ASSOCIATIVE MEMORIES VIA PREDICTIVE CODING

Tommaso Salvatori*
Department of Computer Science
University of Oxford, UK
tommaso.salvatori@cs.ox.ac.uk

Yuhang Song*
Department of Computer Science
University of Oxford, UK
yuhang.song@some.ox.ac.uk

Yujian Hong*
Department of Computer Science
University of Oxford, Oxford, UK
yujian.hong@cs.ox.ac.uk

Simon Frieder
Department of Computer Science
University of Oxford, Oxford, UK
frieder.simon@cs.ox.ac.uk

Lei Sha
Department of Computer Science
University of Oxford, Oxford, UK
lei.sha@cs.ox.ac.uk

Zhenghua Xu
State Key Laboratory
Hebei University of Technology, Tianjin, China
zhenghua.xu@hebut.edu.cn

Rafal Bogacz
MRC Brain Network Dynamics Unit
University of Oxford, UK
rafal.bogacz@ndcn.ox.ac.uk

Thomas Lukasiewicz
Department of Computer Science
University of Oxford, UK
thomas.lukasiewicz@cs.ox.ac.uk

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 \( s \), and heteroassociative memories, able to store and recall multi-modal data. In this paper, we present a novel neural model for realizing associative memories, based on a hierarchical generative network that receives external stimuli via sensory neurons. This model is trained using predictive coding, an error-based learning algorithm inspired by information processing in the cortex. To test the capabilities of this model, we perform multiple retrieval experiments from both corrupted and incomplete data points. In an extensive comparison, we show that this new model outperforms in retrieval accuracy and robustness popular associative memory models, such as autoencoders trained via backpropagation, and modern Hopfield networks. 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 only a tiny fraction of pixels of the original images is presented. Furthermore, we show that this method is able to handle multi-modal data, retrieving images from descriptions, and vice versa. We conclude by discussing the possible impact of this work in the neuroscience community, by showing that our model provides a plausible framework to study learning and retrieval of memories in the brain, as it closely mimics the behavior of the hippocampus as a memory index and generative model.

1 Introduction

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 \([1]\). In particular, visual memories are stored in a hierarchical network of visual and associative areas \([2]\). 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 \([3]\). Accordingly, early visual areas learn common regularities present in the stimuli \([4]\), while at the top of this hierarchy, associative areas (such as hippocampus, entorhinal cortex, and perirhinal cortex) store the relationships between extracted features, which encode an entire stimulus or episode \([5]\). The memory system of the brain is able to both recall complex memories \([1,6]\), and use them to generate predictions to guide behavior \([7]\). Learning in these associative memories shapes our understanding of the world around us and builds the foundations of human intelligence.

Building models that are able to store and retrieve information has been an important direction of research in artificial intelligence. 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 via an energy minimization process when presenting their corrupted variants \([8,9]\). Classic AMs include Hopfield networks and their modern formulation, called modern Hopfield networks (MHNs) \([10]\). The latter are one-shot learners, which are able to store exponentially many memories, and to perfectly retrieve them. However, the retrieval process often fails when dealing with complex data, such as natural images. 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 \([11]\).

In this work, we present a novel AM model that is based on a new energy-based 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 inspired by learning in the visual cortex \([4]\). 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 \([6,12]\). From a machine learning perspective, predictive coding networks (PCNs) are able to perform both supervised and unsupervised tasks with a high accuracy \([4,13]\), and are completely equivalent to backpropagation when trained with a specific algorithm \([14,15,16]\).

We show that the new AM model is not only interesting from a neuroscience perspective, but it also outperforms popular AM models when it comes to the retrieval of complex data points. Our results can be briefly summarized as follows:

- We define generative PCNs and empirically show that they store training data points as attractors of their dynamics by demonstrating that they can restore original data points from corrupted versions. In an extensive comparison of the new AM model against standard AEs, the new model considerably outperforms AEs (in storage capacity, retrieval accuracy, and robustness) when tested on neural networks of the same size.
- The reconstruction of incomplete data points is a challenging task in the field of AMs. Our model naturally solves the task of reconstructing complex and colored images with a surprisingly high accuracy, outperforming autoencoders and MHNs by a large margin on Tiny ImageNet and CIFAR10. We also test our model on ImageNet, perfectly reconstructing single pictures even after removing all but 1/8 of the original image. We then show that to increase the overall capacity of the model, it suffices to add additional layers.
- We show that our model is also able to handle multi-modal data, and hence perform hetero-associative memory experiments. Particularly, we train a model to memorize captioned images, where the captions are taken from a dictionary of 1000 words, and use the description to retrieve the original image and vice versa. Note that retrieving the images from the captions implies retrieving 3072 pixels, using a vector of only 25 dimensions, which corresponds to less than 1/\% of the total information. We show that other AM models fail in performing this complex task.

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 \([17]\). 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 \([4]\). 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 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 \(x^l\), and every pair of consecutive layers is connected via weight matrices \(\theta^l\), 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:

$$\mu_{i,t}^l = \begin{cases} \sum_{j=1}^{n_{l+1}} \theta_{i,j}^l f(x_{j,t}^l) & \text{if } 0 \leq l < L \\ b & \text{if } l = L, \end{cases}$$

(1)

where $f$ is a non-linear activation function. To conclude, the difference between the value $\bar{x}_t^l$ and their predictions $\bar{\mu}_l^t$ is the error $\epsilon_{i,t}^l = x_{i,t}^l - \mu_{i,t}^l$. 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 $E_t$:

$$E_t = \frac{1}{2} \sum_{i,t}(\epsilon_{i,t})^2.$$ 

(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 $\epsilon_{i,t}^0 = s_i - \mu_{i,t}^0$. The process of minimizing $E_t$ by modifying all $x_{i,t}^l$ leads to the following changes in the value nodes:

$$\Delta x_{i,t}^l = \begin{cases} \gamma \cdot (e_{i,t}^l + f'(x_{i,t}^l) \sum_{k=1}^{n_{l-1}} e_{k,t}^{l-1} \theta_{k,i,t}^l) & \text{if } 0 < l \leq L \\ 0 & \text{if } l = L, \end{cases}$$

(3)

where $\gamma$ is the integration step, a constant determining by how much the activity changes in each iteration. The computations in Eqs. (1) and (3) are biologically plausible, as they have a neural implementation that can be realized in a network with value nodes $x_{i,t}^l$ and error nodes $e_{i,t}^l$, as shown in Fig 1. The inference phase works as follows: starting from a given configuration of the value nodes $x_{0}$, inference continuously updates the value nodes according to Eq. (3) until it has converged. We call the configuration of the value nodes at convergence $x_T$, where $T$ is the number of steps needed to reach convergence (in practice, it is a fixed large number).
Inference

When the value nodes of the sensory layer are fixed to an input signal we presented the model with a corrupted variant (by adding Gaussian noise) of the training set. We then used the PCNs was stored as an attractor, we expect the prediction \( \bar{x} \) to be the configuration that is not a local minimum. While running inference, it may converge to it when minimizing the total energy via running inference. Right: Representation of a generative PCN, while the pseudocode can be found in Alg. 1. Detailed derivations of graphs and matrices are in the supplementary material (and in [13]).

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 \( \alpha \) is the learning rate. Particularly, the derived update rule is the following:

\[
\Delta \theta_{i,j}^{l+1} = -\alpha \cdot \delta E_T / \partial \theta_{i,j}^{l+1} = \alpha \cdot \varepsilon_{i,T} f (w_{i,j}^{l+1}), \\
\Delta b_i = -\alpha \cdot \delta E_T / \partial b_i = -\alpha \varepsilon_{i,T}.
\]

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 of generative PCNs, while the pseudocode can be found in Alg. 1. Detailed derivations of Eqs. (3) and (5) are in the supplementary material (and in [13]).

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 \( \bar{\epsilon} \in \mathbb{R}^d \) 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., \( \bar{x}_i = \bar{\epsilon} \) for the whole process. Then, we run inference until convergence and save the prediction \( \bar{\mu}_T^L \) of the sensory layer. If the original data point was stored as an attractor, we expect the prediction \( \bar{\mu}_T^L \) to be a less corrupted version of it. Let \( F : \mathbb{R}^d \rightarrow \mathbb{R}^d \) be the function that sends \( \bar{\epsilon} \) to \( \bar{\mu}_T^L \) 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 \( b_T \) 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 the street view house number dataset (SVHN), Tiny ImageNet, and CIFAR10. After training, we 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
Figure 3: 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. Top row corresponds to experiments performed on CIFAR10, middle row the SVHN dataset, bottom row to Tiny Imagenet.

Experiments: We trained 2-layer PCNs with ReLU non-linearity and hidden dimension $n \in \{256, 512, 1024, 2048\}$ on subsets of the aforementioned datasets 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 of a 2-layer PCN is smaller than the one of a 3-layer AE with the same hidden dimension. This follows, as the additional layer (input layer, not needed in generative PCNs) almost doubles the number of parameters in some cases. 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. 3). AEs trained with BP did not perform well: AEs with less than 1024 hidden neurons always failed to restore even a single data point on the Tiny Imagenet dataset, and very few on other ones. 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 correctly store datasets of 250 images of both CIFAR10 and Tiny ImageNet, and that networks with 2048 hidden 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 \( x^{(1)} \) 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 \( x^{(j)} \) 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. Note that the geometry of the mask does not influence the final performance, as our model simply memorizes single pixels.

**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 0.001. Furthermore, we plotted the partial images together with their reconstructions, for a visual check. Note that we have used the thresholds that provided the fairest comparison: the denoising experiments fail to have a perfect retrieval, despite the fact that most of the images look visually good. Hence, we have determined the threshold to be equal to 0.005. Then, with the same threshold for the retrieval of partial images, our method always successfully retrieved all the images, and so we have opted for a smaller threshold, which was more informative.

**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 above 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.
Algorithm 2 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 $s'_i$.
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: return $\bar{x}^0_T$

Inference

Figure 5: 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.

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 perfectly 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 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 [11]. 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.

6 Comparison with Modern Hopfield Networks

Hopfield networks are the most famous AM architectures. However, they have a limited capacity and only work with binary datasets. To solve these limitations, modern Hopfield networks (MHNs) were introduced [18][19]. There are significant differences between MHNs and our generative PCNs, which strongly influence the possible application
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.

Table 1: Number of correctly retrieved images by an MHN trained on $N = \{50, 100, 250, 500\}$ images of the CIFAR10 dataset on different tasks. Left: The model was provided with $\frac{1}{2}$ of the original image. Centre: The model was provided with $\frac{1}{4}$ of the original image. Right: The model was provided with a corrupted version of the original image, with injected Gaussian noise $\eta = 0.2$.

6.1 Comparison with Deep Associative Neural Networks

As shown, the original formulation of MHNs does not allow to perform image reconstruction experiments. To solve this problem, a new model has been presented, which consists of a MHN augmented with a convolutional multilayer...
structure that performs unsupervised feature detection using a deep belief network. The resulting architecture, called deep associative neural network (DANN), is not a pure associative memory model, as it 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 a partial image of the CIFAR10 dataset, where a squared patch covers the centre of each image.

Note that the comparison we propose is purely qualitative, as DANNs augment MHNs with an unsupervised feature detection, using convolutions to further improve their results, and are both memorizing the images and learning the main feature patterns. This allows them to have a large capacity, which allows DANNs to be trained on large datasets. Hence, comparing it against a pure AM model such as ours may not be indicative. 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 [20]. Particularly, we used 500 images. The comparison is given in Fig. 10. Unlike the image in [20], our reconstruction shows no visible error. The results show that the reconstruction of our AM model are much clearer than the ones in [20], 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. However, DANNs show a larger capacity, as the provided images are obtained after training on the whole CIFAR10 dataset.

7 Hetero-Associative Memory Experiments

Generally speaking, associative memories come in two high-level forms: auto-associative and hetero-associative memories. While both are able to retrieve memories given a set of inputs, auto-associative memories are primarily
focused on recalling a specific pattern when provided a partial or noisy variant of X. So far, we have only performed this kind of task. By contrast, hetero-associative memories are able to recall not only patterns of different sizes from their inputs, but are also able to memorize multi-modal information, such as associate natural language with sounds or images. In this section, we study the performance of this model with respect to multi-modal dataset, by training it on images and their textual descriptions, and then using the description to retrieve the related image.

**Experiment:** The dataset used for this task is *flickr_30k* [21], which consists of colored captioned images of varying size, that we resize and reduce to $32 \times 32$ colored images. Each image has 5 associated captions, from which we only use the first caption. The captions are tokenized by first building a vocabulary (using the package *spacy*) and then discretized using said vocabulary. Afterwards, they are transformed into floating point numbers and normalized to $[0, 1]$, to remain in the same range as the images. Finally, the captions are padded with 0s to the size of the longest caption, to maintain the same data size. For example, the captions associated with the first three images are:

1. Two young guys with shaggy hair look at their hands while hanging out in the yard;
2. Several men in hard hats are operating a giant pulley system;
3. A child in a pink dress is climbing up a set of stairs in an entry way.

Note that the task of retrieving images from captions is complex, as every pixel of an image image of size 3072 has to be retrieved using a key of dimension 25 (hence, using < 1% of the original information).

To show how depth influences the retrieval quality, we have trained different PCNs with depth $L \in \{1, 3, 5, 7, 9, 11\}$ and width $n \in \{512, 1024, 2048\}$ to memorize 25, 50, and 100 captioned images. Then, the networks are given the images without to recover the captions and vice-versa. The number of recovered captions and recovered images are then recorded. An image is considered to be recovered if its error is less than 0.001, while a caption is considered to be recovered if it is *exactly* recovered: unlike pixels, words are embedded as discrete information, and hence a word is either exactly retrieved, or mistaken by a wrong word of the dictionary. A caption is considered to be correctly recovered if every word is correctly recovered.

**Results:** The number of images and captions recovered are given in Fig. [11]. We found that increasing the number of neurons in each layer did not affect the results much, then only the best result from each depth is shown. The results indicate that increasing the number of layers in the network increases the capacity of the network, as expected. Most of the networks successfully recovered most of 25 images from the respective captions, which means that from $\approx 1\%$ of the original data the networks remembered the remaining $99\%$ and more than half of the images when trained on 50 datapoints. When trained on 100 datapoints, only deep networks (9 and 11 hidden layers) were able to perfectly retrieve more than half of the datapoints. The shallow networks of depth 1 performed poorly on all the tasks, not recovering any image or caption. Compared to AEs, PCNs perform slightly worse when retrieving captions, but significantly better when retrieving pictures, as AEs were never able to retrieve more than two, regardless of the parametrization or the number of data points used.

The results show that generative PCNs are able to both handle multimodal data, and retrieve stored memories using less than 1% of the original information. Furthermore, an analysis of the retrieved images of the AEs, show that they are always able to correctly retrieve images of the dataset, even when it does not correspond to the one we were looking for. This shows that AEs store images as attractors, but some attractors are stronger than others, and hence most of the retrieval processes converge to the unrelated memories. This result is visually shown in Fig. [12] which represents the recovery of 25 images by both PCNs and AEs.

**8 Biological Relevance**

The model described in this paper is closely related with theories of functioning of the memory system in the brain. It is typically assumed that memories are stored in a network of cortical areas organized hierarchically, where sensory information is processed and encoded by a several neo-cortical regions, and then it is integrated across modalities in the hippocampus [2][5]. It has been recently proposed that this hippocampo-neocortical system can be viewed as a predictive coding network [6] similar to that simulated in this paper. In particular, Barron et al. [6] proposed that the hippocampus could be viewed as a top layer in the hierarchical predictive coding network (which would correspond to the memory layer $L$ in our model), while neo-cortical areas would correspond to lower layers in the network. A contribution of our paper is to implement this idea and demonstrate in simulations that predictive coding networks, which were only theoretically proposed to act as a memory system [6], can actually work and effectively store and retrieve memories.

The mechanisms of learning and retrieval proposed in the predictive coding account of the brain memory system in [6] closely resemble those occurring in our model. Barron et al. proposed that during learning, the generative model of the hippocampus is updated until the hippocampal predictions correct the prediction errors by suppressing neo-cortical
Figure 11: Depths comparison for generative PCNs with respect to hetero-associative memories. The top row shows the results for generative PCNs, and the bottom row the ones for AEs. The number of captions (resp., images) recovered is shown on the left (resp., right) side. Both models are trained for the depths $L \in \{1, 3, 5, 7, 9, 11\}$ and the widths $n \in \{512, 1024, 2048\}$, with the best number across $n$ being shown.

9 Related Work

Predictive coding [4] and classic AM models [8, 26] 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 \( [27] \), 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 \([8]\) and continuous \([26]\) 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 \([10]\). This capacity can be increased even more when having exponential activations \([28, 19]\). To improve the retrieval process of stored memories, different energy-based models have been developed, such as \([29]\).

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 \([30, 11]\) and transformers \([18]\). Hence, there is a growing belief that understanding AM models will help the understanding and improvements of deep learning \([31, 50, 52, 53]\). In fact, different classification models have been shown to perform well when augmented with AM capabilities. DANNs \([20]\), e.g., use deep belief networks to filter information, while \([34]\) improves over standard convolutional models by adding an external memory that memorizes associations among images.

Predictive Coding (PC): PC is an influential theory of cortical function in both theoretical and computational neuroscience. It is unique in the sense that it provides a computational framework, able to describe information processing in multiple brain areas \([35]\). It has different theoretical interpretations, such as free-energy minimization \([36, 37, 35, 38]\) and variational inference of probabilistic models \([13]\). Furthermore, its single mechanism is able to account for different perceptual phenomena observed in the brain. Among them, we can list end-stopping \([4]\), repetition-suppression \([39]\), illusory motions \([40, 41]\), bistable perception \([42, 43]\), and attentional modulation of neural activity \([44, 45]\).

Due to this solid biological grounding, PC is also attracting interest in the machine learning community recently, especially focusing on finding the links between PC and BP \([13, 46]\). From the more practical point of view, PC can be used to train high-performing neural architectures \([13, 46]\), as it has been shown that it can exactly replicate the weight update of BP \([14, 15, 16]\).

10 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 retrieval accuracy and robustness, and that can be expressed using small and simple fully connected neural architectures. This has been extensively shown in a large number of experiments, for different architectures and datasets. Particularly, we have performed denoising and image reconstruction tasks, comparing against popular AM frameworks. 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. From the neuroscience perspective, we have shown that our model closely resembles the “learn and recall” phases of the hippocampus.
Furthermore, 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.

Acknowledgments

This work was supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1, by the AXA Research Fund, the EPSRC grant EP/R013667/1, and by the EU TAILOR grant. We also acknowledge the use of the EPSRC-funded Tier 2 facility JADE (EP/P020275/1) and GPU computing support by Scan Computers International Ltd.

References

[1] L. Squire and S. Zola-Morgan, “The medial temporal lobe memory system,” Science, vol. 253, no. 5026, 1991.
[2] 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.
[3] 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.
[4] 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.
[5] J. McClelland, B. Menaughton, 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.
[6] H. C. Barron, R. Aukstulewicz, and K. Friston, “Prediction and memory: A predictive coding account,” Progress in Neurobiology, vol. 192, p. 101821, 2020.
[7] 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.
[8] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” Proceedings of the National Academy of Sciences, vol. 79, 1982.
[9] 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.
[10] D. Krotov and J. J. Hopfield, “Dense associative memory for pattern recognition,” in Advances in Neural Information Processing Systems, 2016.
[11] A. Radhakrishnan, M. Belkin, and C. Uhler, “Overparameterized neural networks implement associative memory,” arXiv preprint arXiv:1909.12362, 2019.
[12] 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.
[13] 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.
[14] 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.
[15] 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.
[16] T. Salvatori, Y. Song, T. Lukasiewicz, R. Bogacz, and Z. Xu, “Predictive coding can do exact backpropagation on any neural network,” 2021.
[17] 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.
[18] H. Ramsauer, B. Schäffl, 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.
[19] D. Krotov and J. J. Hopfield, “Large associative memory problem in neurobiology and machine learning,” in *International Conference on Learning Representations*, 2021.

[20] 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.

[21] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, “Flickr30k entities: Collecting region-to-phase correspondences for richer image-to-sentence models,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV ’15, (USA), p. 2641–2649, IEEE Computer Society, 2015.

[22] T. V. Bliss and T. Lømo, “Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path,” *The Journal of Physiology*, vol. 232, no. 2, pp. 331–356, 1973.

[23] R. C. O’reilly and J. L. McClelland, “Hippocampal conjunctive encoding, storage, and recall: avoiding a trade-off,” *Hippocampus*, vol. 4, no. 6, pp. 661–682, 1994.

[24] K. K. Cowansage, T. Shuman, B. C. Dillingham, A. Chang, P. Golshani, and M. Mayford, “Direct reactivation of a coherent neocortical memory of context,” *Neuron*, vol. 84, no. 2, pp. 432–441, 2014.

[25] K. Z. Tanaka, A. Pevzner, A. B. Hamidi, Y. Nakazawa, J. Graham, and B. J. Wiltgen, “Cortical representations are reinstated by the hippocampus during memory retrieval,” *Neuron*, vol. 84, no. 2, pp. 347–354, 2014.

[26] 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.

[27] K. Steinbuch, “Die Lernmatrix,” *Kybern.*, vol. 1, no. 1, pp. 36–45, 1961.

[28] 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.

[29] S. Bartunov, J. W. Rae, S. Osindero, and T. P. Lillicrap, “Meta-learning Deep Energy-based Memory Models,” *arXiv preprint arXiv:1910.02720*, 2019.

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

[31] M. Widrich, B. Schäfl, H. Ramsauer, M. Pavlović, L. Gruber, M. Holzlzeitner, 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.

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

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

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

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

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

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

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

[39] R. Auksztulewicz and K. Friston, “Repetition suppression and its contextual determinants in predictive coding,” *Cortex*, vol. 80, 2016.

[40] W. Lotter, G. Kreiman, and D. Cox, “Deep predictive coding networks for video prediction and unsupervised learning.” *arXiv:1605.08104*, 2016.

[41] E. Watanabe, A. Kitaoka, K. Sakamoto, M. Yasugi, and K. Tanaka, “Illusory motion reproduced by deep neural networks trained for prediction,” *Frontiers in Psychology*, vol. 9, p. 345, 2018.

[42] J. Hohwy, A. Roepstorff, and K. Friston, “Predictive coding explains binocular rivalry: An epistemological review,” *Cognition*, vol. 108, no. 3, 2008.

[43] V. Weilnhammer, H. Stuke, G. Hesselmann, P. Sterzer, and K. Schmack, “A predictive coding account of bistable perception-a model-based fmri study,” *PLoS Computational Biology*, vol. 13, no. 5, 2017.
[44] H. Feldman and K. Friston, “Attention, uncertainty, and free-energy,” *Frontiers in Human Neuroscience*, vol. 4, 2010.

[45] R. Kanai, Y. Komura, S. Shipp, and K. Friston, “Cerebral hierarchies: Predictive processing, precision and the pulvinar,” *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 370, 2015.

[46] B. Millidge, A. Tschantz, and C. L. Buckley, “Predictive coding approximates backprop along arbitrary computation graphs,” *arXiv:2006.04182*, 2020.
A Detailed Derivation of Eqs. (3) and (5)

A.1 Derivation of Eq. (3)

By expanding Eq. (2) with the definition of \( \varepsilon_{i,t}^l = x_{i,t}^l - \mu_{i,t}^l \), we have:

\[
E_t = \sum_{l=0}^{L-1} \sum_{n=1}^{n_l^l} \frac{1}{2} (\varepsilon_{i,t}^l)^2 = \sum_{l=0}^{L-1} \sum_{n=1}^{n_l^l} \frac{1}{2} (x_{i,t}^l - \mu_{i,t}^l)^2 .
\]

(6)

Inference minimizes \( E_t \) by modifying \( x_{i,t}^l \) proportionally to the gradient of the objective function \( E_t \). We note that each \( x_{i,t}^l \) influences \( E_t \) in two ways: (i) it occurs in Eq. (6) explicitly, but (ii) it also determines the values of \( \mu_{k,t}^{l-1} \) via Eq. (1). Therefore, the derivative of \( E_t \) over \( x_{i,t}^l \) contains two terms, which are formally as follows:

\[
\Delta x_{i,t}^l = -\gamma \cdot \frac{\partial E_t}{\partial x_{i,t}^l}
\]

(7)

\[
\Delta x_{i,t}^l = -\gamma \cdot \left( \frac{\partial \frac{1}{2}(x_{i,t}^l - \mu_{i,t}^l)^2}{\partial x_{i,t}^l} + \frac{\partial \sum_{k=1}^{n_{l-1}^l} \frac{1}{2} (x_{k,t}^{l-1} - \mu_{k,t}^{l-1})^2}{\partial x_{i,t}^l} \right) 
\]

(8)

\[
\Delta x_{i,t}^l = \gamma \cdot \left( - (x_{i,t}^l - \mu_{i,t}^l) + f'(x_{i,t}^l) \sum_{k=1}^{n_{l-1}^l} (x_{k,t}^{l-1} - \mu_{k,t}^{l-1}) \theta_{k,i}^l \right) 
\]

(9)

\[
\Delta x_{i,t}^l = \gamma \cdot \left( - \varepsilon_{i,t}^l + f'(x_{i,t}^l) \sum_{k=1}^{n_{l-1}^l} \varepsilon_{k,t}^{l-1} \theta_{k,i}^l \right) .
\]

(10)

Considering also the special cases of \( l = L \) and \( l = 0 \), we obtain Eq. (3).

A.2 Derivations of Eq. (5)

The update of weights minimizes \( E_t \) by modifying \( \theta_{i,j}^{l+1} \) proportionally to the gradient of the objective function \( E_t \). We note that \( \theta_{i,j}^{l+1} \) affects the value of the function \( E_t \) of Eq. (6) by influencing \( \mu_{i,t}^l \) via Eq. (1), hence, the derivative of the objective function \( E_t \) over \( \theta_{i,j}^{l+1} \) can be formally defined as:

\[
\Delta \theta_{i,j}^{l+1} = -\alpha \cdot \frac{\partial E_t}{\partial \theta_{i,j}^{l+1}}
\]

(11)

\[
\Delta \theta_{i,j}^{l+1} = -\alpha \cdot \frac{\partial \frac{1}{2}(x_{i,t}^l - \mu_{i,t}^l)^2}{\partial \theta_{i,j}^{l+1}}
\]

(12)

\[
\Delta \theta_{i,j}^{l+1} = \alpha \cdot \varepsilon_{i,t}^l .
\]

(13)

B Further Details on the Experiments

Here, we provide further details about training PCNs, useful to reproduce them. Particularly, every PCN was trained using IL until convergence, we varied the following hyperparameters and reported the best result: \( T \in \{12, 16, 24, 32\} \) for 2-layer networks, and \( T \in \{32, 48, 72\} \) for multilayer ones, \( \gamma \in \{1, 0.5, 0.1, 0.05, 0.01\} \), and \( \alpha \in \{0.0001, 0.00005\} \). Furthermore, we used standard PyTorch initialization and ReLU activations in every layer. To retrieve corrupted images, we have iterated the function \( F \) 30 times, using \( T \in \{100, 250, 500\} \), and reported the best results.
C Analysis of Different Levels of Noise

Figure 13: Percentage of correctly retrieved images by 2-layer generative PCNs (PC) with hidden-layer dimensions of 512, 1024, and 2048, when presented with a corrupted image with Gaussian noise of different variances, trained on the Tiny ImageNet dataset.

So far, we analyzed images with Gaussian noise of variance 0.2. We now extend the analysis to different levels of corruptions. Particularly, we have used networks of width \( n \in \{512, 1024, 2048\} \) trained on \( N \in \{100, 250, 500\} \) natural images taken from the Tiny ImageNet dataset, and tried to retrieve them using Gaussian noise of variance \( \eta \in \{0.3, 0.5, 0.7, 1.0\} \).
Results: The results show that our method is able to retrieve stored data points even with a larger amount of variance, although the results get worse the more we increase the level of noise. Particularly, when presented with data points with Gaussian noise with variance 0.3, every network was able to restore at least one image, even when trained on 500 examples. The numbers of retrieved images decreased when increasing the noise. When presented with noise with variance $\eta = 1.0$, only networks with 2048 hidden units were able to retrieve a tiny fraction of the original data points. We report the results in Fig. 13.

D Plots of Images with Extreme Levels of Noise

![Images with Extreme Levels of Noise]

Figure 14: Analysis of the reconstruction capabilities of generative PCNs under different levels of noise $\eta \in \{1.2, 1.5, 1.7, 2.0\}$. Every plot was obtained using a 2-layer generative PCN with 2048 hidden units, trained on 100 images of the Tiny ImageNet dataset.

We now plot the images generated using different levels of corruptions. Note that all the images shown here were clearly classified as not retrieved by the above analysis. However, we believe that an analysis of not well-retrieved images is still interesting, to understand the limitations of our method. Hence, we trained a generative PCN on 100 images of the Tiny ImageNet dataset, and tried to reconstruct them using extreme levels of noise. Then, we printed the reconstructions.

Results: These representations show that generative PCNs are able to identify the original data points, even if the retrieval process was not accurate enough for the error to fall below the decided threshold 0.005. Particularly, our model was able to identify original memories even when presented with extreme levels of noise ($\eta \in \{1.2, 1.5, 1.7, 2.0\}$), although leaving visible amount of corruption. The reconstructions are given in Fig. 14.
Figure 15: Analysis of the reconstruction capabilities of 10 iterations of the function $F$ under different levels of noise $\eta \in \{0.3, 0.5, 0.7, 1.0\}$. Particularly, we plotted the results after 1, 6, and 11 iterations of $F$.

E Analysis of the Retrieval Function

Here, we show visually how the function $F$ restores corrupted images. Particularly, we trained a generative PCN with 2048 hidden neurons on 100 images of the Tiny ImageNet dataset, and printed the reconstructions after 1, 6, and 11 iterations of the retrieval function $F$.

Results: The reconstructions show that the first iteration is able to clear most of the noise. However, further iterations are needed to clear the remaining details, especially if the noise level is high. In fact, it can be observed that one iteration of $F$ is able to retrieve the original data point when the level of corruption is $\eta = 0.3$, but fails when $\eta = 1.0$. This process usually converges around $m = 15$, depending on the number of iterations $T$ and the level of noise. The results are shown in Fig. 15.
F Reconstruction of Partial Images

Figure 16: Percentage of correctly retrieved images as a function of the number of hidden layers, for generative PCNs on a dataset of 500 images of the CIFAR10 dataset. We used networks with \( n = \{1024, 2048\} \) hidden neurons, and tried to reconstruct them using partial images containing a fraction of 1/2 and 1/4 of the original pixels.

To further show the robustness of our experiments, we replicate the experiments of Section 5.1 on the CIFAR10 dataset, training networks of different depths \( (L \in \{5, 7, 10\}) \) to generate 500 images, and report the results in Fig. 16. Again, we consider a data point to be perfectly reconstructed if the error between the original image and the reconstruction is below 0.001.

**Results:** The plots confirm the results stated in the body of this work: the deeper the network, the more images are perfectly reconstructed.

G Analysis of the Retrieval Time

Table 2: Time (in seconds) to perform a retrieval from corrupted images with injected Gaussian noise of variance \( \eta = 0.2 \).

| Hidden Dimension | 256 | 512 | 1024 | 2048 |
|------------------|-----|-----|------|------|
| One Iteration    | 0.02| 0.04| 0.09 | 0.15 |
| Five Iterations  | 0.10| 0.19| 0.53 | 0.84 |
| Ten Iterations   | 0.55| 1.18| 2.85 | 4.37 |

Table 3: Time (in seconds) to perform a retrieval from an incomplete image, where only 1/2 of the original pixels are provided.

| Hidden Dimension | Error = 0.005 | Error = 0.001 |
|------------------|---------------|---------------|
| 256              | 0.74          | 14.69         |
| 512              | 1.08          | 24.19         |
| 1024             | 1.83          | 37.73         |
| 2048             | 2.48          | 53.12         |

In the AM literature, it is important to quickly retrieve an image. In fact, modern Hopfield networks and similar models are able to retrieve memories 1-shot. Our model is slower than other AM models, as the retrieval process relies on an energy minimization framework, however, it is able to retrieve a memory in a couple of seconds. Note that this does not alter its biological plausibility, mainly because of two reason:

- The brain performs computations much more efficiently, and hence the same process could take milliseconds. The neural information processing is, in fact, certainly not based on digital GPU/CPU hardware and may realize the presented algorithm in a much more natural (and efficient) way, such as using analogue relaxation dynamics to rapidly converge to the attractor state.
- The retrieval process takes some time, because it retrieves exact memories at the pixel level, while memory recall is not as detailed. If the inference process is stopped after a shorter period of time, it simply recalls a slightly fuzzier memory. This flexibility of variable-length computation is also highly desirable and realistic from a biological viewpoint.
The time in seconds needed to retrieve original images from corrupted data is given in Table 2, while the time needed to retrieve original images from incomplete data is given in Table 3. Note that these numbers provide an upper bound on the real efficiency of the model, limited by the current implementation. In fact, while every computation during the retrieval process can theoretically run in parallel, this cannot be implemented in popular deep learning frameworks such as PyTorch and Tensorflow. A correct implementation, which would require significant engineering work, going beyond the scope of this paper, could make this process significantly faster.

All experiments are conducted on two Nvidia GeForce GTX 1080Ti GPUs and eight Intel Core i7 CPUs, with 32 GB RAM.

H Full-Page Reconstructions of ImageNet

To show the reconstruction capabilities of our method, in what follows we plot full-page reconstructions of ImageNet pictures.
Figure 17: To show the level of accuracy of our proposed method, we represent reconstructions of ImageNet pictures. Here, a reconstruction when a patched image is presented to our generative PCN. From top to bottom: partial image, reconstructed image, original image, and reconstruction error (difference between original and reconstruction).
Figure 18: To show the level of accuracy of our proposed method, we represent reconstructions of ImageNet pictures. Here, a reconstruction when only 1/8 of the original picture is presented to our generative PCN. From top to bottom: partial image, reconstructed image, original image, and reconstruction error (difference between original and reconstruction).