Associative Memories via Predictive Coding

Anonymous Author(s)
Affiliation
Address
email

Abstract

Associative memories in the brain receive and store patterns of activity registered by the sensory neurons, and are able to retrieve them when necessary. Due to their importance in human intelligence, computational models of associative memories have been developed for several decades now. They include autoassociative memories, which allow for storing data points and retrieving a stored data point s when provided with a noisy or partial variant of s. Such associative memories also play an important role in deep learning, as it has been shown that overparametrized neural networks trained with backpropagation are able to memorize data points. In this paper, we present a novel neural model for realizing (auto)associative memories, based on a hierarchical generative model that receives external stimuli via sensory neurons. This model is trained using predictive coding, an error-based learning algorithm inspired by learning in the cortex. To test the capabilities of this model, we perform multiple retrieval experiments from both corrupted and partial data points. In an extensive comparison, we show that this new model outperforms (in storage capacity and in retrieval accuracy and robustness) state-of-the-art associative memories (including especially autoencoders trained via backpropagation) by a large margin. In particular, in completing partial data points, our model achieves remarkable results on natural image datasets, such as ImageNet, with a surprisingly high accuracy, even when presented with only a tiny fraction of pixels of the original images.

1 Introduction

Building models that are able to store and retrieve information has been an important direction of research in artificial intelligence (AI). Particularly, such models include (auto)associative memories (AMs), which allow for the storage of data points and their contents-based retrieval, i.e., for retrieving a stored data point s from a corrupted or a partial variant of s. One way to realize AMs is to store data points as attractors, so that they can be easily recovered when presenting their corrupted variants [1, 2]. Classic AMs include Hopfield networks, which are energy-based models that memorize and retrieve data points by minimizing a suitable energy function. Recent works have shown that overparametrized autoencoders (AEs) are excellent AMs as well. Particularly, when training an AE to generate a specific point s when s itself is presented as an input, it gets stored as an attractor. This has been shown formally and empirically for heavily overparametrized models [3].

At first sight, AMs seem to be opposite to standard neural networks performing machine learning tasks. In AMs, the more training points they are able to memorize, the better, and feature learning is meaningless, as there is no test set that they have to generalize to. Whereas in machine learning, the ability to memorize a training set has often been considered an obstacle for generalizing to a test set: the more is memorized by the model, the less has been learned. However, memorization is a key ingredient of learning and understanding the world. Hence, recent works on neural networks look at *memory* and *learning* as being two faces of the same coin, instead of being one opposite to the other [4, 5]. In particular, it has in fact been shown that overparametrized neural networks are good

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

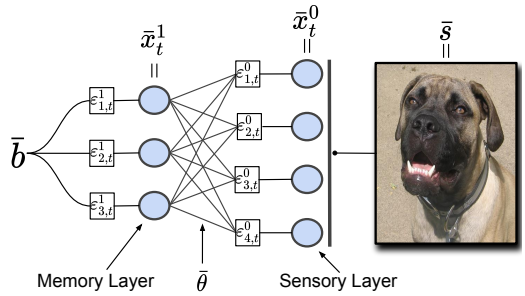
associative memory models, as they are able to perfectly store and retrieve training data points [3]. Furthermore, training them on the full dataset by gradient descent, they are approximately equivalent to kernel machines [4], which memorize data points and use memorized information for predictions. Roughly speaking, standard neural networks can intuitively be considered as AMs that store data points and/or more abstract patterns (i.e., combinations of feature values) of data points.

This interpretation of standard neural networks as abstract AMs is very much aligned with the roles and importance that AMs play in neuroscience at the lower levels of human intelligence. Through our life, we learn a huge number of associations between concepts: the taste of a particular food, the meaning of a gesture, or to stop when we see a red light. Every time we acquire new information of this kind, it gets stored in our long-term memory, situated in distributed networks of brain areas [6]. In particular, visual memories are stored in a hierarchical network of visual and associative areas [7]. These regions learn progressively more abstract representation of visual stimuli, so they participate in both perception and memory as each area memorizes relationships present in their inputs [8]. Accordingly, early visual areas learn common regularities present in the stimuli [9], while at the top of this hierarchy, associative areas (such as hippocampus and entorhinal and perirhinal cortices) store the relationships between extracted features, which encode an entire stimulus or episode [10]. The memory system of the brain is able to both recall complex memories [6, 11], and use them to generate predictions to guide behavior [12]. Learning in these associative memories shapes our understanding of the world around us and builds the foundations of human intelligence.

In this work, bridging recent trends in deep learning with recent discoveries in neuroscience, we present a novel AM model that is based on a new energy-based hierarchical generative approach. This AM model differs from standard Hopfield networks, as it is trained using predictive coding (PC), which is a biologically plausible learning algorithm that replicates learning in the visual cortex [9]. PC describes learning in hierarchical networks (unlike the classic Hopfield model that includes only a single layer of neurons). The idea that PC may naturally be related to AMs is inspired by recent works showing that the generative neural architecture that connects the hippocampus to the neocortex is based on an error-driven learning algorithm, which can be interpreted with a PC framework [11, 13]. From a machine learning perspective, the original formulation of PC is able to perform both supervised and unsupervised tasks with a high accuracy [9, 14]. Particularly, with the addition of some constraints, it has been shown to be completely equivalent to backpropagation [15–17].

We show that the new AM model is not only interesting from a neuroscience perspective, but it also considerably outperforms state-of-the-art AM models (including especially autoencoders) when it comes to retrieve stored data points. Our results can be briefly summarized as follows:

- We first introduce a new family of PC networks that are able to perform generative tasks. We then present our new AM models based on these generative PC networks. In particular, we show empirically that they store training data points as attractors of their dynamics by demonstrating that they can reconstruct data points from corrupted ones. Particularly, they are able to store and retrieve visual memories such as high-resolution natural images.
- In an extensive comparison of the new AM model against standard autoencoders as state-of-the-art AMs, the new AMs considerably outperform autoencoders (in storage capacity and in retrieval accuracy and robustness) when tested on neural networks of the same size. Particularly, the new AMs are able to retrieve images from CIFAR10 and the Tiny ImageNet dataset even when having only one layer of 256 hidden neurons, while autoencoders need two layers of over 2000 neurons to achieve similar (but still worse) results.
- The reconstruction of incomplete data points is a challenging task in the field of AMs. It has been shown that some models are able to achieve it. Particularly, a mixed model formed by dense associative Hopfield networks [18] together with a restricted Boltzmann machine [19] is able to reconstruct entire MNIST digits even when presented with only $1/3$ of the original memory. We show that our model naturally solves the task of reconstructing complex and colored images with a surprisingly high accuracy. Particularly, we test this on ImageNet, perfectly reconstructing them even after removing all but $1/8$ of the original image.
- Further experimental results explore the limits of our model, in terms of both capacity and reconstruction capabilities from tiny fractions of the original pixels. We then conclude by showing that to increase the overall capacity of the model, it suffices to add additional layers. We test multi-layer networks against overparametrized autoencoders, and show that we also significally outperform the best-performing model reported in the original work [3].



Algorithm Learning to generate $\bar{s}$ with PC

Require: $\bar{x}_0$ is fixed to $\bar{s}$.
1: **for** $t = 1$ to T **do**
2: **for** each neuron i in each level l **do**
3: Update $x^l_{i,t}$ to minimize E_t.
4: **end for**
5: **end for**
6: Update each $\theta^{l+1}_{i,j}$ and b_i to minimize E_t

Figure 1: Left: Generative PCN with 2 layers. Each line linking neurons denotes a pair of connections (excitatory in left direction, and inhibitory in right direction). At $t = 0$, the value nodes of the output layer are fixed to the vector representing the pixels of the figure. Then, during the learning phase, both weight parameters and value nodes are updated. Right: Pseudocode of the described algorithm.

2 Generative Predictive Coding Networks

We now briefly recall predictive coding networks (PCNs), and we introduce generative PCNs, which are the underlying neural model for the novel AMs introduced in the subsequent section.

Deep neural networks have a multi-layer structure, where each layer is formed by a vector of neurons [20]. While in standard deep learning the goal is to minimize the error on a specific layer, PC defines an error in every layer of the network, minimized by gradient descent on a global energy function [9]. Particularly, let M be a PCN with $L-1$ fully connected layers of dimension n, followed by a fully connected layer of dimension d. We call the d-dimensional layer *sensory* layer (indexed as layer 0), which biologically corresponds to sensory neurons (see left panel of Fig 1). We call the most internal layer (layer L) *memory*, which is equipped with an n-dimensional memory vector b. Every layer l contains value nodes $x^l_{i,t}$, and every pair of consecutive layers is connected via weight matrices $\bar{\theta}^l$, which represent the synaptic weights between neurons of different layers. The value nodes, the weight matrices, and the memory vector are all trainable parameters of the model. The signal passed from layer $l+1$ to layer l, called prediction $\bar{\mu}^l_t$, is computed as follows:

$$\mu^l_{i,t} = \begin{cases} \sum_{j=1}^{n^{l+1}} \theta^{l+1}_{i,j} f(x^{l+1}_{j,t}) & \text{if } 0 \leq l < L \\ b & \text{if } l = L\,, \end{cases} \tag{1}$$

where f is a non-linear activation function. To conclude, the difference between the value $\bar{x}^l_t$ and their predictions $\bar{\mu}^l_t$ is the *error* $\varepsilon^l_{i,t} = x^l_{i,t} - \mu^l_{i,t}$. We now describe how PCNs are trained. To do this, we explain one iteration of a training algorithm, called *inference learning* (*IL*), that is divided into an inference phase and a weight update phase.

Inference: Only the value nodes of the network are updated, while both the weight parameters and the memory vector are fixed. Particularly, the value nodes are modified via gradient descent to minimize the global error of the network, expressed by the following energy function E_t:

$$E_t = \tfrac{1}{2} \sum_{i,l} (\varepsilon^l_{i,t})^2\,. \tag{2}$$

Assume that we train a generative PCN on a training point $\bar{s} \in \mathbb{R}^d$. To do this, the value nodes of the sensory layer are fixed to the training point $\bar{s}$, and are never updated. Thus, the error on every neuron of the sensory layer is equal to $\varepsilon^0_{i,t} = s_i - \mu^0_{i,t}$. The process of minimizing E_t by modifying all $x^l_{i,t}$ leads to the following changes in the value nodes:

$$\Delta x^l_{i,t} = \begin{cases} \gamma \cdot (-\varepsilon^l_{i,t} + f'(x^l_{i,t}) \sum_{k=1}^{n^{l-1}} \varepsilon^{l-1}_{k,t} \theta^l_{k,i,t}) & \text{if } 0 < l \leq L \\ 0 & \text{if } l = 0\,, \end{cases} \tag{3}$$

where γ is a constant determining by how much the activity changes in each step. The computations in Eqs. (1) and (3) can be realized in a network including value nodes $x^l_{i,t}$ and error nodes $\varepsilon^l_{i,t}$ [9], as shown in Fig 1. The inference phase works as follows: starting from a given configuration of the value nodes $\bar{x}_0$, inference continuously upates the value nodes according to Eq. (3) until it has converged. We call the configuration of the value nodes at convergence $\bar{x}_T$, where T is the number of steps needed to reach convergence (in practice, it is a fixed large number).

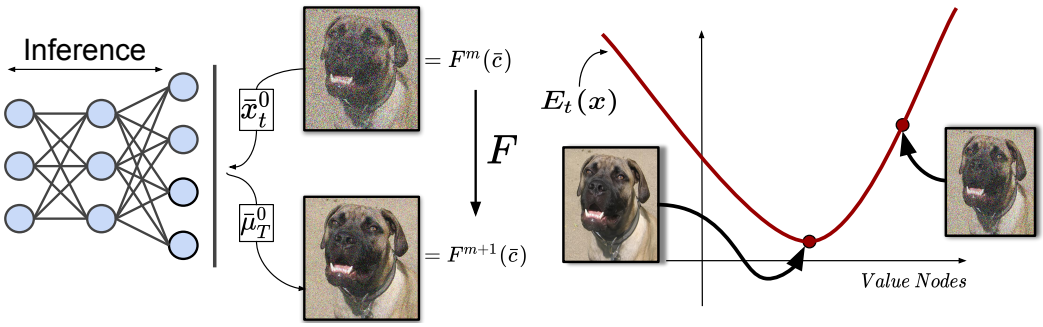


Figure 2: Left: Representation of the function F, used to retrieve stored images. It can be decomposed into three steps: (1) The value nodes of the sensory layer are fixed to the pixels of the corrupted image $F^m(\bar{c})$. (2) Inference runs for T operations (until convergence). (3) We set $F^{m+1}(\bar{c})$ to the prediction of the sensory layer $\bar{\mu}_T^0$. Note that the weight parameters are never updated during the above steps. We have omitted the error nodes for simplicity. Right: Representation of an AM, where an image $\bar{s}$ (photo of a dog) is stored as an attractor of the dynamics. A corrupted image that lies in a specific neighborhood of $\bar{s}$ converges to it when minimizing the total energy via running inference.

Weight Update: When the value nodes of the sensory layer are fixed to an input signal $\bar{s}$, inference may not be sufficient to reduce the total energy to zero. Hence, to further decrease the total error, a single *weight update* is performed: both the weight matrices and the memory vector are updated by gradient descent to minimize the same objective function E_t, and behave according to the following equations, where α is the learning rate. Particularly, the derived update rule is the following:

$$\Delta\theta_{i,j}^{l+1} = -\alpha \cdot \partial E_T / \partial \theta_{i,j}^{l+1} = \alpha \cdot \varepsilon_{i,T}^{l} f(x_{j,T}^{l+1}), \tag{4}$$

$$\Delta b_i = -\alpha \cdot \partial E_T / \partial b_i = -\alpha \varepsilon_{i,T}^{L}. \tag{5}$$

The phases of inference and weight update are iterated until the total energy E_t reaches a minimum. This learning algorithm learns a dataset by using only local computations, which minimize the same energy function. Fig. 1 gives a graphical representation and the pseudocode for generative PCNs. Detailed derivations of Eqs. (3) and (5) are in the supplementary material (and in [14]).

3 Predictive Coding for Associative Memories

So far, we have shown how PCNs can perform generative tasks. We now show how generative PCNs can be used as associative memories, i.e., how the model stores the data points that it is trained on, and how these data points can be retrieved when presenting corrupted versions to the network, returning the most similar stored data point. Let $\bar{s}$ be a training data point, and M be the PCN considered above, already trained until convergence to generate $\bar{s}$. Moreover, assume that M makes the total energy converge to *zero* at iteration T. At this point, the energy function defined on the value nodes has a local minimum $\bar{x}$ in which the value nodes of the sensory layer are equal to the entries of $\bar{s}$. Note that $\bar{x}$ is actually an attractor of the dynamics of E_t: when given a configuration that is not a local minimum, inference will update the value nodes until the total energy reaches a minimum. If this configuration lies in a specific neighborhood of $\bar{x}$, inference will converge to $\bar{x}$. So, given a dataset, we obtain an AM of the dataset if all the training points are stored in the above way as attractors.

The above can be used to retrieve stored data points $\bar{s}$: given a corrupted version $\bar{c} \in \mathbb{R}^d$ of $\bar{s}$, one can retrieve $\bar{s}$ as follows. First, we set the value nodes of the sensory layer to the corrupted points, i.e., $\bar{x}_t^0 = \bar{c}$ for the whole process. Then, we run inference until convergence and save the prediction $\bar{\mu}_T^0$ of the sensory layer. If the original data point was stored as an attractor, we expect the prediction $\bar{\mu}_T^0$ to be a less corrupted version of it. Let $F\colon \mathbb{R}^d \to \mathbb{R}^d$ be the function that sends $\bar{c}$ to $\bar{\mu}_T^0$ just described, and summarized in Fig. 2. Many iterations of this function allow to retrieve the stored data point. Hence, summarizing the above, training points are stored in the memory vector $\bar{b}_T$ and the weight parameters, and what the algorithm does to retrieve them is simply the inference phase of PCNs. Since visual memories are stored in hierarchical networks of brain areas, PC could be a highly plausible algorithm to better understand how memory and prediction work in the brain.

To experimentally show that generative PCNs are AMs, we trained a 2-layer network with ReLU non-linearity on a subset of 100 images of MNIST, Tiny ImageNet, and CIFAR10. After training, we

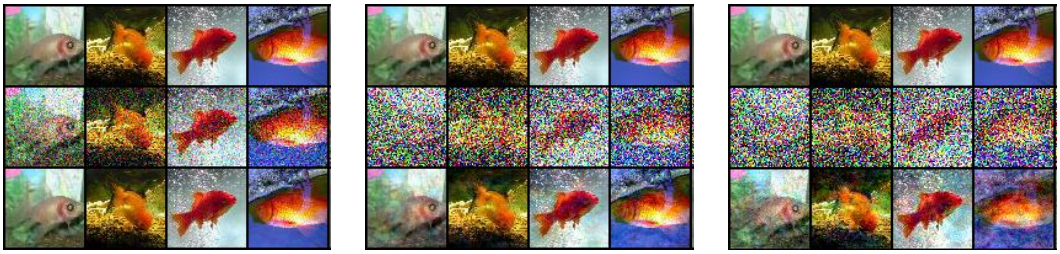


Figure 3: Reconstructions of 64×64 Tiny ImageNet images with noise levels 0.2, 0.7, and 0.9, respectively (original, noisy, and reconstructed image in first, second, and third row, respectively).

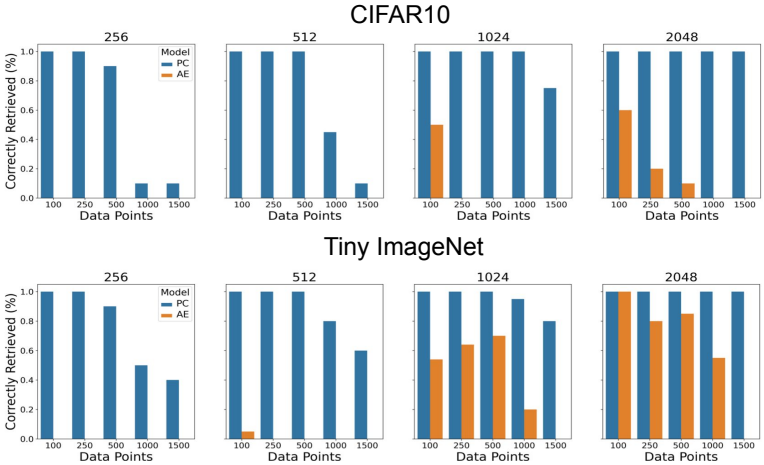


Figure 4: Percentage of correctly retrieved images by 2-layer generative PCNs (PC) and 3-layer autoencoders (AE) with hidden-layer dimensions of 256, 512, 1024, and 2048, when presented with a corrupted image with Gaussian noise of variance 0.2, trained on CIFAR10 and Tiny ImageNet.

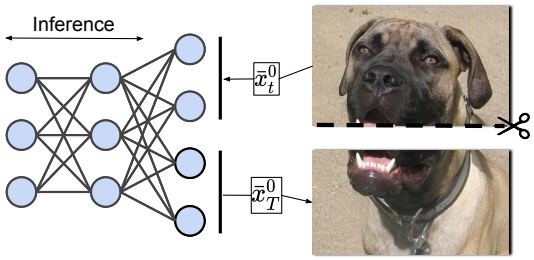
presented the model with a corrupted variant (by adding Gaussian noise) of the training set. We then used the PCNs to reconstruct the original images from the corrupted ones. The experimental results confirm that the model is able to retrieve the original image, given a corrupted one. The obtained reconstructions are shown in Fig. 3. We now provide a more comprehensive analysis, which studies the capacity of generative PCNs when changing the number of data points and parameters.

Experiments: We trained 2-layer PCNs with ReLU non-linearity and hidden dimension $n \in \{256, 512, 1024, 2048\}$ on subsets of CIFAR10 and Tiny ImageNet of cardinality $N = \{100, 250, 500, 1000, 1500\}$. Every model is trained until convergence, and all the images are retrieved as explained in Section 3. To provide a numerical evaluation, an image is considered recovered when the error between the original image and the recovered image is less than 0.005.

To compare our results against a standard baseline, we also trained 3-layer autoencoders (AEs) with the same hidden dimension on the same task, and compared the results. Note that the number of parameters between a 2-layer PCN and a 3-layer AE with the same hidden dimension is not the same, as the additional layer (input layer, not needed in generative PCNs) almost doubles the number of parameters in some cases. We trained every AE with different activations, and reported the best achieved results. Further details about the experiments and used hyperparameters are given in the supplementary material.

Results: The analysis shows that our model is able to store and retrieve data points even when the network is not overparametrized (see Fig. 4). AEs trained with BP did not perform well: AEs with less than 1024 hidden neurons always failed to restore even a single data point, and the performance of AEs with 2048 hidden neurons were always worse than PCNs with 256 hidden neurons. This shows that overparametrization is essential for AEs to perform AM tasks, and that our proposed method offers a much more network-efficient alternative.

In terms of capacity, we show that 2-layer PCNs with 256 hidden neurons are able to perfectly store datasets of 250 images of both CIFAR10 and Tiny ImageNet, and that networks with 2048 hidden



Algorithm Retrieving $\bar{s}$ given a non corrupted fraction $\bar{s}'$

Require: if s'_i a correct entry of the original memory, then $x^0_{i,t}$ is fixed to s'_i.

```
for t = 1 to T do
    for each neuron i in each level l do
        Update x^l_{i,t} to minimize E_t.
    end for
end for
return x̄^0_T
```

Figure 5: Left: Algorithm used to retrieve a stored data point from a fraction of it: (1) The value nodes of the sensory layer that correspond to the available entries of $\bar{s}$ are fixed to the respective values. (2) Inference runs for T operations (until convergence). (3) We set our prediction to be equal to the value nodes of the sensory layer $\bar{x}^0_T$. The weight parameters are never updated during the above steps. The error nodes are omitted for simplicity. Right: Pseudocode of the described algorithm.

neurons always managed to store and retrieve all the presented datasets. As typical in the AM domain, small models trained on large datasets fail to store data points, as the space of parameters is not large enough to store each data point as an independent attractor, and many attractors in the same small region lead to a chaotic dynamic. Our model is no different: PCNs with 256 and 512 hidden units are able to store almost 500 Tiny ImageNet images when trained on datasets of that size, but fail to store more than 200 when trained on larger ones (1000 and 1500).

4 Retrieval from Partial Data Points

So far, we have shown how the proposed generative model can be used to retrieve stored data points when presented with corrupted variants. We now tackle the different task of retrieving data points when presented with partial ones. Let $\bar{s}$ be the stored data point, and assume that a fraction of pixels of $\bar{s}$ are accessible, and the goal is to retrieve the remaining ones. Let $\bar{s}'$ be the vector of the same dimension of $\bar{s}$, where a fraction of pixels are equal to the ones of the stored data point, and assume that the position of the pixels that are equal to the ones of $\bar{s}$ is known. We now show how to retrieve the complete data point by using the same network M, trained as already shown in Section 3.

Let M be a PCN trained to generate $\bar{s}$. Then, given the partial version of the original data point $\bar{s}'$, it is possible to retrieve the full data point using M as follows: first, we only fix the value nodes of the sensory layer $\bar{x}^0_t$ to the entries of the partial data point $\bar{s}'$ that we know are equal to the ones of the stored data point, leaving the rest free to be updated. Then, we run inference on the whole network until convergence. At this point, we expect the value nodes $\bar{x}^0_T$ to have converged to the entries of the original data point, stored as an attractor. A graphical representation of the above mechanism is described in Fig. 5. To show the capabilities of this network, we have performed multiple experiments on the Tiny ImageNet and ImageNet datasets, and compared against existing models in the literature. We now start by providing visual evidence on the effectiveness of this method.

Experiments: We trained two networks with hidden dimensions of 1024 and 2048, to generate 50 images of the first class of Tiny ImageNet (corresponding to goldfishes), and a network of 8192 hidden neurons to reconstruct 25 pictures taken form ImageNet. Then, we used inference as explained to retrieve the original images. We considered an image to be correctly reconstructed when the error between the original and the retrieved image was smaller than 0.0001. Furthermore, we plotted the partial images together with their reconstructions, for a visual check.

Results: On Tiny ImageNet, a generative PCN with 1024 hidden neurons managed to reconstruct all the images when presented with $1/8$ of the original image, and more than half for the smallest fraction considered, $1/16$. The network with 2048 neurons also failed to reconstruct all the images when presented with $1/16$ of the original image. However, even when presented with a portion as small as $1/16$ of the original image, the reconstruction was clear, although not perfect and hence now below our threshold. This result is shown in Fig. 6 (left).

Surprisingly, generative PCNs trained on ImageNet correctly stored all the presented training images, and correctly reconstructed them with no visible error. This shows that PCNs can be used to store high-dimensional and high-quality images in practical setups, which can be retrieved using only a low-dimensional key, formed by a fraction of the original image. This is, to our knowledge, the first

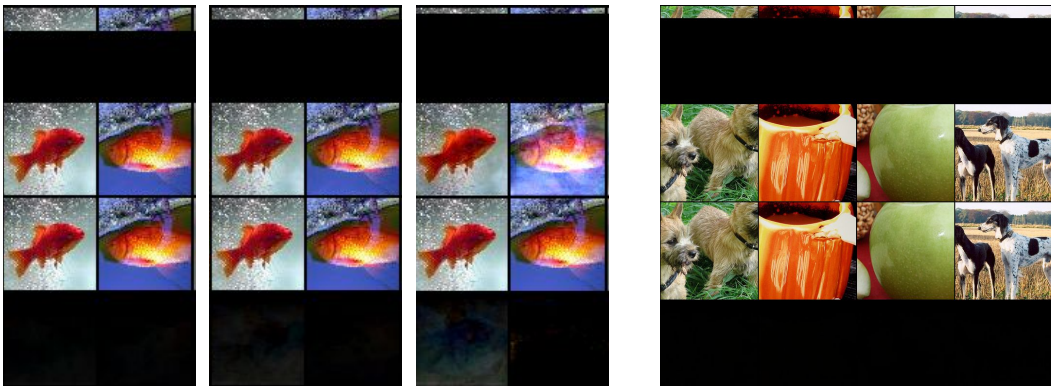


Figure 6: Partial images and their reconstructions (from top to bottom: partial image, reconstructed image, original image, and reconstruction error (difference between original and reconstruction)). Left: 64×64 Tiny ImageNet images presented with $1/4$, $1/8$, and $1/16$ of the original image, respectively, using a network of 1024 hidden neurons. Right: 224×224 ImageNet images presented with $1/8$ of the original image, using a network with 8196 hidden neurons.

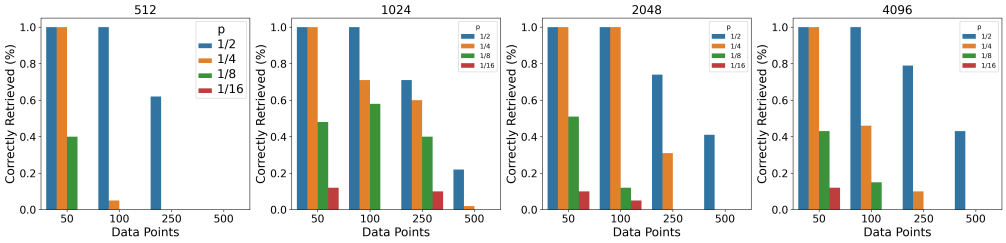


Figure 7: Left: Percentage of correctly retrieved images given different fractions of the original images for PCNs with 512, 1024, 2048, and 4096 hidden neurons trained on datasets of different sizes.

AM model able to reconstruct ImageNet images to this level of precision. Particularly, Fig. 6 (right) shows the perfect reconstruction obtained on ImageNet when the network is presented with only $1/8$ of the original pixels. Further experiments on ImageNet are shown in the comparison with DANNs below and in the supplementary material, where we present multiple high-quality reconstructions.

5 Further Experiments and Comparison with DANNs

In the above experiments, a shallow architecture is able to store 100 natural images from Tiny ImageNet, and to reconstruct them with no visible error when provided with $1/8$ of the original images. We now study how this changes when either the cardinality of the dataset is increased, or the number of provided pixels during reconstruction is updated. To study this, we have trained two models with $n = \{1024, 2048\}$ hidden units on subsets of Tiny ImageNet of cardinality $N = \{50, 100, 250, 500\}$, and reconstructed images with only $p = \{1/2, 1/4, 1/8, 1/16\}$ of the original pixels. According to our visualization experiments, we have noted that an image has no visible reconstruction noise when the error between the original image and the reconstruction is less than 0.001. Hence, we consider an image to be correctly retrieved if the reconstruction error is below this threshold.

Results: Every network with 50 stored memories was able to reconstruct the original image when provided with $1/4$ of the original image, and about half when provided with only $1/8$. However, this changes when increasing the training set, as no network trained on more than 50 images was able to correctly reconstruct an image when provided with $1/16$ of the original images, besides a few cases. In terms of capacity, the more images we train on, the harder the reconstruction becomes: no training image was correctly retrieved when training on $N = 500$ images when provided with a fraction smaller than $1/2$ of the original image, regardless of the width of the network. A summary of these results is given in Fig. 7

These experiments show the limits of shallow generative PCNs on both reconstruction capabilities and capacity. As expected and common in standard AM experiments, increasing the number of training

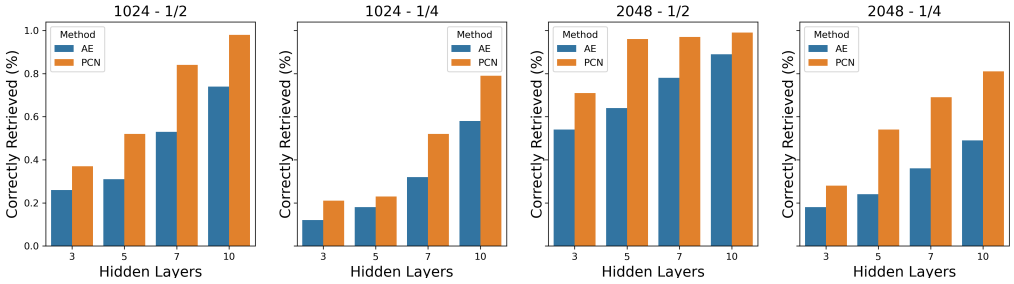


Figure 8: Percentage of correctly retrieved images as a function of the number of hidden layers, for both AEs and generative PCNs, on a dataset of 500 images. We used networks with $n = \{1024, 2048\}$ hidden neurons, and the images are taken from Tiny ImageNet.

points and reducing the number of available information for reconstruction hurts the performance. We now show how increasing the number of layers solves this capacity problem.

5.1 Deep Generative PCNs

We now test how increasing the depth of generative PCNs increases their capacity. We then compare the results against overparametrized AEs. Particularly, we trained generative PCNs with $n = \{1024, 2048\}$ hidden neurons, depth $L = \{3, 5, 7, 10\}$ on $N = 500$ images of Tiny ImageNet. Furthermore, we reconstructed the stored images using a fraction of $p = \{1/2, 1/4\}$ of the original pixels. To further compare against state-of-the-art AEs, we trained equivalent AEs, which is the best-performing one according to [3]. Also, all the hyperparameters used for training are the ones reported by the authors. To make the comparison completely fair, we also assume that the correct pixels of the incomplete images are known when testing the AE. Particularly, we fixed these pixels at every iteration of the reconstruction process of the AEs. As above, we consider an image to be correctly retrieved if the reconstruction error is less than 0.001.

Results: The results confirm the hypothesis: the deeper the network, the higher the capacity. Particularly, a 10-layer network was able to reconstruct more than 98% (against the 72% of the AE) of the images when providing half of the original pixels, and 74% (against the 48% of the AE) of the images when providing $1/4$ of the pixels. This shows that our model clearly outperforms state-of-the-art AEs in image reconstruction, even in the overparametrized regime. Fig. 8 summarizes these results.

5.2 Comparison with Deep Associative Neural Networks

One of the main AM models present in the literature is the dense associative memory network [18], a Hopfield-based architecture with high capacity. The original formulation does not allow to perform image reconstruction experiments, as it is designed to memorize binary information. This model, however, has been extended to continuous datasets, and augmented with a convolutional multilayer structure that performs unsupervised feature detection [19]. The resulting architecture, called *deep associative neural network* (DANN), is able to perform both AM and classification tasks. We now compare the image reconstruction experiments performed by the authors, with the ones performed by our generative PCN. We replicate the experiment proposed, which consists in presenting the network an partial image of the CIFAR10 dataset, where a squared patch covers the centre of each image. Note that this is not a completely fair comparison, as DANNs merge AM with an unsupervised feature detection, and are both memorizing the images and learning the main feature patterns. Hence, comparing it against a pure AM model such as ours may be unfair. However, we believe that presenting comparisons against models that perform AM tasks such as DANNs is still interesting in understanding how our model compares against existing ones in the literature.

Results: We first ran the same experiment on CIFAR10 as in [19]. Particularly, we used 250 images. The comparison is given in Fig. 9. Unlike the image in [19], our reconstruction shows no visible error. Then, we extended the analysis on Tiny ImageNet and ImageNet. The results show that the reconstruction of our AM model are much clearer than the ones in [19], and so consistently improving over this particular task. Furthermore, we replicated the experiment using 50 images of ImageNet, and showed that our model is again able to provide perfect reconstructions of the original images.

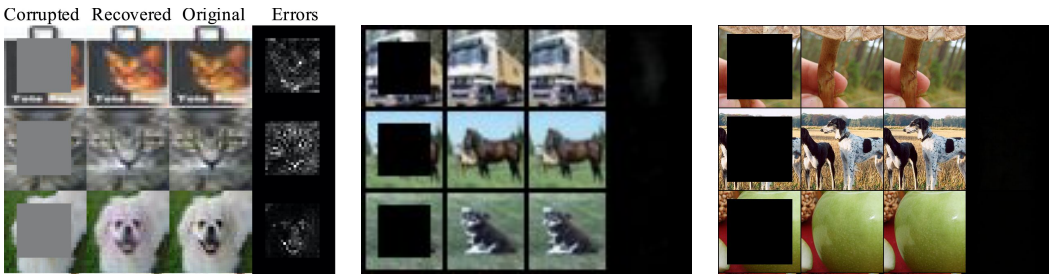


Figure 9: Left: Reconstruction of DANN on CIFAR10 32×32 images (taken from [19]). Centre: Our reconstruction using a generative PCN with 1024 hidden neurons on the same dataset. Right: Our reconstruction on 25 ImageNet 224×224 images and a network with 8192 hidden neurons.

6 Related Work

Predictive coding [9] and classic AM models [1, 21] have always been unrelated topics in the literature. However, they share multiple similarities, the main of which is that both are energy-based models that update their parameters using local information.

Associative Memories (AMs): In computer science, the concept of AMs dates back to 1961, with the introduction of the *learn matrix* [22], a hardware implementation of hetero-associative memories using ferromagnetic properties. However, the milestones of the field are Hopfield networks, presented in the early eighties in their discrete [1] and continuous [21] version. Recently, AMs have increasingly gained popularity. Particularly, a 2-layer version of Hopfield networks with polynomial activations has been shown to have a surprisingly high capacity [18]. This capacity can be increased even more when having exponential activations [23, 24].

The recent focus of the machine learning community on AM models, however, does not lie in applying them directly to solve practical hetero-associative memory tasks. On the contrary, it has been shown that modern architectures are implicitly AMs. Examples of these are standard deep neural networks [5, 3] and transformers [25]. Hence, there is a growing belief that understanding AM models will help the understanding and improvements of deep learning [26, 5, 4, 27]. In fact, different classification models have been shown to perform well when augmented with AM capabilities. DANNs [19], e.g., use deep belief networks to filter information, while [28] improves over standard convolutional models by adding an external memory that memorizes associations among images.

Predictive Coding (PC): This framework includes both a biologically plausible neural architecture and a learning algorithm [9] developed to emulate learning in the visual cortex. Particularly, PC allows to describe multiple phenomena happening in the brain, using a single framework, such as free-energy minimization with local computations [29–32]. From the more practical point of view, PC can be used to train high-performing neural architectures [14, 33], as it has been shown that it can exactly replicate the weight update of BP [15–17].

7 Conclusion

In this paper, we have shown that predictive coding (which was originally designed to simulate learning in the visual cortex) naturally implements associative memories that have a high capacity and a high retrieval accuracy and robustness, and that can be expressed using small and simple fully connected neural architectures. We have shown this empirically in a large number of experiments. Particularly, we have performed denoising and image reconstruction tasks, comparing against standard autoencoders. Furthermore, we have shown that this model is able to reconstruct with no visible error natural high-resolution images of the ImageNet dataset, when provided with a tiny fraction of them. All in all, this work strengthens the connection between the machine learning and the neuroscience community. It underlines the importance of predictive coding in both areas, both as a highly plausible algorithm to better understand how memory and prediction work in the brain, and as an approach to solve corresponding problems in machine learning and artificial intelligence.

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, 1982.

[2] D. Krotov and J. J. Hopfield, "Unsupervised learning by competing hidden units," *Proceedings of the National Academy of Sciences*, vol. 116, no. 16, pp. 7723–7731, 2019.

[3] A. Radhakrishnan, M. Belkin, and C. Uhler, "Overparameterized neural networks implement associative memory," *arXiv preprint arXiv:1909.12362*, 2019.

[4] P. Domingos, "Every model learned by gradient descent is approximately a kernel machine," *arXiv preprint arXiv:2012.00152*, 2020.

[5] T. Poggio, "From associative memories to powerful machines," *CBMM Memo No. 114*, 01/2021 2021.

[6] L. Squire and S. Zola-Morgan, "The medial temporal lobe memory system," *Science*, vol. 253, no. 5026, 1991.

[7] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex.," *Cerebral cortex (New York, NY: 1991)*, vol. 1, no. 1, pp. 1–47, 1991.

[8] E. A. Murray, T. J. Bussey, and L. M. Saksida, "Visual perception and memory: a new view of medial temporal lobe function in primates and rodents," *Annu. Rev. Neurosci.*, vol. 30, pp. 99–122, 2007.

[9] R. P. Rao and D. H. Ballard, "Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects," *Nature Neuroscience*, vol. 2, no. 1, pp. 79–87, 1999.

[10] J. Mcclelland, B. Mcnaughton, and R. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory," *Psychological review*, vol. 102, pp. 419–57, 08 1995.

[11] H. C. Barron, R. Auksztulewicz, and K. Friston, "Prediction and memory: A predictive coding account," *Progress in Neurobiology*, vol. 192, p. 101821, 2020.

[12] K. L. Stachenfeld, M. M. Botvinick, and S. J. Gershman, "The hippocampus as a predictive map," *Nature neuroscience*, vol. 20, no. 11, p. 1643, 2017.

[13] K. Liu, J. Sibille, and G. Dragoi, "Generative predictive codes by multiplexed hippocampal neuronal tuplets," *Neuron*, vol. 99, no. 6, pp. 1329–1341.e6, 2018.

[14] J. C. Whittington and R. Bogacz, "An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity," *Neural Computation*, vol. 29, no. 5, 2017.

[15] Y. Song, T. Lukasiewicz, Z. Xu, and R. Bogacz, "Can the brain do backpropagation? — Exact implementation of backpropagation in predictive coding networks," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, 2020.

[16] T. Salvatori, Y. Song, T. Lukasiewicz, Z. Xu, and R. Bogacz, "Predictive coding can do exact backpropagation on convolutional and recurrent neural networks," *arXiv:2103.03725*, 2021.

[17] T. Salvatori, Y. Song, T. Lukasiewicz, R. Bogacz, and Z. Xu, "Predictive coding can do exact backpropagation on any neural network," 2021.

[18] D. Krotov and J. J. Hopfield, "Dense associative memory for pattern recognition," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), Curran Associates, Inc., 2016.

[19] J. Liu, M. Gong, and H. He, "Deep associative neural network for associative memory based on unsupervised representation learning," *Neural Networks*, vol. 113, pp. 41–53, 2019.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[21] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences*, vol. 81, 1984.

[22] K. Steinbuch, "Die Lernmatrix," *Kybern.*, vol. 1, no. 1, pp. 36–45, 1961.

[23] 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*, vol. 168, 2017.

[24] D. Krotov and J. J. Hopfield, "Large associative memory problem in neurobiology and machine learning," in *International Conference on Learning Representations*, 2021.

[25] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, T. Adler, D. Kreil, M. K. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "Hopfield networks is all you need," in *International Conference on Learning Representations*, 2021.

[26] M. Widrich, B. Schäfl, H. Ramsauer, M. Pavlović, L. Gruber, M. Holzleitner, J. Brandstetter, G. K. Sandve, V. Greiff, S. Hochreiter, *et al.*, "Modern Hopfield Networks and Attention for Immune Repertoire Classification," *arXiv preprint arXiv:2007.13505*, 2020.

[27] V. Feldman and C. Zhang, "What neural networks memorize and why: Discovering the long tail via influence estimation," *CoRR*, vol. abs/2008.03703, 2020.

[28] G. Yang and F. Ding, "Associative memory optimized method on deep neural networks for image classification," *Information Sciences*, vol. 533, 2020.

[29] R. Bogacz, "A tutorial on the free-energy framework for modelling perception and learning," *Journal of Mathematical Psychology*, vol. 76, pp. 198–211, 2017.

[30] K. Friston, "Learning and inference in the brain," *Neural Networks*, vol. 16, no. 9, pp. 1325–1352, 2003.

[31] K. Friston, "A theory of cortical responses," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 360, no. 1456, 2005.

[32] J. C. Whittington and R. Bogacz, "Theories of error back-propagation in the brain," *Trends in Cognitive Sciences*, 2019.

[33] B. Millidge, A. Tschantz, and C. L. Buckley, "Predictive coding approximates backprop along arbitrary computation graphs," *arXiv:2006.04182*, 2020.

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] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This paper do not raise any negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] This paper do not raise any ethical issues.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We do not have theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A] We do not have theoretical results.
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)? [No] No, because the code is proprietary, however, we provide all the information needed to reproduce the results.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See supplementary material.
   (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 supplementary material.
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]
   (b) Did you mention the license of the assets? [N/A] We used public data.
   (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] We used public data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We used public data, which do not contain any of these.
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] We do not use crowdsourcing resources.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We do not use crowdsourcing resources.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We do not use crowdsourcing resources.