On the Relationship Between Variational Inference and Auto-Associative Memory

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this article, we propose a variational inference formulation of memory retrieval in auto-associative memories, allowing us to combine perceptual inference with representation learning into the same mathematical framework. In this formulation, the prior probability distribution onto representations is made memory dependent, thus pulling the inference process towards stored representations. We then study how different neural network approaches to variational inference can be applied in this framework. We compare methods relying on amortized inference such as Variational Auto Encoders and methods relying on iterative inference such as Predictive Coding and suggest combining both approaches to design new auto-associative memory models. We evaluate the obtained algorithms on the CIFAR10 and CLEVR image datasets and compare them with other associative memory models such as Hopfield Networks, End-to-End Memory Networks and Neural Turing Machines.

1 Introduction

In the recent years, many methods have been proposed in order to augment deep learning models with long-term memories. These models provide writing and reading mechanisms allowing to store and retrieve certain patterns inside a memory. Among these models, associative memories are memories that perform content-based addressing, meaning that they retrieve stored patterns based on an approximate or incomplete version of those.

While many models consider one-step reading mechanisms to retrieve patterns from the memory, biological memory retrieval is an iterative process that can take a variable amount of time. Taking this into consideration, Hopfield networks [14] have been proposed as an Associative Memory (AM) implemented by a Recurrent Neural Network (RNN). Hopfield networks perform memory retrieval as an iterative process where stored patterns constitute local attractors of the RNN dynamics.

In this work, we show how AM models such as Hopfield networks can be formulated as variational inference methods using a memory dependent probabilistic model of the observed data. In variational inference algorithms such as Variational Auto-Encoders (VAE) [18, 31] and Predictive Coding (PC) [30, 11], the representation inferred from the observed data tries to minimize an energy function based on the probabilistic model $p(\boldsymbol{x})$ generating the observed data. In the Free Energy Principle (FEP) literature [9, 10], this function is called Variational Free-Energy (VFE), and is equivalent to the negative Evidence Lower Bound (ELBO) more often used in machine learning. The probabilistic model $p(\boldsymbol{x})$ can be decomposed according to a prior probability distribution over representations $p(\boldsymbol{z})$, and a likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ describing the probability of observed data based on the representation. We propose to make $p(\boldsymbol{z})$ depend on the patterns stored in memory, which allows us to derive a new expression of the VFE. The obtained energy function has minima that should be close to one of the stored patterns while properly encoding the observed input. As such, inference based on

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

this energy function can be seen both as perceptual inference and memory retrieval. While using memory-dependent generative models or Gaussian Mixture Models (GMM) with VAEs [8, 2, 22] are not novel ideas, the PC approach has never been applied to this problem.

The paper is organized as follows: in section 2, we provide a deeper presentation of the concepts connected to our work and review related approaches. In section 3 we present an overview of the FEP mathematical framework and derive an expression of the VFE depending on the patterns stored in memory. In section 4 we design several AM models minimizing this energy function. In section 5, we evaluate the obtained algorithms on the task of memory retrieval on two image datasets, and compare their performance with other AM models.

The contributions brought by this work are the following:

- We design four AM models based on a variational inference formulation of memory retrieval.
- We draw a connection between PC and the modern continuous Hopfield network [29].

2 Related work

2.1 Auto-associative memory

Several approaches to the long-term storage of information in artificial neural networks have been proposed. Memory networks [37], and End-to-End Memory Networks (MemN2N) [33] propose to store information in a memory matrix that can be addressed using attention coefficients computed based on the content of different memory locations. The Neural Turing Machine (NTM) [12] and the differentiable neural computer [13] combine this content-based addressing using attention mechanisms with a location-based addressing allowing more computer-like memory accesses.

Standing out from these approaches that consider a feedforward reading mechanism, Hopfield networks [14] and continuous Hopfield networks [15] instead store patterns as local attractors of an RNN. These memories can thus be addressed by initializing the RNN state with the input pattern, and retrieving the stored attractor after convergence. In order to improve the memory capacity, modern Hopfield networks [21, 20, 7] propose several variants of the energy function using polynomial or exponential interactions. Extending these models to the continuous case, [29] proposed the Modern Continuous Hopfield Network (MCHN) with update rules implementing self attention, that they relate to the transformer model [35]. In [25], the authors introduce a general Hopfield network framework where the update rules are built using three components: a similarity function, a separation function, and a projection function.

It has been shown that overparameterized auto-encoders also implement AM [27, 32]. These methods embed the stored patterns as attractors through training, and retrieval is performed by iterating over the auto-encoding loop. In contrast, our methods allow one-shot writing: a new pattern $\boldsymbol{x}$ can be inserted in memory simply by computing its representation and adding it to the memory matrix $\boldsymbol{M}$.

Another line of research takes inspiration from the Sparse Distributed Memory model [17], building connections with attention mechanisms [3] and with the variational inference framework [39, 40, 28]. In particular, the Kanerva Machine [39, 40, 28] is similar to the models we build in many aspects: they use an iterative reading mechanism, a memory-dependent prior on the representation, all within the variational inference framework. Though, the PC toolbox provides methods that set the proposed models apart from these approaches, as detailed in the methods section.

2.2 Predictive Coding

PC is a theory of brain function [30, 6] extending the idea that neural representations emerge as part of an inference process of the causes of sensory observations, as already suggested by Helmholtz in 1867 [36]. The FEP [9, 10] provides a principled derivation of PC networks based on the minimization of variational free-energy (VFE), a quantity equivalent to the negative evidence lower bound (ELBO) used in variational Bayesian methods, and defined as:

$$F(\boldsymbol{x}) = \int_z \log\Big(\frac{q(\boldsymbol{z})}{p(\boldsymbol{x},\boldsymbol{z})}\Big) q(\boldsymbol{z}) \boldsymbol{dz} \tag{1}$$

where $p(\boldsymbol{x}, \boldsymbol{z})$ denotes the generative model and $q(\boldsymbol{z})$ denotes the approximate posterior (also called recognition density) on $\boldsymbol{z}$. VAEs [18, 31] are a well-known method applying the idea of VFE minimization (equivalently ELBO maximization) to neural networks. In VAEs, this quantity is only optimized during the model training, and perceptual inference is performed as a simple forward pass through the encoder.

In contrast, neural network models based on PC intertwine the prediction (decoder) and inference (encoder) mechanisms within a single hierarchical recurrent architecture comprising a population of representation neurons and a population of prediction error neurons at each layer. Perceptual inference is then an iterative mechanism supported by the dynamics of this RNN. Given an observed input $\boldsymbol{x}$, the representation $\boldsymbol{z}$ is updated at each time step based on a bottom-up signal pushing $\boldsymbol{z}$ towards values that minimize the reconstruction error, and on a top-down signal pushing $\boldsymbol{z}$ towards values that maximize its prior probability $p(\boldsymbol{z})$. While this prior probability on $\boldsymbol{z}$ is often ignored, we show that it can bring auto-associative capacities to the PC network, and we draw a connection between the obtained mechanism and the modern continuous Hopfield network proposed in [29].

While both VAEs and PC networks output an estimation of the approximate posterior $q(\boldsymbol{z})$, they differ in the computational mechanisms used for inference: VAEs perform amortized inference via a forward pass through an encoder, while PC networks perform an iterative inference implemented by its recurrent dynamics. Some works have suggested combining both approaches: the iterative inference can be initialized using the estimation provided by the amortized inference method [1, 34].

3 Memory-based Variational Inference

3.1 Framework definition

Our framework is based on a generative model $p(\boldsymbol{x}, \boldsymbol{z})$ that can be factored into a prior probability $p(\boldsymbol{z})$ and a likelihood $p(\boldsymbol{x}|\boldsymbol{z})$. The prior probability on $\boldsymbol{z}$ is defined as a memory dependent distribution:

$$p(\boldsymbol{z}) = p(\boldsymbol{z}; \boldsymbol{M}) \tag{2}$$

where the vectors $\boldsymbol{M}_k$ (the columns of $\boldsymbol{M}$) constitute a repertoire of representations.

The probability $p(\boldsymbol{x}|\boldsymbol{z})$ is defined as a hierarchical generative model featuring several intermediate variables $\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_{L-1}\}$, where L denotes the number of layers. By extension we use the notations $\boldsymbol{h}_0 = \boldsymbol{x}$ and $\boldsymbol{h}_L = \boldsymbol{z}$. We assume that the generative model is a cascade of multivariate Gaussians:

$$p(\boldsymbol{h}_l|\boldsymbol{h}_{l+1}) = \mathcal{N}(\boldsymbol{h}_l; \boldsymbol{f}^l_{\boldsymbol{\theta}}(\boldsymbol{h}_{l+1}), \mathbb{I}) \tag{3}$$

where $\mathbb{I}$ is the covariance matrix of the Gaussians and is uniform across all layers, and $\boldsymbol{f}^l_{\boldsymbol{\theta}}$ are functions (typically neural network layers) parameterized by $\boldsymbol{\theta}$. Note that this can be adapted to arbitrary computation graphs by replacing $\boldsymbol{h}_{l+1}$ by the set of parent nodes of $\boldsymbol{h}_l$ for each node. For simplicity, we assume in the following derivations that each node only has one parent. In our experiments, the functions $\boldsymbol{f}^l_{\boldsymbol{\theta}}$ correspond to the different layers of a Convolutional Neural Network (CNN) on the CIFAR10 dataset, and to a MONet [5] decoder on the CLEVR dataset. We denote by $\boldsymbol{f}_{\boldsymbol{\theta}}$ the composition of all layers $\boldsymbol{f}^l_{\boldsymbol{\theta}}$.

We can derive the VFE corresponding to the described generative model. The FEP formulation of PC uses different approximations that allow us to greatly simplify the expression of the VFE introduced in equation (1). We refer to appendix A for detailed derivations and simply provide the simplified expression:

$$F(\boldsymbol{x}, \hat{\boldsymbol{h}_1}, \ldots, \hat{\boldsymbol{h}_{L-1}}, \hat{\boldsymbol{z}}) = \sum_{l=0}^{L-1} \frac{1}{2} \|\hat{\boldsymbol{h}_l} - \boldsymbol{f}^l_{\boldsymbol{\theta}}(\hat{\boldsymbol{h}_{l+1}})\|^2 - \log p(\hat{\boldsymbol{z}}; \boldsymbol{M}) + C \tag{4}$$

where C is a quantity independent from $\{\boldsymbol{x}, \hat{\boldsymbol{h}_1}, \ldots, \hat{\boldsymbol{h}_{L-1}}, \hat{\boldsymbol{z}}\}$. The vectors $\{\hat{\boldsymbol{h}_1}, \ldots, \hat{\boldsymbol{h}_{L-1}}, \hat{\boldsymbol{z}}\}$ correspond to the means of the approximate posterior $q(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{L-1}, \boldsymbol{z}) = q(\boldsymbol{z}) \prod_{i=1}^{L-1} q(\boldsymbol{h_i})$.

The term corresponding to the prior probability is often omitted, which can be justified as being equivalent to having no prior preferences over different values of $\boldsymbol{z}$. Key to our method is the idea that this prior can pull the inference process towards values of $\boldsymbol{z}$ previously stored in the memory $\boldsymbol{M}$, turning the patterns $\boldsymbol{M}_k$ into attractors of the PC network. As such, a suitable distribution would be one that associates high probabilities for the patterns stored in memory.

3.2 Classification of related methods in this framework

Here, we show that we can formulate Modern Continuous Hopfield Networks (MCHN) as PC networks derived from this expression of the VFE under some conditions:

- There is no representation component: $L = 0$.
- The PC network is initialized with $\hat{\boldsymbol{z}} = \boldsymbol{x}$.
- The prior distribution $p(\boldsymbol{z}; \boldsymbol{M})$ is defined as:

$$p_{MCHN}(\boldsymbol{z}; \boldsymbol{M}) = \sum_{k=1}^{N} \pi_k \mathcal{N}\big(\boldsymbol{z}; \boldsymbol{M}_k, \beta^{-1}\mathbb{I}\big) \tag{5}$$

$$\text{with } \forall k, \pi_k = \frac{\exp\{\frac{\beta}{2}\boldsymbol{M}_k^\intercal \cdot \boldsymbol{M}_k\}}{\sum_{k'=1}^{N} \exp\{\frac{\beta}{2}\boldsymbol{M}_{k'}^\intercal \cdot \boldsymbol{M}_{k'}\}} \tag{6}$$

This prior distribution is a Gaussian Mixture Model (GMM) with mixture means corresponding to the N stored patterns and mixing coefficients π_k depending on the patterns' Euclidean norm. It is parameterized by a coefficient $\beta > 0$. Based on these assumptions, we can derive the following expression for the VFE:

$$F(\hat{\boldsymbol{z}}) = \frac{\beta}{2}\hat{\boldsymbol{z}}^\intercal \cdot \hat{\boldsymbol{z}} - \log \sum_{k=1}^{N} \exp\{\beta \hat{\boldsymbol{z}}^\intercal \cdot \boldsymbol{M}_k\} + C \tag{7}$$

which is, up to a constant C and a factor β, equivalent to the MCHN energy function proposed in [29]. The complete derivations of this expression are provided in appendix B. We can note that VFE does not depend on the input $\boldsymbol{x}$. The input is not part of the energy function, but serves as an initial estimate of the approximate posterior mean $\hat{\boldsymbol{z}}$. According to the FEP formulation of PC, applying gradient descent on the VFE with regard to the approximate posterior mean $\hat{\boldsymbol{z}}$ yields the update rules of the PC network. Using our expression of the VFE, we obtain:

$$\hat{\boldsymbol{z}} \leftarrow \text{softmax}(\beta \hat{\boldsymbol{z}} \cdot \boldsymbol{M})\boldsymbol{M}^\intercal \tag{8}$$

This equation is identical to the update rule of MCHN (detailed derivations are provided in appendix). We have shown that our framework can be related to MCHNs, but variational inference methods for perceptual inference can also be retrieved in this framework by simply using a neutral prior.

For VAEs and variants, this prior probability distribution is exploited during learning, but the amortized inference cannot be dynamically adapted to this distribution. Therefore, to take into account a change in this prior, we would need to retrain the encoder. In the derivations of PC networks [11, 4], the term $p(\boldsymbol{z})$ is often ignored, which is equivalent to having a flat prior distribution. However, theoretically the iterative inference could take into account a prior distribution, and dynamically adapt to changes in this distribution.

In the family of iterative inference algorithms, we can also mention the methods based on backpropagation (BP) to estimate the representation. Instead of using neural computations to simulate the gradient descent on the energy function as done in PC, these methods directly use BP to optimize the representation $\hat{\boldsymbol{z}}$. For instance [26] minimize prediction error using BPTT to adjust a latent variable in an RNN. This method is in fact very similar to PC, since it has been shown that under some conditions, PC approximates the update rules entailed by BP applied on the reconstruction error [38, 24]. We provide in table 1 a simple classification of these approaches depending on whether they use a representation and memory component, and whether the inference mechanism is amortized or iterative.

Table 1: Summary of the related models and proposed models.

Model	Representation	Associative memory
MCHN [29]	✗	Iterative
VAE [18]	Amortized	✗
PC [30]	Iterative	✗
BP [26]	Iterative	✗
HPC [34]	Amortized and iterative	✗
GMVAE [8]	Amortized	Amortized
Overparameterized VAE [27]	Amortized	Iterative
Kanerva Machines [39]	Amortized	Iterative
VAE-PC-GMM (Ours)	Amortized and iterative	Iterative
VAE-BP-GMM (Ours)	Amortized and iterative	Iterative
VAE-GMM (Ours)	Amortized	Iterative

4 Methods

In this section, we present the AM models that we have designed using this framework. All these models are based on a pre-trained VAE that is used to provide initial estimates $\tilde{\boldsymbol{z}}$ and predictions of $\boldsymbol{x}$ based on $\tilde{\boldsymbol{z}}$. The proposed methods are represented in figure 1.

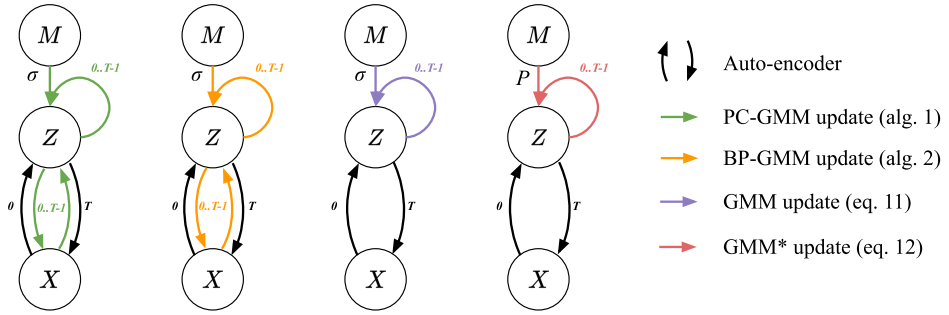

Figure 1: Representation of the memory retrieval process in the different proposed models. From left to right: VAE-PC-GMM, VAE-BP-GMM, VAE-GMM and VAE-GMM*.

4.1 Probabilistic model

The prior probability over the representation $\boldsymbol{z}$ is crucial to obtain a model that infers representations close to stored patterns. We have shown that MCHN can be obtained starting from a GMM prior biased towards stored patterns of large Euclidean norm. Since there is no intuitive justification for this bias, we choose to instead start with a balanced GMM prior distribution:

$$p(\boldsymbol{z}; \boldsymbol{M}) = \sum_{k=1}^{N} \frac{1}{N} \mathcal{N}(\boldsymbol{z}; \boldsymbol{M}_k, \Sigma) \tag{9}$$

where Σ denotes the covariance matrix, uniform across the N mixtures. In the following models, we consider the simpler case where $\boldsymbol{\Sigma} = \sigma^2 \mathbb{I}$, except for the last model where $\boldsymbol{P} = \boldsymbol{\Sigma}^{-1}$ is trained.

4.2 PC based inference (VAE-PC-GMM)

Following the FEP formulation of PC, we can derive the gradient of F according to each layer's representation $\hat{\boldsymbol{h}}_l$. As such, performing gradient descent on F yields a system of update rules on each representation, that can be interpreted as RNN dynamics. This PC network comprises at each

layer two populations of neurons, one encoding the layer's representation, and one encoding the layer prediction error $\boldsymbol{\epsilon}_l = \hat{\boldsymbol{h}}_l - \boldsymbol{f}^l_{\boldsymbol{\theta}}(\hat{\boldsymbol{h}_{l+1}})$.

Detailed derivations of this model are provided in appendix C, where the forward pass through the model is given in algorithm 1. Since the RNN dynamics implement a gradient descent, the initialization of this network is responsible for the local minimum to which it converges. We label VAE-PC-GMM the version of this algorithm where $\hat{\boldsymbol{z}}$ is initialized using the result of the amortized inference via the encoder of the VAE.

4.3 BP based inference (VAE-BP-GMM)

In this second algorithm, we instead use BP as the iterative inference mechanism used to optimize $\hat{\boldsymbol{z}}$. BP minimizes the following loss function:

$$\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{z}}) = \|\boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\boldsymbol{z}}) - \boldsymbol{x}\|^2 - \gamma \log p(\hat{\boldsymbol{z}}; \boldsymbol{M}) \tag{10}$$

where γ is an hyperparameter weighting the influence of bottom-up and top-down mechanisms, and $\boldsymbol{f}_{\boldsymbol{\theta}}$ denotes the decoder of the VAE. This gradient descent is parameterized by a learning rate λ. Once again, we can initialize the gradient descent using the estimate obtained with the encoder of the VAE. Derivations of this model are provided in appendix C, where the forward pass through the model is given in algorithm 2.

Investigating the relationship between PC and BP based inference, we have found that PC networks needed a larger number of iterations to convey information from the reconstruction error, and that this number of iterations was exponential with regard to the depth of the decoder. For this reason, experiments with the VAE-PC-GMM model were prohibitively slow, and we only conducted experiments with the remaining models.

4.4 Restricting the iterative inference to the memory component (VAE-GMM)

Backpropagating the reconstruction error might not be necessary when the estimate provided by the VAE encoder already conveys enough information from the observed input $\boldsymbol{x}$. As such, we also experiment with a simpler version of the previous model where iterative inference of $\hat{\boldsymbol{z}}$ only considers the top-down update rule coming from the memory. In this simpler version, the update rule for $\hat{\boldsymbol{z}}$ provided in algorithm 1 becomes (proof in appendix C):

$$\hat{\boldsymbol{z}} \leftarrow \text{softmax}\Big(- \frac{\|\hat{\boldsymbol{z}} - \boldsymbol{M}\|_2^2}{2\sigma^2}\Big) \cdot \boldsymbol{M}^\intercal \tag{11}$$

This update mechanism is computationally lighter that the BP-based inference model, and has the advantage of being differentiable.

4.5 Training the precision matrix (VAE-GMM*)

For this last model, we suggest optimizing the precision coefficients $\boldsymbol{P} = \boldsymbol{\Sigma}^{-1}$ of the GMM. These coefficients condition the shape of the Gaussian mixtures. Some memory retrieval tasks might need to ignore partial information from the observed input $\boldsymbol{x}$ and as such could benefit from such an adaptation. In our experiments, we design such tasks on the CLEVR dataset, where the objective is to retrieve scenes using as input a shifted image of the same scene, or an image of the same scene where the colors of the objects have been modified. To properly accomplish these two tasks, the AM model needs to learn to give less importance to position information in the first case, and to color information in the second case.

The model starts from the estimate $\hat{\boldsymbol{z}}$ inferred by the VAE, and uses the following update rule:

$$\hat{\boldsymbol{z}} \leftarrow \text{softmax}\Big(- \frac{1}{2}(\hat{\boldsymbol{z}} - \boldsymbol{M})^\intercal \cdot \boldsymbol{P} \cdot (\hat{\boldsymbol{z}} - \boldsymbol{M})\Big) \cdot \boldsymbol{M}^\intercal \tag{12}$$

During training, $\boldsymbol{P}$ is optimized using BP in order to reduce the mean squared error between the inferred representation $\hat{\boldsymbol{z}}$ and the correct memory pattern $\boldsymbol{M}^*_k$. During evaluation, the values of $\boldsymbol{P}$ are fixed.

5 Experiments

In this section, we present the experiments performed to evaluate the proposed models. In all experiments, we measure performance using the percentage of properly retrieved patterns from associative memories containing $N = 100$ patterns. We consider that a pattern is properly retrieved if the distance between the inferred representation $\hat{\boldsymbol{z}}$ and the correct memory pattern $\boldsymbol{M}_{k*}$ is lower than a threshold value chosen manually. All the implementation details (benchmark models, hyperparameters, training, dataset splits) are provided in appendix D.

5.1 Datasets

We evaluate the proposed models on two image datasets: CIFAR10 and CLEVR. CIFAR10 [19] (MIT License) consists of 32×32 RGB images of 10 classes. We pretrain a VAE on the training set using a convolutional architecture, with a latent space of dimension $d = 16$.

The CLEVR dataset [16] (CC BY 4.0 License) consists of 64×64 RGB images of 3D scenes composed of simple 3D objects. We use a pretrained MONet [5] model as the VAE, with a latent space of dimension $d = 4 \times 16$. The MONet encoder infers a set representation given an input scene image. In our experiments, we flatten this set representation, which can lead to wrong measures of similarity between two representations. Indeed, a permuted set representation still encodes the same information, but once flattened the permuted and original representations might have a low similarity score. To limit this issue, we have capped the number of objects present in the scene images to 3, and capped the size of the set representation to 4. Examples of images from the CLEVR dataset along with possible transformations are presented in figure 2.

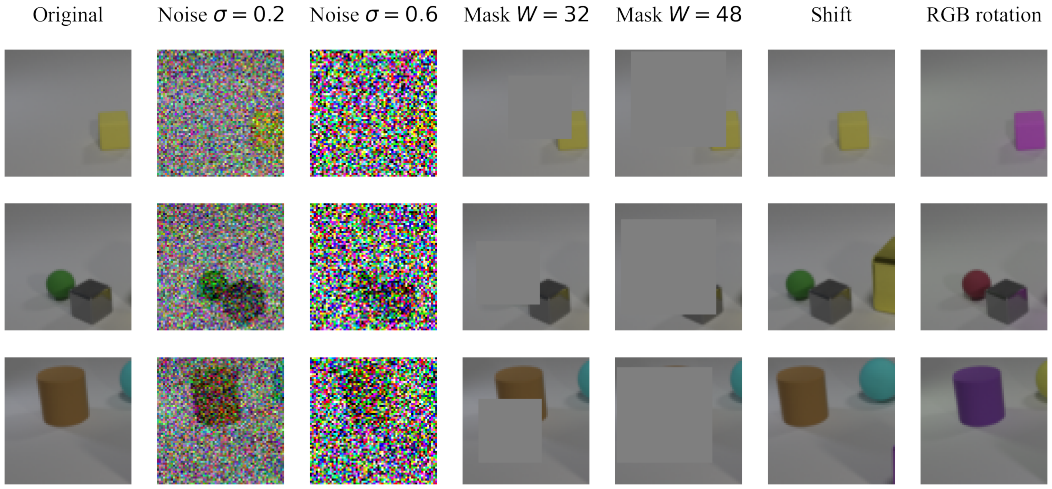

Figure 2: Examples of original, corrupted, and transformed images from the CLEVR dataset.

5.2 Benchmark models

We compare our AM models with Modern Continuous Hopfield Networks (MCHN), Neural Turing Machines (NTM) and End-to-End Memory Networks (MemN2N). For fair comparison, in all benchmark models, the memory is initialized with the stored patterns (we do not use the writing mechanisms of NTMs). We also experiment with these models in the representation space, where the memory is initialized with representations of the stored images, and the networks receive as input the representation of the corrupted image. We indicate this modification with the VAE prefix.

The NTM and MemN2N models are trained using BP using as loss function the mean squared error between the predicted output $\hat{\boldsymbol{x}}$ (respectively $\hat{\boldsymbol{z}}$ when using the VAE) and the correct memory pattern $\boldsymbol{M}_k^*$. For a fair comparison, this training is performed with clean inputs $\boldsymbol{x} = \boldsymbol{M}_k^*$, to avoid giving any prior information about the type of transformation the inputs might be corrupted with during evaluation. Finally, we also experiment with a version of the MCHN using a balanced GMM as

starting prior distribution, that we denote GMM in our experiments. Equivalently, this is the version of our algorithm without a representation component.

5.3 Corrupted inputs

First, we evaluate whether the associative memory models are able to properly retrieve stored patterns when given corrupted versions of these patterns. We apply different level of noise to the images ($\sigma \in \{0.2, 0.4, 0.6, \cdots, 2.0\}$), as well as different sizes of masks (16 and 24 pixels wide on the CIFAR10 dataset, and 32 and 48 pixels wide on the CLEVR dataset). The results of these experiments are reported in table 2 and 3. More figures and examples are provided in appendix E.

Table 2: Memory retrieval with corrupted inputs on the CIFAR10 dataset. Mean percentages of successful retrieval and standard deviations are reported.

Input	Clean input	Noise σ=0.2	Noise σ=0.6	Mask W=16	Mask W=24
GMM	$\mathbf{100 \pm 0.0}$	$\mathbf{100 \pm 0.0}$	94.3 ± 6.6	$\mathbf{99.7 \pm 1.3}$	48.7 ± 12.7
Hopfield	3.0 ± 3.9	3.9 ± 4.0	1.5 ± 2.4	2.3 ± 3.6	1.6 ± 2.6
NTM	99.3 ± 2.4	98.9 ± 2.7	91.6 ± 15.0	94.7 ± 8.0	$\mathbf{62.7 \pm 20.9}$
MemN2N	81.1 ± 11.3	77.9 ± 11.1	65.8 ± 13.4	63.2 ± 13.9	32.3 ± 11.5
VAE-BP-GMM	$\mathbf{100 \pm 0.0}$	$\mathbf{100 \pm 0.0}$	81.1 ± 8.9	98.1 ± 3.3	39.5 ± 10.7
VAE-GMM	$\mathbf{100 \pm 0.0}$	$\mathbf{100 \pm 0.0}$	86.3 ± 9.1	98.3 ± 2.8	39.5 ± 11.8
VAE-Hopfield	92.5 ± 6.2	91.7 ± 6.0	79.6 ± 10.3	82.5 ± 8.1	52.5 ± 10.1
VAE-NTM	$\mathbf{100 \pm 0.0}$	$\mathbf{100 \pm 0.0}$	$\mathbf{98.6 \pm 2.3}$	99.5 ± 1.5	58.7 ± 10.2
VAE-MemN2N	95.0 ± 4.7	92.6 ± 6.1	82.2 ± 8.7	82.2 ± 8.0	44.4 ± 11.7

Table 3: Memory retrieval with corrupted inputs on the CLEVR dataset. Mean percentages of successful retrieval and standard deviations are reported.

Input	Clean input	Noise σ=0.2	Noise σ=0.6	Mask W=32	Mask W=48
GMM	$\mathbf{100 \pm 0.0}$	$\mathbf{100 \pm 0.0}$	$\mathbf{73.0 \pm 10.1}$	69.3 ± 8.6	30.3 ± 10.9
Hopfield	1.3 ± 2.3	1.3 ± 2.8	1.0 ± 2.5	0.8 ± 1.8	1.5 ± 2.4
NTM	82.1 ± 13.1	84.5 ± 11.9	47.8 ± 16.8	56.4 ± 13.2	17.6 ± 9.7
MemN2N	17 ± 11.3	16 ± 12.3	14.5 ± 10.5	9 ± 11.2	5.5 ± 6.0
VAE-BP-GMM	$\mathbf{100 \pm 0.0}$	49.5 ± 16.0	2.0 ± 2.5	69.2 ± 10.2	36.5 ± 5.8
VAE-GMM	$\mathbf{100 \pm 0.0}$	46.3 ± 11.8	4.4 ± 4.4	68.9 ± 9.7	31.5 ± 11.6
VAE-Hopfield	94.2 ± 4.9	34.5 ± 11.4	2.7 ± 4.0	64.4 ± 11.5	32.7 ± 9.9
VAE-NTM	99.9 ± 0.7	40.8 ± 10.0	3.4 ± 4.3	$\mathbf{75.0 \pm 10.2}$	$\mathbf{38.3 \pm 11.2}$
VAE-MemN2N	97.9 ± 3.1	31.7 ± 11.4	2.4 ± 3.1	66.1 ± 10.4	34.2 ± 10.1

We can observe that the two models using dot-product attention, MCHN (Hopfield in the tables) and MemN2N, perform poorly compared to other methods, not even reaching 100% of correctly retrieved patterns when presented with clean inputs. The NTM model, using cosine similarity based attention, works better and sometimes outperforms the proposed models. Among the proposed models, we can see that the VAE-BP-GMM does not perform significantly better than the simpler VAE-GMM model.

An important observation from these results is that in most scenarios, the best performing methods do not use a representation component. In particular, the MONet model seems very sensitive to noise. However, our intuition is that memory retrieval on the representation level should be more powerful if the transformation applied on the observed inputs $\boldsymbol{x}$ has a limited effect on the representations.

5.4 Scene transformations

To verify this hypothesis, we perform two additional experiments on the CLEVR dataset. If we assume that the representations provided by MONet encode object positions, shapes and colors, then an AM model working on the representation level could perform well when we apply transformations on these specific object features. We propose two such scenarios. In the first scenario, we perform an RGB rotation on the images, which results in identical scenes with the exception of the object colors.

In the second scenario, we take a shifted crop of the original CLEVR image, which can simulate a shifted point of view of the same scene. Examples images are displayed in figure 2.

We measure the success of memory retrieval with these transformed inputs using our models as well the NTM and MemN2N models. We also experiment with the VAE-GMM* model performing precision coefficient training. We expect the model to learn that some information conveyed by the encoder (for instance color in the first scenario) is irrelevant for memory retrieval. For fair comparison, we also experiment with the VAE-NTM and VAE-MemN2N models trained in the two scenarios, that we denote with an asterisk. The results are reported in table 4.

Table 4: Memory retrieval with transformed inputs on the CLEVR dataset. Mean percentages of successful retrieval and standard deviations are reported.

Input	Color rotation	Shift
GMM	76 ± 12.4	28 ± 8.5
NTM	48.8 ± 16.9	0.7 ± 1.9
MemN2N	12.6 ± 11.5	6.4 ± 8.1
VAE-GMM	53.6 ± 12.0	63.1 ± 10.2
VAE-NTM	54.6 ± 10.7	61.5 ± 11.6
VAE-MemN2N	41.2 ± 11.0	56.2 ± 12.1
VAE-GMM*	$\mathbf{98.9 \pm 2.5}$	93.7 ± 5.3
VAE-NTM*	98.2 ± 3.2	$\mathbf{96.3 \pm 4.4}$
VAE-MemN2N*	91.6 ± 6.9	84.2 ± 7.8

Using a representation component improves the performance of the models in these scenarios. Additionally, training on the retrieval task where the transformations are applied provides almost perfect retrieval scores for the VAE-GMM and VAE-NTM models.

6 Discussion

We have proposed a formulation of auto-associative memories based on the FEP and the variational inference framework, where the stored patterns condition the prior probability on the representation $\boldsymbol{z}$. This framework has allowed us to draw a connection between MCHNs and the PC theory, as well as to design several AM models with high retrieval accuracy and robustness.

Combining representation and memory allows to retrieve patterns on the representation level, which decreases the size of the memory store, and performs retrieval based on a similarity measured in the representation space, where we can expect more meaningful features to appear. For instance, we have shown that this allows to recognize visual scenes from a shifted point of view, while pixel-level AM models failed most of the time. Our results also seem to demonstrate that distance based attention as used in our models (see equation 11) outperforms dot-product based attention as used in MCHNs, although this performance gap could be mitigated with normalization techniques, as hinted by the results obtained with NTMs, that use cosine similarity instead of dot-product.

The proposed PC and BP based models take into account information from the query $\boldsymbol{x}$ at each inference iteration, by directly minimizing an energy function that depends on $\boldsymbol{x}$ (eq. 4 and eq. 10). In contrast, the Kanerva Machine [39, 40] or the VAE-GMM and VAE-GMM* models we proposed only use this information to output an initial estimate $\hat{\boldsymbol{z}}$ that is later optimized in order to minimize an energy function only depending on the memory. According to our experiments, this feature of the PC and BP based models does not seem useful for memory retrieval. However, in other applicative settings, these models could benefit from their ability to weight query pattern information and stored patterns information, performing a form of memory-aided iterative perceptual inference.

The proposed formulation is currently limited to content-based reading mechanisms. In contrast, models such as NTMs offer a larger variety of addressing schemes as well as writing mechanisms that allow to only store relevant information in the memory. Another limitation of this work is that we did not investigate the memory capacity and convergence properties of the proposed models. Future work should focus on these analyses, and possible improvements of the PC-based inference, for instance using precision weighting, or the "learning to optimize" approach of [23].

References

[1] Y. Bengio, B. Scellier, O. Bilaniuk, J. Sacramento, and W. Senn. Feedforward initialization for fast inference of deep generative networks is biologically plausible. *arXiv preprint arXiv:1606.01651*, 2016.

[2] J. Bornschein, A. Mnih, D. Zoran, and D. Jimenez Rezende. Variational memory addressing in generative models. *Advances in Neural Information Processing Systems*, 30, 2017.

[3] T. Bricken and C. Pehlevan. Attention approximates sparse distributed memory. *Advances in Neural Information Processing Systems*, 34:15301–15315, 2021.

[4] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth. The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, 81: 55 – 79, 2017. ISSN 0022-2496. doi: https://doi.org/10.1016/j.jmp.2017.09.004. URL `http://www.sciencedirect.com/science/article/pii/S0022249617300962`.

[5] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[6] A. Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3):181–204, 2013. doi: 10.1017/S0140525X12000477.

[7] M. Demircigil, J. Heusel, M. Löwe, S. Upgang, and F. Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168(2):288–299, 2017.

[8] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

[9] K. Friston. The free-energy principle: a rough guide to the brain? *Trends in Cognitive Sciences*, 13(7):293–301, 2009. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2009.04.005. URL `https://www.sciencedirect.com/science/article/pii/S136466130900117X`.

[10] K. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11 (2):127–138, 2 2010. ISSN 1471-0048. doi: 10.1038/nrn2787. URL `https://doi.org/10.1038/nrn2787`.

[11] K. Friston and S. Kiebel. Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 364:1211–21, 2009.

[12] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[13] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[14] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. ISSN 0027-8424. doi: 10.1073/pnas.79.8.2554. URL `https://www.pnas.org/content/79/8/2554`.

[15] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

[16] J. Johnson, B. Hariharan, L. Van Der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[17] P. Kanerva. *Sparse distributed memory*. MIT press, 1988.

[18] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[19] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[20] D. Krotov and J. Hopfield. Dense associative memory is robust to adversarial inputs. *Neural computation*, 30(12):3151–3167, 2018.

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

[22] H. Le, T. Tran, T. Nguyen, and S. Venkatesh. Variational memory encoder-decoder. *Advances in neural information processing systems*, 31, 2018.

[23] J. Marino, Y. Yue, and S. Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pages 3403–3412. PMLR, 2018.

[24] B. Millidge, A. Tschantz, and C. L. Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *CoRR*, abs/2006.04182, 2020. URL `https://arxiv.org/abs/2006.04182`.

[25] B. Millidge, T. Salvatori, Y. Song, T. Lukasiewicz, and R. Bogacz. Universal hopfield networks: A general framework for single-shot associative memory models. *arXiv preprint arXiv:2202.04557*, 2022.

[26] S. Otte, T. Schmitt, K. Friston, and M. V. Butz. Inferring adaptive goal-directed behavior within recurrent neural networks. In *International Conference on Artificial Neural Networks*, pages 227–235. Springer, 2017.

[27] A. Radhakrishnan, M. Belkin, and C. Uhler. Overparameterized neural networks implement associative memory. *Proceedings of the National Academy of Sciences*, 117(44):27162–27170, 2020.

[28] J. Ramapuram, Y. Wu, and A. Kalousis. Kanerva++: Extending the kanerva machine with differentiable, locally block allocated latent memory. In *International Conference on Learning Representations*, 2020.

[29] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

[30] R. Rao and D. Ballard. Predictive coding in the visual cortex a functional interpretation of some extra-classical receptive-field effects. *Nat Neurosci*, 2:79–87, 1999.

[31] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014.

[32] T. Salvatori, Y. Song, Y. Hong, L. Sha, S. Frieder, Z. Xu, R. Bogacz, and T. Lukasiewicz. Associative memories via predictive coding. *Advances in Neural Information Processing Systems*, 34, 2021.

[33] S. Sukhbaatar, a. szlam, J. Weston, and R. Fergus. End-to-end memory networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[34] A. Tschantz, B. Millidge, A. K. Seth, and C. L. Buckley. Hybrid predictive coding: Inferring, fast and slow. *arXiv preprint arXiv:2204.02169*, 2022.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] H. Von Helmholtz. *Handbuch der physiologischen Optik: mit 213 in den Text eingedruckten Holzschnitten und 11 Tafeln*, volume 9. Voss, 1867.

[37] J. Weston, S. Chopra, and A. Bordes. Memory networks. *CoRR*, abs/1410.3916, 2015.

[38] J. C. R. Whittington and R. Bogacz. An Approximation of the Error Backpropagation Algorithm in a Predictive Coding Network with Local Hebbian Synaptic Plasticity. *Neural Computation*, 29(5):1229–1262, 05 2017. ISSN 0899-7667. doi: 10.1162/NECO_a_00949. URL https://doi.org/10.1162/NECO_a_00949.

[39] Y. Wu, G. Wayne, A. Graves, and T. Lillicrap. The kanerva machine: A generative distributed memory. In *International Conference on Learning Representations*, 2018.

[40] Y. Wu, G. Wayne, K. Gregor, and T. Lillicrap. Learning attractor dynamics for generative memory. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9401–9410, 2018.

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] We believe that the three claimed contributions in Introduction are properly reflected in sections 3 and 4.
   (b) Did you describe the limitations of your work? [Yes] See the last paragraph of the Discussion.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We did not identify any direct potential negative societal impact. If an indirect potential impact is brought to our attention, we would be happy to discuss it.
   (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? [Yes] The assumptions and definitions used in our derivations are stated (see sections 3.2 and 4, and appendices A,B and C).
   (b) Did you include complete proofs of all theoretical results? [Yes] We provide detailed derivations of our models and theoretical results in appendices A, B and C.
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] See appendix D.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See tables 2, 3 and 4, as well as figures in appendix E.
   (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)? [Yes] See appendix D.
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] Datasets were credited in section 5, code assets are credited in appendix D.
   (b) Did you mention the license of the assets? [Yes] Licences for the datasets were mentionned in section 5, the license for used code assets is mentionned in D.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Implementations of our models are linked in appendix D.
   (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? [No] The data we used are commonly used ML datasets. CLEVR is an automatically generated dataset and to our knowledge, the CIFAR10 dataset does not contain any personally identifiable of offensive content.
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]