L-1}, U_0^{1:L-1}, \mu_0, \Sigma_0$ be the memory prior parameters and β be the forget strength. Then, BayesPCN's top layer diffusion update is

$$\mu^* = \sqrt{1-\beta}\mu + (1 - \sqrt{1-\beta})\mu_0 \tag{29}$$

$$\Sigma^* = (1-\beta)\Sigma + \beta\Sigma_0 \tag{30}$$

while the update for all other layers is

$$R^{l*} = \sqrt{1-\beta}R^l + (1 - \sqrt{1-\beta})R_0^l \tag{31}$$

$$U^{l*} = (1-\beta)U^l + \beta U_0^l \tag{32}$$

E Connections to Hopfield Networks

We show that modern Hopfield network's recall is equivalent to the recall of a BayesPCN model with $L = 0$. BayesPCN's activation log density is defined as $\log p(x^0, \mathbf{h}) = \log\left(\sum_{n=1}^{N}\omega^{(n)}p^{(n)}(x^0, \mathbf{h})\right)$, and its gradient w.r.t. the activations is

$$\nabla_{x^0,\mathbf{h}}\log p(x^0, \mathbf{h}) = \sum_{n=1}^{N}\frac{\exp(\log\omega^{(n)}p^{(n)}(x^0, \mathbf{h}))}{\sum_{n'=1}^{N}\exp(\log\omega^{(n')}p^{(n')}(x^0, \mathbf{h}))}\nabla_{x^0,\mathbf{h}}\log p^{(n)}(x^0, \mathbf{h}), \tag{33}$$

On the other hand, MHN's energy is

$$E(q) = \frac{\beta}{2} q q^T - \log \sum_{j=1}^{N} \exp(\beta q K_j^T) \tag{34}$$

where $q \in \mathbb{R}^{1 \times d_k}$ is the query row vector, $K \in \mathbb{R}^{N \times d_k}$ is the key matrix, and $K_j \in \mathbb{R}^{1 \times d_k}$ is the j-th key row vector. q and K correspond to x^0 and W^0 in our paper's notation. The negative of the energy can be converted to the following Gaussian mixture log density.

$$\log p(q) = \log \left(\sum_{j=1}^{N} \frac{e^{\frac{\beta}{2} K_j K_j^T}}{\sum_{j'=1}^{N} e^{\frac{\beta}{2} K_{j'} K_{j'}^T}} \frac{1}{(2\pi\beta^{-1})^{\frac{d_k}{2}}} e^{-\frac{\beta}{2}(q-K_j)(q-K_j)^T} \right) \tag{35}$$

Recall in both BayesPCN and MHN is gradient ascent on the above log density. When we take the gradient with respect to the input vector q, we recover Equation 33.

$$\nabla_q \log p(q) = \sum_{j=1}^{N} \frac{e^{\frac{\beta}{2} q K_j^T}}{\sum_{j'=1}^{N} e^{\frac{\beta}{2} q K_{j'}^T}} \beta (K_j - q) \tag{36}$$

$$= \sum_{n=1}^{N} \frac{\exp(\log \omega^{(n)} p^{(n)}(x^0, \mathbf{h}))}{\sum_{n'=1}^{N} \exp(\log \omega^{(n')} p^{(n')}(x^0, \mathbf{h}))} \nabla_{x^0, \mathbf{h}} \log p^{(n)}(x^0, \mathbf{h}) \tag{37}$$

$$\text{where} \quad \omega^{(n)} = \frac{\beta}{2} K_n K_n^T, \quad \log p^{(n)}(x^0, \mathbf{h}) = \log \mathcal{N}(q; K_n, \frac{1}{\beta} I) \tag{38}$$

■

We conclude that recall in Modern Hopfield Network is equivalent to recall under our framework, which is gradient descent on the log joint of a normal mixture w.r.t. neuron activations, where there are no hidden layers ($L = 0$). However training our model with $L = 0$ does not lead to the same memory update as the suggested training procedure of MHN, which is setting each key vector K_j to some observed datapoint.

Universal Hopfield network [Millidge et al., 2022] proposes a framework for single-shot associative memory that decomposes recall into three components: similarity function, separation function, and projection matrix. Let $\mathbf{x} = (\bar{x}^0, \mathbf{h})$ be the row vector of all initial network activations when give the query $\bar{x}^0$. Equation 33 suggests that BayesPCN $read$'s implementation of those components is per particle weighted log joint $\log \omega^{(n)} p^{(n)}(\mathbf{x})$ of the query $\bar{x}^0$ for the similarity function, $softmax$ for the separation function, and the matrix $\begin{bmatrix} \mathbf{x} + \gamma \nabla_{\mathbf{x}} \log p^{(1)}(\mathbf{x}) \\ \vdots \\ \mathbf{x} + \gamma \nabla_{\mathbf{x}} \log p^{(N)}(\mathbf{x}) \end{bmatrix}$ for the projection matrix where γ is the learning rate. We can accommodate the fact that BayesPCN's $read$ does iterated conditional modes by zeroing out the fixed variable gradients in the projection matrix.

F Additional Experiment Details

All GPCN and BayesPCN models had $\sigma_W = 1, \sigma_x = 0.01$, and used Adam with learning rate 0.01 as the neuron activation gradient descent optimizer. All energy minimization w.r.t. the neuron activations took 500 gradient steps during the $write$ phase. During the $read$ phase, the outer loop iterated conditional mode in Algorithm 3 was repeated 30 times and the energy minimization in Algorithm 4 took 30×500 gradient steps. MHN models had $\beta = 10,000$ (equivalent to $\sigma_x = 0.01$) and used Adam with learning rate 1.0 as the gradient descent optimizer. All experiments were run on CIFAR10 and/or Tiny ImageNet datasets (both of which have the MIT License) and the image pixel values were normalized to fall between $[-1, 1]$. Online GPCNs took a single gradient step w.r.t. the network weights after the hidden activations converged per observation, a treatment consistent with that of the fast weight memory in Schlag et al. [2021]. BayesPCN $forget$ models had four hidden layers of width 1024, a single particle, and GELU activations. All hyperparameter ranges were chosen based on Salvatori et al. [2021]'s experiments and GPCN/BayesPCN's empirical results. Lastly, experiments used NVIDIA Tesla V100 GPUs and were run on the university's internal clusters.

White Noise CIFAR10 MSE				
Sequence Length	128	256	512	1024
Identity	0.1596 ± 0.0003	0.1600 ± 0.0001	0.1600 ± 0.0000	0.1600 ± 0.0001
MHN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
GPCN (Offline)	0.0029 ± 0.0000	0.0047 ± 0.0001	0.0074 ± 0.0001	0.0121 ± 0.0001
GPCN (Online)	0.0102 ± 0.0002	0.0150 ± 0.0001	0.0191 ± 0.0000	0.0210 ± 0.0001
BayesPCN	0.0017 ± 0.0002	0.0079 ± 0.0015	0.0146 ± 0.0001	0.0339 ± 0.0057
BayesPCN ($forget$)	0.0064 ± 0.0001	0.0101 ± 0.0001	0.0145 ± 0.0001	0.0180 ± 0.0001

White Noise Tiny ImageNet MSE				
Sequence Length	128	256	512	1024
Identity	0.1600 ± 0.0000	0.1602 ± 0.0000	0.1601 ± 0.0000	0.1600 ± 0.0001
MHN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
GPCN (Offline)	0.0005 ± 0.0000	0.0010 ± 0.0000	0.0019 ± 0.0000	0.0058 ± 0.0004
GPCN (Online)	0.0082 ± 0.0005	0.0109 ± 0.0000	0.0137 ± 0.0001	0.0180 ± 0.0001
BayesPCN	0.0011 ± 0.0001	0.0033 ± 0.0000	0.0064 ± 0.0000	0.6223 ± 0.0282
BayesPCN ($forget$)	0.0027 ± 0.0001	0.0059 ± 0.0001	0.0108 ± 0.0000	0.0176 ± 0.0000

Dropout CIFAR10 MSE				
Sequence Length	128	256	512	1024
Identity	1.1140 ± 0.0059	1.1178 ± 0.0009	1.1353 ± 0.0014	1.1481 ± 0.0010
MHN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
GPCN (Offline)	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0003 ± 0.0000
GPCN (Online)	0.0023 ± 0.0000	0.0034 ± 0.0001	0.0055 ± 0.0001	0.0075 ± 0.0000
BayesPCN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0049 ± 0.0018
BayesPCN ($forget$)	0.0000 ± 0.0000	0.0001 ± 0.0000	0.0004 ± 0.0000	0.0019 ± 0.0000

Dropout Tiny ImageNet MSE				
Sequence Length	128	256	512	1024
Identity	1.0629 ± 0.0014	1.0889 ± 0.0006	1.1154 ± 0.0005	1.1072 ± 0.0008
MHN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
GPCN (Offline)	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0001 ± 0.0000
GPCN (Online)	0.0044 ± 0.0003	0.0061 ± 0.0003	0.0077 ± 0.0001	0.0106 ± 0.0000
BayesPCN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0175 ± 0.0000
BayesPCN ($forget$)	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0002 ± 0.0000	0.0009 ± 0.0000

Mask CIFAR10 MSE				
Sequence Length	128	256	512	1024
Identity	1.1272 ± 0.0000	1.1373 ± 0.0000	1.1619 ± 0.0000	1.1653 ± 0.0000
MHN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0011 ± 0.0000	0.0000 ± 0.0000
GPCN (Offline)	0.0002 ± 0.0000	0.0006 ± 0.0000	0.0011 ± 0.0000	0.0028 ± 0.0001
GPCN (Online)	0.0136 ± 0.0002	0.0281 ± 0.0013	0.0547 ± 0.0005	0.0805 ± 0.0005
BayesPCN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0007 ± 0.0001	0.1375 ± 0.0127
BayesPCN ($forget$)	0.0000 ± 0.0000	0.0007 ± 0.0000	0.0101 ± 0.0001	0.0544 ± 0.0011

Mask Tiny ImageNet MSE				
Sequence Length	128	256	512	1024
Identity	1.0876 ± 0.0000	1.0884 ± 0.0000	1.0982 ± 0.0000	1.1132 ± 0.0000
MHN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0010 ± 0.0000
GPCN (Offline)	0.0001 ± 0.0000	0.0004 ± 0.0000	0.0005 ± 0.0000	0.0018 ± 0.0000
GPCN (Online)	0.0105 ± 0.0016	0.0357 ± 0.0028	0.0491 ± 0.0004	0.0735 ± 0.0004
BayesPCN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0003 ± 0.0000	0.0673 ± 0.0002
BayesPCN ($forget$)	0.0000 ± 0.0000	0.0003 ± 0.0000	0.0031 ± 0.0000	0.0284 ± 0.0000

Table 2: Table 1 results with standard deviations calculated across three seeds.

G BayesPCN vs MHN Recall on Highly Noised Queries

This section reports the recall performance of MHN and BayesPCN models on high query noise associative recall tasks. The experiment setup is similar to that of Section 6 aside from the fact that the white noise tasks had the noise standard deviation set to 0.8 instead of 0.2, the dropout tasks randomly blacked out 75% of the pixels instead of 25%, and the masking tasks blacked out 75% of the rightmost pixels instead of 25%. Because the high query noise task is harder, we show the recall result after the models observed 16, 32, 64, and 128 datapoints. BayesPCN models had four hidden layers of width 256, a single particle, and GELU activations. MHNs again used $\beta = 10,000$.

White Noise CIFAR10 MSE				
Sequence Length	16	32	64	128
Identity	2.5564 ± 0.0086	2.5525 ± 0.0023	2.5586 ± 0.0105	2.5565 ± 0.0052
MHN	0.0086 ± 0.0000	0.0023 ± 0.0000	0.0105 ± 0.0000	0.0052 ± 0.0000
BayesPCN	0.0050 ± 0.0009	0.0127 ± 0.0031	0.0241 ± 0.0004	0.0750 ± 0.0006

White Noise Tiny ImageNet MSE				
Sequence Length	16	32	64	128
Identity	2.5586 ± 0.0105	2.5565 ± 0.0052	2.5584 ± 0.0047	2.5582 ± 0.0036
MHN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
BayesPCN	0.0010 ± 0.0001	0.0019 ± 0.0004	0.0064 ± 0.0016	0.0248 ± 0.0003

Dropout CIFAR10 MSE				
Sequence Length	16	32	64	128
Identity	1.0060 ± 0.0054	1.0276 ± 0.0044	1.0691 ± 0.0026	1.1095 ± 0.0018
MHN	0.3517 ± 0.0027	0.3596 ± 0.0029	0.3635 ± 0.0031	0.3840 ± 0.0010
BayesPCN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000

Dropout Tiny ImageNet MSE				
Sequence Length	16	32	64	128
Identity	1.0365 ± 0.0022	1.0746 ± 0.0012	1.1418 ± 0.0007	1.0625 ± 0.0011
MHN	0.4967 ± 0.0026	0.5096 ± 0.0006	0.5741 ± 0.0021	0.5630 ± 0.0036
BayesPCN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000

Mask CIFAR10 MSE				
Sequence Length	16	32	64	128
Identity	0.9686 ± 0.0000	0.9941 ± 0.0000	1.0563 ± 0.0000	1.0987 ± 0.0000
MHN	0.3361 ± 0.0000	0.3315 ± 0.0000	0.3650 ± 0.0000	0.3957 ± 0.0000
BayesPCN	0.0000 ± 0.0000	0.0001 ± 0.0000	0.0001 ± 0.0000	0.0011 ± 0.0000

Mask Tiny ImageNet MSE				
Sequence Length	16	32	64	128
Identity	1.0019 ± 0.0000	1.0606 ± 0.0000	1.1370 ± 0.0000	1.0597 ± 0.0000
MHN	0.4534 ± 0.0000	0.4896 ± 0.0000	0.7033 ± 0.0000	0.6378 ± 0.0000
BayesPCN	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0001 ± 0.0002

Table 3: Average MSE between the training images and the associative memory $read$ outputs for the high query noise white noise, dropout, and mask tasks on CIFAR10 and Tiny ImageNet datasets.

Figure 4: Sample MHN (**left**) and BayesPCN (**right**) $read$ outputs for the white noise ($\sigma = 0.8$), pixel dropout (75%), and pixel masking (75%) tasks. The top row contains the memory $read$ inputs and the bottom row contains the memory $read$ outputs.

H BayesPCN Scaling, Generalization, and Sampling

H.1 Scaling

Figure 5 qualitatively demonstrates how BayesPCN's read scales with the number of stored datapoints for the CIFAR10 recall tasks. BayesPCN models are able to output images that are very close to the original image even when the inputs are significantly corrupted. As the number of observations increases, the $read$ operation's basin of attraction decreases as it is still able to reconstruct the original image but gets worse at recovering the original image given a corrupted version of it.

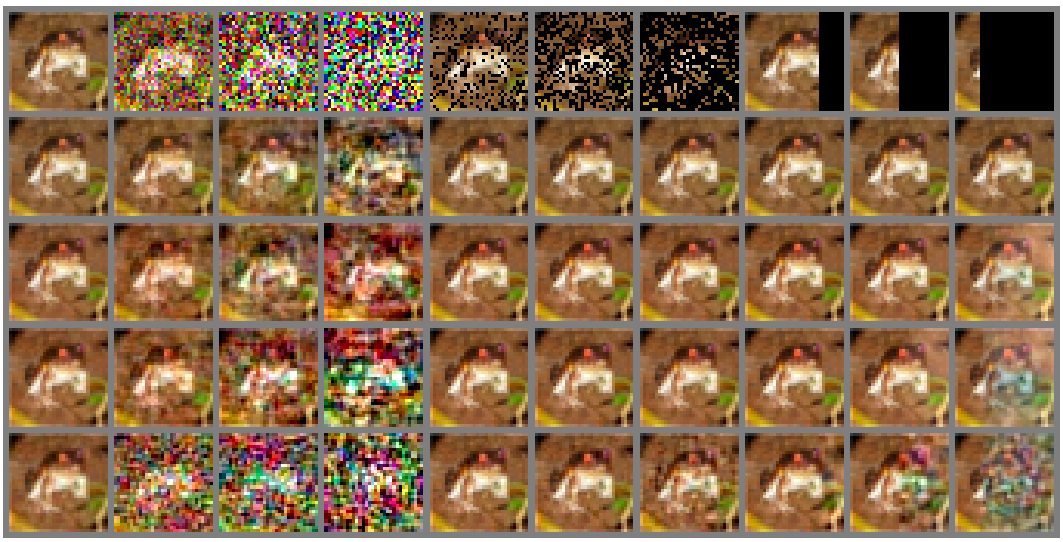

Figure 5: Example BayesPCN recall result from the CIFAR10 auto-associative and hetero-associative tasks. From top to bottom rows are the comparisons of BayesPCN's $read$ inputs and outputs where the task is to recover the first observation seen during the $write$ phase (the top left frog image) after sequentially observing 127, 255, 511, and 1023 additional datapoints.

Figure 6 illustrates how BayesPCN's recall accuracy scales with the network width and depth. A recall is considered correct if the MSE between the ground truth and the recalled data is less than 0.01. We find that the increased network width and depth is generally helpful across all tasks. We note that because the plots depict recall accuracy not MSE, if the memory performance generally declines and the average recall MSE exceeds 0.01, this can lead to very low accuracy even if the actual recall MSE is not much greater than 0.01.

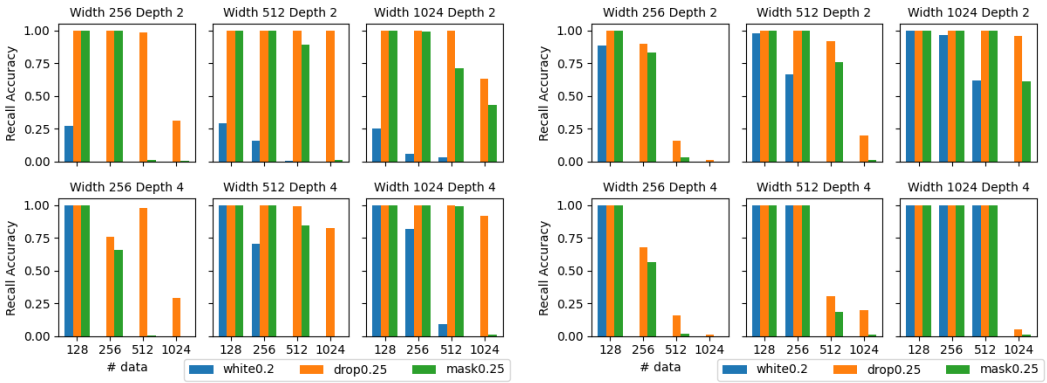

Figure 6: Recall accuracy of BayesPCN with different network width and depth on CIFAR10 (**left**) and Tiny ImageNet (**right**) tasks. The reported values are from the best models of each category.

H.2 Generalization

Figure 7 illustrates BayesPCN's read outputs for unseen image queries after different number of datapoints have been stored into memory. As BayesPCN observes more data, it learns to "generalize" and gets better at reconstructing and even mildly removing white noise from unseen images. This can be attributed to the model continual learning its internal representation that better "describe" the data distribution. We expected this behaviour to occur since S-NCN [Ororbia et al., 2019], a model similar in structure to GPCN, could continually learn to perform discriminative tasks.

Figure 7: BayesPCN's $read$ outputs given ground truth (**left**) and noised (**right**) unseen images as inputs after observing $2, 8, 32, 128, 512$ training images.

H.3 Sampling

Both GPCN and BayesPCN at the core are as much generative models as they are associative memories. We examine the quality of ancestral sampling samples from both models in Figure 8. When ancestral sampling, BayesPCN did not marginalize out the synaptic weights and instead fixed them to the mean parameters of $p(\mathbf{W}|x^0_{1:t-1}, \mathbf{h}^{(n)}_{1:t-1})$.

(a) GPCN trained on 4 observations. (b) GPCN trained on 1024 observations. (c) BayesPCN trained on 4 observations. (d) BayesPCN trained on 1024 observations.

Figure 8: Ancestral sampling results from GPCN and BayesPCN models trained on CIFAR10.

We find that both GPCN and BayesPCN samples are superpositions of the training images. However, as BayesPCN is trained on more and more observations, its sample quality quickly deteriorates. We hypothesize that the poor sample quality for both GPCN and BayesPCN stems from the approximate nature of their parameter estimation. For example, BayesPCN's particle count would have to be much greater than 4 to accurately capture the true posterior $p(\mathbf{W}|x^0_{1:t-1})$ using its sequential importance sampling estimate and the variational distribution over the hidden activations should not be Dirac distributed. However, we note that Ororbia and Kifer [2022] has successfully trained predictive coding networks to be good generative models.