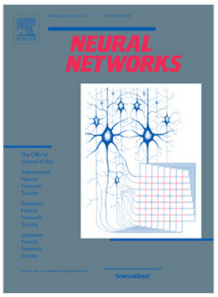
Accepted Manuscript



Deep associative neural network for associative memory based on unsupervised representation learning

Jia Liu, Maoguo Gong, Haibo He

PII: S0893-6080(19)30015-2
DOI: https://doi.org/10.1016/j.neunet.2019.01.004
Reference: NN 4087

To appear in: *Neural Networks*

Received date : 1 July 2018
Revised date : 31 October 2018
Accepted date : 20 January 2019

Please cite this article as: J. Liu, M. Gong and H. He, Deep associative neural network for associative memory based on unsupervised representation learning. *Neural Networks* (2019), https://doi.org/10.1016/j.neunet.2019.01.004

Deep Associative Neural Network for Associative Memory Based on Unsupervised Representation Learning

Jia Liu[a,*], Maoguo Gong[a], Haibo He[b,*]

[a]*Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xian, Shaanxi Province 710071, China*
[b]*Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881, USA*

Abstract

This paper presents a deep associative neural network (DANN) based on unsupervised representation learning for associative memory. In brain, the knowledge is learnt by associating different types of sensory data, such as image and voice. The associative memory models which imitate such a learning process have been studied for decades but with simpler architectures they fail to deal with large scale complex data as compared with deep neural networks. Therefore, we define a deep architecture consisting of a perception layer and hierarchical propagation layers. To learn the network parameters, we define a probabilistic model for the whole network inspired from unsupervised representation learning models. The model is optimized by a modified contrastive divergence algorithm with a novel iterated sampling process. After training, given a new data or corrupted data, the correct label or corrupted part is associated by the network. The DANN is able to achieve many machine learning problems, including not only classification, but also depicting the data given a label and recovering corrupted images. Experiments on MNIST digits and CIFAR-10 datasets demonstrate the learning capability of the proposed DANN.

Keywords:
Unsupervised representation learning, deep neural network, associative memory, image recovery

1. Introduction

Mimicking the efficiency and robustness by which the human brain represents information has been a core challenge in artificial intelligence research for decades [1]. Neural networks are excellent learning models which are inspired from the neurons and their connections in the brain. With the development of computational capability, neural networks with deep layers have achieved many breakthroughs in a wide range of machine learning applications [2, 3, 4],

*This work was conducted when J. Liu was a joint Ph.D. student at the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881, USA.

**Corresponding author. E-mail address: haibohe@uri.edu.

which result in a new research field of deep learning [5]. Deep neural networks contains more trainable parameters and more propagation layers than shallow neural networks which greatly increases the modeling and abstraction capability of networks although training a deep network is much more difficult [6]. Hinton did seminal works where a layer-wise training paradigm was proposed for training a deep neural network efficiently [7, 8]. Then the increase of computational capability further decreases training difficulty of deep networks. Many network architectures such as convolutional neural network (CNN) [9] and recurrent neural network (RNN) [10] have been used to construct deeper networks and achieved state-of-art results in many applications.

The layer-wise training paradigm which reveals the efficiency of deep neural networks was further developed as unsupervised representation learning [11]. Unsupervised representation learning is used to learn the features from raw data or pre-train a deep network layer-wise without considering the labels. There are various unsupervised models trying to learn efficient features, including restricted Boltzmann machine (RBM) [7], auto-encoder [12], and their derivatives [13, 14, 15]. We also did several works in this paradigm [16, 17, 18]. Unsupervised learning had a catalytic effect in reviving interest in deep learning, but has since been overshadowed by the successes of purely supervised learning [5] due to the development of GPU parallel computation. Supervised deep learning follows the "end-to-end" architecture with the input of raw data and the output of label, definition, or other types of data. The features of raw data will be learnt adaptively and automatically in the hierarchical network. With enough iterations and many training tricks, the deep network can be trained to achieve the satisfactory results without pre-training. As a consequence, unsupervised learning plays less important roles in deep learning. However, as discussed by LeCun, Bengio, and Hinton in [5], human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object. The "end-to-end" architecture may not well imitate the learning process of human.

In human learning, an actual behaviour is that data and label are both input of brain. For example, given a picture of a cat, the teacher told you that "This is a cat". The image and the voice "This is a cat" are both transferred to the brain by eyes and ears respectively. Then the two signals propagate through the visual cortex and auditory cortex respectively and finally are associated and memorized by the brain. When you see a similar animal, you remember the definition of "cat". The active learning process is important in early stage of a human when he/she has much less feedback to the environment. It is the basic for a human to understand the world. Associative memory models [19, 20, 21, 22, 23, 24, 25] which have been researched for decades imitate such a learning process.

Associative memory can be divided into two categories, hetero-associative (HA) and auto-associative (AA) memories [19] from the perspective of function. HA memory links one pattern to another, such as images and texts. AA memory associates a pattern with itself, which is useful for recovering a pattern from a corrupted data. From the perspective of technology, there are probability based associative memory and neural associative memory models. Probability based associative memory consists of a multilayer array of processing elements (PEs). Each PE estimates the probability of each input pattern and output the corresponding status according to several rules [23]. Then with a missing element in input, the output and

another input element can be used to associate the missing status. Each PE in the array can self-organize by dynamically adapting its function in response to the input data. However, there are only two binary input units in each PE and therefore it has limited processing capability. Neural associative memory stores patterns in a memory storage matrix [19, 25] which is usually a recurrent architecture with one layer of neurons. Then with a corrupted input, the similar pattern will be associated by recurrently updating the status of these neurons. However, limited by its simple and shallow architecture, the data they processed are usually discrete sequences or binary images [21, 22]. With the development of big data, those models cannot satisfy the requirement of processing large scale complex data. The associative memory models have been overshadowed by feed-forward deep neural networks. There are less works on associative memory last decade [19, 25]. Dmitry Krotov and John J. Hopfield who proposed the widely known Hopfield neural network [26] which is the prototype of neural associative memory discussed the associative memory with large capability in [25] and find the equivalent relationship between the associative memory and a feed-forward neural network with one hidden layer. In order to improve the feature learning ability of associative memory for dealing with large scale complex data, in this paper, we consider to achieve the associative learning process with a hierarchical deep neural network. Considering the difficulty of extending existing associative memory models into deep ones, we propose our model inspired from unsupervised representation learning techniques.

Most unsupervised representation learning models are defined based on the principle of representing the visible data by hidden layers. We find that the representation ability means the network captures the probability distribution of visible data as well as the associative relationship between the elements in data. Therefore, we attempt to construct a deep neural network and train the network parameters via unsupervised representation modeling methods in order to learn associations between data and label or elements in input data. The proposed deep network consists of a perception layer and hierarchical propagation layers without output layer. The propagation layers are used to extract the features of input data and the architecture follows the data type, for instance, convolutional layers can be used for image. The perception layer is the processing core of the whole network. We define an energy function inspired from Hebb learning rule [27] for the perception layer and propose a probabilistic model for the whole network. It deserves to be noted that the probability model is very similar to that of RBM, but RBM is a single layer network. Therefore, in the proposed framework, the main difficulty is to optimize the probabilistic model although the contrastive divergence algorithm [28] is used. We propose a novel sampling method for the deep network to reconstruct the input data during the learning process. After learning, the network is able to associate the data and label as well as elements in input data. Then given a data, the label can be searched from the label space by optimizing the probability density function. Conversely, given the label, a data can be reconstructed and given the data with some missing elements, the network can also predict the missing elements. The proposed model imitates the learning process of human with an unsupervised manner. Meanwhile, it is based on the hierarchical neural network and therefore can process complex high dimensional data. The contributions of this paper can be summarised as follows:

- A novel hierarchical deep associative memory model is proposed based on the unsupervised representation learning. The model inherits the deep learning property of hierarchical feature learning and neural learning property of associative learning. Therefore, it is able to deal with high dimensional data and make complex associations.
- A hierarchical unsupervised representation training method is proposed. In deep learning, most unsupervised learning models are composed of a visible layer and a hidden layer and the deep architecture is constructed by stacking them layer by layer. In this paper, we directly model and optimize the whole network by an unsupervised manner. An improved contrastive divergence algorithm with a novel iterated sampling method is proposed.

The rest of this paper is organized as follows. Section II introduces related work and background about associative memory and unsupervised representation learning. The architecture, model, and learning process are described in detail in Section III. Experiments on different applications with different network architectures are implemented in Section IV. In Section V, we conclude this paper.

2. Related Work and Background

Associative memory is closer to the learning process of human. However, existing models are difficult to deal with large scale complex data. First we introduce some associative memory models.

2.1. Associative Memory

2.1.1. Probability Based Associative Memory

As introduced above, the probability based associative memory is constructed by a multilayer array of PE. Each PE stores the observed probabilities of each status of the two input elements I_1 and I_2 with each element being 0 or 1. The output of the PE is determined by the probabilities according to several rules [23]. With missing input, the PE makes associations in three manners, input only association (IOA), output only association (OOA) and input-output association (INOUA). In IOA, one input is defined while another input and the output status back propagated from other PEs are undefined. In OOA, only the feedback signal is defined and the two input should be associated. In INOUA, only one input is undefined and should be determined according to its observed probability. The whole network for associative memory is a hierarchical architecture composed of sparsely connected self-organizing PEs as shown in Figure 1. Each PE is more likely to connect to other PEs with a short distance. In the training process, each PE estimates the probabilities of the status of its input which will be used to make associations in the feedback operation. Then with missing elements in the input vector, the associations will be made by feedback signals and observed input according to the connecting structure as shown in Figure 1. The probability based associative memory is able to implement both hetero- and auto-associations and to construct a deeper network with the more complex data. However, the basic element of the associative network, i.e., PE may not well handle the large scale complex data, such as nature images and the architecture is difficult to determine since there are only two binary input elements in each PE.

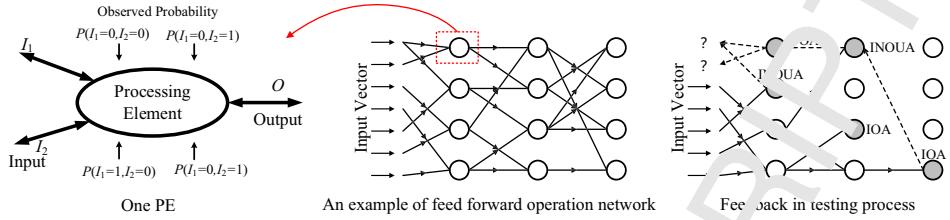


Figure 1: Probability based associative memory. In training process, all the input data are fed to the network and the probabilities in each PE are calculated. In test process, feedback operation is used to make correct associations and recover the missing parts, where gray PEs represent the associative processing elements and the dotted lines represent the association feedback critical path.

2.1.2. Neural Associative Memory

For AA memory, the neural associative memory uses a system of N binary neurons with vales ± 1 [25]. The model is driven by an energy function:

$$E = -\frac{1}{2}\sum_{i,j=1}^{N} x_i T_{ij} x_j, \quad T_{ij} = \sum_{\mu=1}^{K} \xi_i^{\mu}\xi_j^{\mu} \tag{1}$$

where the vector x denotes a status of all the neurons and ξ^{μ} denotes a memory among K memories. A dynamical updating rule is defined to decrease the energy when the network is presented with a corrupted input and the network respond with a stored memory which most closely resembles the input [25]. For HA memory, there are pattern pairs x^{μ} and y^{μ} to be stored in a matrix C with $c_{ij} = \sum_{\mu=1}^{K} x_i^{\mu} y_j^{\mu}$ which denotes the connection weight between neurons i and j [19]. The model can store large amount of patterns although the architecture of them are usually single layer network and is difficult to be stacked to form a deep network.

The above associative memory models achieved good results in many applications. However, they only consider the learner while ignore the feature learning process which is the key to success of deep learning. Deep neural network provides an efficient way for dealing with large scale complex data with a hierarchical deep architecture. In this paper, we attempt to construct a deep architecture to learn the associative memories. But it is difficult for existing models to extend to deep layers. As a consequence, we propose a deep associative neural network (DANN) based on unsupervised representation learning to learn the associations between patterns or input elements.

2.2. Unsupervised Representation Learning

Unsupervised learning had a catalytic effect in reviving interest in deep learning [5] because it provides an efficient way for training a deep network which revels the efficiency of deep architectures. RBM and auto-encoder are two basic unsupervised learning models from which many efficient unsupervised learning models are derived. RBM is a two-layer bidirectional graphical model with a visible layer and a hidden layer. The two layers are connected by a symmetrical weight matrix W and there are no connections in the same layer. RBM can be taken as a Markov random field which represents the input data with hidden layer units [15]. Therefore, RBM is driven by the joint distribution over the visible and hidden units

which is defined by $P(v,h) = (1/Z)\exp[-E(v,h)]$ where Z is the normalization parameter which is the sum over all the status of v and h, i.e., $Z = \sum_v \sum_h \exp[-E(v,h)]$. $E(v,h)$ is the energy function which is defined as:

$$E(v,h) = -\sum_i \sum_j v_i W_{ij} h_i - \sum_j b_j h_j - \sum_i c_i v_i \tag{2}$$

where b and c are the bias vector of hidden and visible layers, respectively. The parameters of RBM, W, b and c are learnt by maximizing the probability that the network assigns to a visible vector $P(v) = \sum_h P(v,h)$ in order to well represent it. It can be optimized by gradient descent on the log-likelihood of training data. However, due to the normalization parameter, it is difficult to estimate the equilibrium distribution by many times of Gibbs sampling. Hinton proposed the contrastive divergence algorithm [28] to solve this problem. Contrastive divergence algorithm modifies the objective function and only one or several Gibbs sampling is needed which greatly increases the computational efficiency for training a RBM. Since RBM aims to increase the probability $P(v)$, the network can represent the input data well which means that the distributions of the input data can be captured by the network. Then with a corrupted input, the correct distribution can be associated by maximizing $P(v)$ with respect to v. As a consequence, RBM is very suitable to learn the associative memory.

Auto-encoder is a neural network model which usually consists of visible and hidden layers as well. It defines an encoder $h = f_e(v,\theta)$ and a decoder by single layer neural network $v' = f_d(h,\theta')$ which encodes the data and reconstructs the original data with the code, respectively. The parameters θ and θ' are learnt by minimizing the reconstruction error $L(v,v') = \sum_v \|v - v'\|_2^2$ which amounts to carrying out the optimization $\min \mathbb{E}_{p(v)}[L(v,v'(v))]$ where $\mathbb{E}_{p(v)}$ is the expectation of the distribution of input data. Note that v' is a deterministic function of v, and therefore the reconstruction error can be rewritten with the log-likelihood form $\max \mathbb{E}_{p(v)}[\log p(v|v' = f_d(f_e(v)))]$ which is equivalent to $\max \mathbb{E}_{p(v)}[\log p(v|h = f_e(v)); \theta']$ [29]. This means that by minimizing the reconstruction error, the code h can well represent the input data which achieves the same effect to that of RBM.

Unsupervised learning provides an efficient way to learn the associative memory due to its distribution learning for the input data. However, most unsupervised learning models are proposed for single-layer network and the deep network is constructed by stacking them layer by layer. It is a forward process. For a corrupted data, it is difficult for a stacked architecture to recover the data backward. Therefore, in this paper, we propose to model the whole network by unsupervised representation learning manner as RBM. The new model is able to learn the associative memories for large scale complex data.

3. Methodology

In this section, we will describe the proposed DANN in detail including architecture, model, and learning process. At last, we provide an analysis for DANN to demonstrate the validity of the model and learning process.

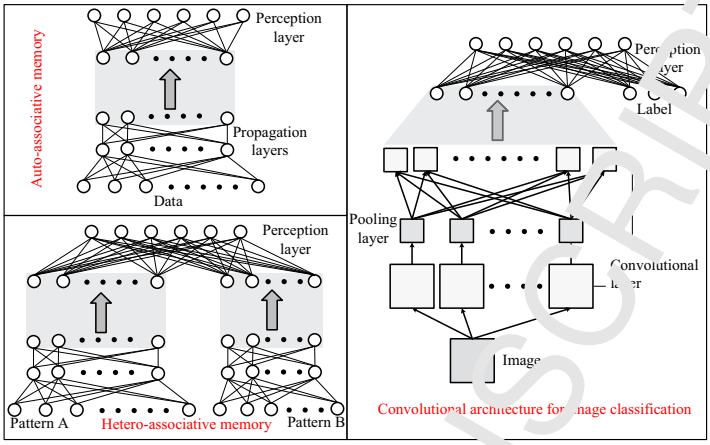


Figure 2: Architectures of DANN on different conditions.

3.1. DANN's Architecture

DANN is composed of a perception layer and hierarchical propagation layers. The propagation layers aim to extract the abstract features for memory. With two different type of patterns, such as data and label, there will be two propagation paths. The perception layer is the core to learn and memory on which the energy and probability model is defined. Note that the propagation layers can vary with different types of data as shown in Figure 2. For vectorial data, the full connected architecture can be used and for 2-dimensional data such as images, the convolutional and pooling layers will be used. Both HA and AA memories can be implemented by DANN with different architectures. For the AA memory architecture, the network stores one pattern and then one propagation path is needed. The top layer is the perception layer which captures the distribution of input data. Thus we call the architecture "end-to-perception". For HA memory, two patterns such as patterns A and B shown in Figure 2, are fed into the network through two propagation paths to the perception layer. An example of image classification is given in the right graph in Figure 2. Since the image is the complex data, a deep architecture is necessary to extract the abstract features. Therefore, the convolutional and pooling layers are constructed. The label is the simple binary data and then it is directly fed to the perception layer together with the feature vector.

3.2. Probabilistic Model for DANN

Similar to RBM, we propose to learn the network parameters by best capturing the distribution of input data or pattern pairs. Therefore, we define a probability model $P(x)$ on the perception layer with respect to the feature vector extracted from propagation layers denoted by $f(x, \varphi)$ with x being the input data and φ being the parameter set in the propagation layers:

$$P(x; \varphi, \theta) = \frac{1}{Z} \exp[-E(f(x, \varphi), \rho, \theta)] \tag{3}$$

where ρ denotes the vector in the top perception layer which is computed by neural activity $\rho = f_e(f(x, \varphi), \theta)$ like auto-encoder. θ denotes the network parameter set including weight

matrix W and bias vector b in the perception layer, i.e., $\theta = \{W, b\}$. Similarly, Z is the normalization function to guarantee that the sum over all the probabilities is 1, i.e., $Z = \sum_{\chi} \exp(-E(f(\chi, \varphi), \theta)$ where χ denotes one of all the possible data over the whole data space. The probability model is driven by the energy function $E(v, f_e(v), \theta)$ which is defined inspired from the Hebb learning rule [27, 18]:

$$E(v, f_e(v), \theta) = -\sum_{ij} f_e(v)_i v_j W_{ij} - \sum_{j} f_e(v)_i b_i \tag{4}$$

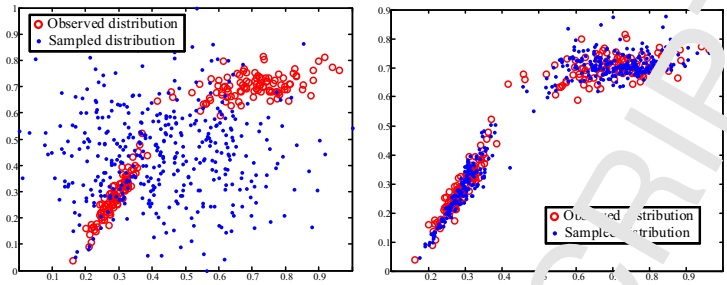
Different from RBM and auto-encoder, here v represents the feature vector fed to the perception layer, $v = f(x, \varphi)$. Hebb learning describes the plasticity of a synapsis between two neurons. When both neurons are activated, the connection between them will be increased. Then the energy is defined based on the inner product between the product of the status in two neurons and the connecting strength between them. Such an energy is widely used in the neural networks, such as RBM and Hopfield neural network [30]. This model is similar to that of RBM. However, the proposed model is based on the neural network and the value of perception layer vector is the deterministic function of the input vector and then it directly models the input data instead of the joint probability of visible and hidden units. Moreover, this model is implemented on the whole network instead of the single-layer network with the input data propagating to make the perception. Then the parameters of propagation layers, φ, will be learnt along with the learning process of perception layer and the task induced features will be learnt.

Like RBM and other unsupervised representation learning models, the aim of the probabilistic model is to best capture the distribution of observed data by a deep architecture. Maximizing the parameterized probabilistic model means that decreasing the energy of observed data while hold the energy of all the other data by optimizing the network parameters. Then the observed data will be at the high density region of the parameterized probability distribution. Thus the observed data will be well modeled by the deep network. After training, the whole story will be revealed from a clue. Therefore, with this unsupervised representation learning inspired model, DANN is able to learn associative memories. A toy example is shown in Figure 3 where little red circles denote the observed data with 2 dimensions. The blue dots denote the sampled data from the network. Before trained, the randomly initialized network generates a random distribution as shown in Figure 3 (a). After trained by the observed data, the network generate a distribution which is similar to that of observed data as shown in Figure 3 (b). The sampled data distribute around the observed data which means that the trained DANN well captures the distribution of observed data. The training process and a novel sampling method will be described in the next subsection.

The above model is based on the AA memory architecture and it is easy to derive the HA memory model as shown in Figure 2 by integrating another vector propagated from another path:

$$P(x, y; \varphi, \theta) = \frac{1}{Z} \exp[-E([f(x, \varphi_1), f(y, \varphi_2)], \rho, \theta)] \tag{5}$$

where φ_1 and φ_2 denotes the propagation parameter set in the two propagation paths, respec-



(a) (b)

Figure 3: A toy example showing the distributions of observed data and sampled data from DANN. (a) Sampling from randomly initialized DANN. (b) Sampling from trained DANN by observed data.

tively. For the other architectures, it is also easy to derive the corresponding models, such as the convolutional architecture in Figure 2:

$$P(x, l; \varphi, \theta) = \frac{1}{Z} \exp[-E(f_c(x, \varphi_c), l], \rho, \theta)] \tag{6}$$

where l is the label with the corresponding input data. $f_c()$ and φ_c denotes the propagation function and parameter set of the convolutional architecture, respectively. As a consequence, the proposed model is able to adapt to different tasks and data types with corresponding network architecture. It is of great importance to design an efficient learning process for this model although it is much difficult to optimize a probabilistic hierarchical model.

3.3. DANN's Learning Process

The network parameters can be trained by maximizing the probability over all the observed data $\max_{\varphi,\theta} \sum_x P(x)$ to best represent the input data. This can be achieved by computing the log-likelihood gradient as the updating direction:

$$\begin{aligned}
\triangle[\varphi, \theta] &= \frac{\partial \log(\sum_x P(x))}{\partial \varphi, \theta} \\
&= \frac{\partial \log \sum_x \exp(-E(x))}{\partial \varphi, \theta} - \frac{\partial \log \sum_\chi \exp(-E(\chi))}{\partial \varphi, \theta} \\
&= -(\sum_x \frac{\exp(-E(x))}{T} \frac{\partial E(x)}{\partial \varphi, \theta} - \sum_\chi \frac{\exp(-E(\chi))}{Z} \frac{\partial E(\chi)}{\partial \varphi, \theta}) \\
&= -(\langle \frac{\partial E(x)}{\partial \varphi, \theta} \rangle_{q^0} - \sum_\chi P(\chi) \frac{\partial E(\chi)}{\partial \varphi, \theta})
\end{aligned} \tag{7}$$

where T is the sum over all the observed data $T = \sum_x \exp(-E(x))$ and $\langle\rangle_{q^0}$ denotes the expectation of the partial differential over all the observed data, i.e., over the observed distribution q^0. It is easy to compute the first term with the observed data. However, it is much more difficult to estimate the second term directly because it is impossible to obtain all the possible

data in the whole data space, especially for large scale data. It can be estimated by numerous Gibbs sampling to obtain the equilibrium distribution q^∞ for fantasy data. However, it is time consuming to sample the input data numerous times. RBM encounters the same problem and Hinton proposed a contrastive divergence algorithm [28] to solve this problem.

3.3.1. Contrastive Divergence Algorithm

Optimizing the log-likelihood of the data amounts to minimizing the Kullback-Liebler (KL) divergence between the distribution of the observed data q^0 and the equilibrium distribution q^∞ generated by numerous Gibbs samplings. The KL divergence is defined by:

$$\begin{aligned} q^0 \| q^\infty &= q^0 \log q^0 - q^0 \log q^\infty \\ &= -H(q^0) - \langle \log q^\infty \rangle_{q^0} \end{aligned} \tag{8}$$

where $H(q^0)$ is the entropy of observed data and is a constant. The equilibrium distribution q^∞ is assigned by the network and actually is $P(x; \varphi, \theta)$ defined in Eq. (3). Then the derivative of the log-likelihood can be written as:

$$\langle \frac{\partial \log q^\infty}{\partial \varphi, \theta} \rangle_{q^0} = -(\langle \frac{\partial E(\chi)}{\partial \varphi, \theta} \rangle_{q^0} - \langle \frac{\partial E(\chi)}{\partial \varphi, \theta} \rangle_{q^\infty}) \tag{9}$$

Instead of minimizing $q^0 \| q^\infty$, contrastive divergence algorithm minimizes the difference between $q^0 \| q^\infty$ and $q^1 \| q^\infty$, where q^1 is the distribution of the reconstructed data generated by one step of Gibbs sampling. The contrastive divergence can only be zero if the model is perfect [28] which implies that $q^1 = q^\infty$. Then the update gradient in Eq. (7) can be rewritten as:

$$\Delta[\varphi, \theta] = \langle \frac{\partial - E(x)}{\partial \varphi, \theta} \rangle_{q^0} - \langle \frac{\partial - E(\tilde{x})}{\partial \varphi, \theta} \rangle_{q^1} \tag{10}$$

where $\tilde{x}$ is the reconstructed input data by one step of sampling. Gibbs sampling method samples two types of variables. It is easy for a single-layer network to sample such as RBM, because it is convenient to compute the probabilities of input or output neurons. However, for the hierarchical architecture, it is much difficult to estimate the status probability of each input unit. It is even impossible to derive the probabilities as in Gibbs sampling with the complex architecture. In this paper, we propose an optimization based method to reconstruct the sampled input data.

3.3.2. Reconstructing $\tilde{x}$

The reconstructed data $\tilde{x}$ should have been sampled from the distribution $P(x; \varphi, \theta)$. This means that the sampled data is more likely to locate in the high density region of the probability distribution. Then we can obtain $\tilde{x}$ by maximizing the probability with respect to x, i.e., $\tilde{x} = \arg\max_x P(x; \varphi, \theta)$. Fortunately, during this process, the network parameters φ and θ keep fixed which leads to Z being a constant. As a consequence, $\tilde{x}$ can be driven to move from a uniform distribution region to the high density region of distribution $P(x; \varphi, \theta)$ by maximizing $\exp(-E(x))$. Such a learning process can be achieved by back-propagation algorithm.

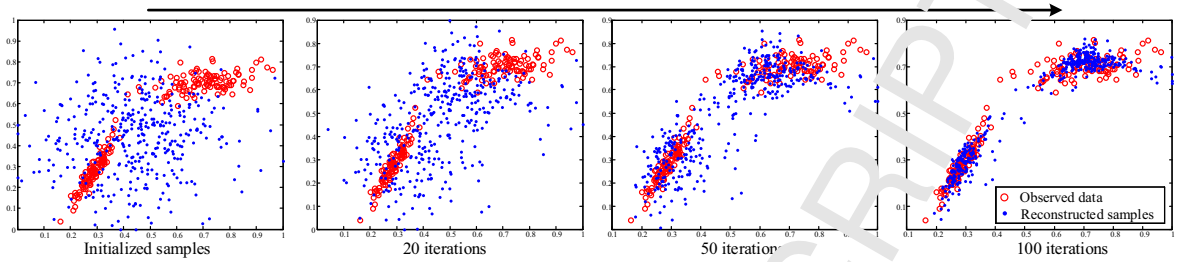


Figure 4: Illustration of the reconstruction process on the toy example.

Back-propagation is implemented by back-propagating errors of the top layer to lower layers to compute the gradient in each layer which is widely used in "end-to-end" architectures. In this "end-to-perception" architecture, the information back-propagated is different from that of the "end-to-end" architectures. The update gradient is computed by $\partial \exp(-E(x))/\partial x = -\exp(-E(x))\partial E(x)/\partial x$. The gradient information can be back-propagated from the perception layer:

$$\frac{\partial E(x)}{\partial x} = \frac{\partial E(x)}{\partial f(x,\varphi)} \frac{\partial f(x,\varphi)}{x} \tag{11}$$

where $f(x,\varphi)$ is the general propagation function. It is important to derive the partial gradient in the perception layer:

$$\begin{aligned} \delta_{v_t} &= \frac{\partial E(v,\rho,\theta)}{\partial v_t} = -\frac{\partial \sum_{ij}\rho_i v_j W_{ij} + \sum_i \rho_i b_i}{\partial v_t} \\ &= -\sum_i \rho_i W_{it} - (\sum_{ij} v_j W_{ij} + \sum_i b_i)\frac{\partial \rho_i}{\partial v_t} \\ &= -\sum_i \rho_i W_{it} - (\sum_{ij} v_j W_{ij} + \sum_i b_i) g_a(v_t,\rho_i) W_{it} \end{aligned} \tag{12}$$

where $v = f(x,\varphi)$ and $g_a(v_t,\rho_i)$ is the derivative of the active function, for example, with a sigmoid function, $g_a(v_t,\rho_i) = \rho_i(1-\rho_i)$. v_t is the t-th units in the input to the perception layer. Then the gradient information δ_{v_t} is back-propagated through the propagation layers to the input layer to obtain δ_x. $\tilde{x}$ will be updated according to:

$$\tilde{x}^{n+1} = \tilde{x}^n - \exp(-E(\tilde{x}^n))\delta_x \tag{13}$$

where n is the number of iterations. Then with several iterations, the reconstructed $\tilde{x}$ locating in the high density region of the distribution is obtained. With x and $\tilde{x}$, the updating gradient of the network parameters can be obtained according to Eq. (10). Figure 4 illustrates the reconstruction process with a trained DANN on the toy example in Figure 3. The red circles denote the observed data which are taken as reference. First the samples are randomly initialized. With increasing iterations, the reconstructed samples gradually move to the high density region of the observed distribution. This process achieves similar result to general sampling methods even though they are implemented in totally different ways.

3.3.3. Updating Network Parameters

Note that the derivatives of φ and θ are different. First for φ:

$$\frac{\partial E(x)}{\partial \varphi} = \frac{\partial E(x)}{\partial f(x,\varphi)} \frac{\partial f(x,\varphi)}{\partial \varphi}. \tag{14}$$

where $\partial f(x,\varphi)/\partial\varphi$ is computed following the back-propagation algorithm for "end-to-end" architectures and $\partial E(x)/\partial f(x,\varphi)$ is computed as the back-propagated information δ_v in Eq. (12). Then δ_v will be back-propagated as error information to compute the network parameters in the propagation layers. For the perception layer, it is easier to derive the gradient:

$$\begin{aligned} \triangle W_{kl} &= \frac{\partial - E(v,\rho,\theta)}{\partial W_{kl}} = \frac{\partial \sum_{ij} v_i W_{ij} + \sum_i \rho_i b_i}{\partial W_{kl}} \\ &= \rho_k v_l + (\sum_j v_j W_{kj} + b_k) \frac{\partial \rho_k}{\partial W_{kl}} \\ &= \rho_k v_l + (\sum_j v_j W_{kj} + b_k) g_a(v_l, \rho_k) v_l \end{aligned} \tag{15}$$

The whole learning procedure is summarized in **Algorithm** 1. The updating procedure

Algorithm 1 Learning Procedure for DANN

Step 1. Initialization: randomly initializing φ and θ.

Step 2. Obtaining reconstructed data.

Step 2.1. Randomly initializing $\tilde{x}$

Step 2.2. Updating $\tilde{x}$ according to Eq. (13) for several iterations.

Step 2.3. Repeating Steps 2.1 and 2.2 for several times to estimate $\langle \frac{\partial - E(\tilde{x})}{\partial \varphi, \theta} \rangle_{q^1}$.

Step 3. Updating network parameters φ and θ.

Step 3.1. Computing the first term in Eq. (10) by using the observed data.

Step 3.2. Computing the second term in Eq. (10) by using the reconstructed data.

Step 3.3. Computing the update gradient by Eq. (10) and update φ and θ.

Step 4. Repeating **Steps 2** and **3** until the stop criterion is satisfied.

can be implemented on the whole data or data batches with a large scale dataset. After the learning procedure, the network is able to capture the distribution of observed data as shown in Figure 2. The above learning procedure is based on the AA memory architecture. It is convenient to derive the learning procedure for HA memory architecture with the gradient information back-propagated from two separate paths. With a convolutional architecture, the gradient information is back-propagated by de-convolution operator.

3.3.4. Making Association via DANN

The proposed DANN makes association by sampling the missed element, x_i for example, from the distribution $P(x_i|x_1, ..., x_{i-1}, x_{i+1}, ...; \varphi, \theta)$:

$$\begin{aligned} & P(x_i|x_1, ..., x_{i-1}, x_{i+1}, ...; \varphi, \theta) \\ &= \frac{P(x; \varphi, \theta)}{\sum_{\chi_i} P(x_1, ..., x_{i-1}, \chi_i, x_{i+1}, ...; \varphi, \theta)} \\ &= \frac{\exp(-E(x))}{\sum_{\chi_i} \exp(-E(x_1, ..., x_{i-1}, \chi_i, x_{i+1}, ...))} \end{aligned} \tag{16}$$

Similar to reconstruction process, the sampling process is achieved by maximizing the probability with respect to the missed elements:

$$\tilde{x}_i = \arg\max_{\tilde{x}_i} \frac{\exp(-E(x_1, ..., x_{i-1}, \tilde{x}_i, x_{i+1}, ...))}{\sum_{\chi_i} \exp(-E(x_1, ..., x_{i-1}, \chi_i, x_{i+1}, ...))} \tag{17}$$

Since all the elements except x_i and network parameters are fixed and χ_i is all the possible values in the data space, the denominator in Eq. (17) is a constant. Therefore, the missed element $\tilde{x}_i$ can be obtained by $\max_{x_i} \exp(-E(x_1, ..., x_{i-1}, \tilde{x}_i, x_{i+1}, ...))$ using the same method as in reconstruction process with the other elements fixed.

3.3.5. Additional Learning Tips

The proposed DANN is with a deep architecture with numerous trainable parameters for learning. Therefore, some tricks are used to improve the computational efficiency. First of all, reconstruction process, i.e., **Step 2** in **Algorithm 1**, is time consuming. We define a neighborhood prior for this process when input data are images. It is known that adjacent pixels in an image usually have similar gray values. Thus the neighborhood prior is defined by:

$$C(\tilde{x}) = \sum_{(i,j)} \sum_{(k,l)\in\Omega_{ij}} \frac{1}{\sqrt{(k-i)^2 + (l-j)^2}} (\tilde{x}(i,j) - \tilde{x}(k,l))^2 \tag{18}$$

where (i, j) denotes the pixel position in an image and Ω_{ij} denotes the neighbor pixel set with the pixel (i, j) as the central pixel. This constraint is added to the objective function for reconstruction, i.e., $\max \exp(-E(\tilde{x})) - \lambda C(\tilde{x})$ with λ controlling the importance of the two terms. Then during the learning process, some subtle pixels will be constrained to speed up the learning process.

Another obstacle that decreases the computational efficiency is the vanishing gradient in deep architecture by using back-propagation. The problem can be relieved by using proper active functions. In this model, we use the rectified linear units (ReLu) [31] active function for propagation layers and sigmoid function for perception layer. The linear part in ReLu can avoid the saturation regime in sigmoid function and back-propagate more gradient information. Such an active function can relieve the vanishing gradient effectively. For perception layer, ReLu may cause the gradient explosion due to that ReLu has no upper limit and the model decreases the energy implicitly. Therefore, the sigmoid function is used to increase

the stability during the learning process.

3.4. Analysis of DANN

The proposed DANN aims to maximize the probability the network assigns to the observed input data $P(x;\varphi,\theta)$, i.e., a joint probability over all the elements in the vector, i.e., $P(x_1, x_2, ..., x_m;\varphi,\theta)$ with m being the number of dimensions of the vector x. Taking a classification problem as an example, the joint probability can be represented by $P(x_d, x_l;\varphi,\theta)$ with x_d being the data and x_l being the label. Maximizing the probability over the observed data means that the observed data achieves the lowest energy through the network among all the possible data in the space:

$$\exp(-E(x_d, x_l)) \geq \forall_{\chi_d,\chi_l} \exp(-E(\chi_d,\chi_l)) \tag{19}$$

Then the conditional probability $P(x_l|x_d;\varphi,\theta) = \exp(-E(x_d, x_l))/\sum_{\chi_l}\exp(-E(x_d,\chi_l))$ will be maximized. This means that the data and label will be associated together and given a data with missed label, the network is able to sample correct label from $P(x_l|x_d;\varphi,\theta)$. Conversely, with a label, a data can also be sampled. Moreover, if we take the label as the missed part, the corrupted data can also be recovered. Whether the proposed model could achieve the expected result depends on the effectiveness of the contrastive divergence algorithm inspired learning process. Figure 5 shows the behaviours of some variables during the learning process.

First the learning process aims to maximize the probability the network assigns to the observed data. This means that the energy distinction between the observed data and other data will be increased. Figure 5 (a) shows the exponential energy changes of observed data, random data and data in the observed space, i.e., $\exp(-E(x_o))$, $\exp(-E(x_r))$, and $\exp(-E(x_{or}))$ where x_o, x_r, and x_{or} respectively denotes the observed data, random data, and data in the observed space. x_r is obtained by uniformly sampling in the whole data space and x_{or} is obtained by uniformly sampling between two observed data which means that x_{or} is not the observed data but the data in the observed space. From Figure 5 (a), with the increase of iterations, both $\exp(-E(x_o))$ and $\exp(-E(x_r))$ increase while $\exp(-E(x_o))$ increases with a higher speed. After about 2500 iterations, $\exp(-E(x_r))$ begin to flatten out while $\exp(-E(x_o))$ continue to increase. This demonstrate that during the learning process, the contrastive divergence algorithm is able to reduce the energy of observed data while hold the other data. $\exp(-E(x_{or}))$ increases along with the $\exp(-E(x_o))$ which demonstrates that DANN models the observed data space via finite observed data. It is different from traditional associative memory models that store limited patterns. In other words, DANN does not recover an image from fixed patterns like most existing associative memory models but from the learned distribution density. Then an approximated probability $P(x_o) \approx \sum_{x_o}\exp(-E(x_o))/\sum_{x_r}\exp(-E(x_r))$ is computed and exhibited in Figure 5 (b). The increase of both $P(x_o)$ and $P(x_{or})$ demonstrates the effectiveness of the learning process.

As analysed above, the DANN achieves classification by maximizing the conditional probability $P(x_l|x_d;\varphi,\theta) = \exp(-E(x_d, x_l))/\sum_{\chi_l}\exp(-E(x_d,\chi_l))$. Then we plot the approximate conditional probability $P(x_l|x_d) \approx \sum_{x_o}\exp(-E(x_l, x_d))/\sum_{x_{rl},x_d}\exp(-E(x_{rl}, x_d))$ in each

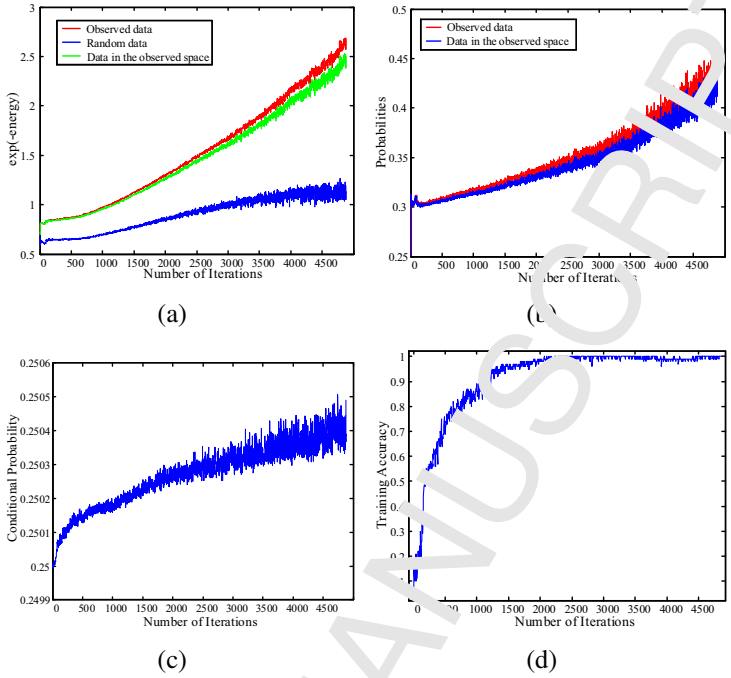


Figure 5: Plots of intermediate data during the learning process. (a) Exponential negative energies of observed data, random data, and data in the observed space, i.e., $\exp(-E(x_o))$, $\exp(-E(x_r))$, and $\exp(-E(x_{or}))$ in each iteration (b) Approximate probabilities of observed data and data in the observed space, i.e., $P(x_o)$ and $P(x_{or})$ in each iteration (c) Conditional probability between data and label, i.e., $P(x_l|x_d)$ in each iteration (d) Training accuracy in each iteration

iteration shown in Figure 5 (c). x_d and x_l denote the data and label respectively. x_{rl} is the randomly generated label of the observed data. In general, the conditional probability increases along with the iterations. This means that the correct label will be more deterministic via the model with the increase of iterations. On the contrary, given the label or a corrupted data, the conditional probability will also be increased via the learning of the model. The training accuracy in each iteration is exhibited in Figure 5 (d). It gradually converge to 1 which means that the network is able to achieve classification tasks by recovering the label for data.

4. Experimental Study

The experiments are implemented with two problems, image classification and image recovery. MNIST digits and CIFAR-10 datasets are used in the experiments. First these datasets and the detailed architectures are introduced.

4.1. Datasets and network architecture settings

The MNIST dataset consists of handwritten digits with 60000 training samples and 10000 test samples from digit "0" to "9". Each sample is an image of the size being 28×28 pixels with a label indicating the class the sample belongs to. MNIST dataset is widely used in many neural network models [15, 25, 32]. CIFAR-10 dataset consists of color images with

the size of 32×32 categorized into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. There are 50000 training and 10000 test images. Because the images are distinct with each other, even the images of the same class and they are captured with complex background, it is of great difficulty to achieve high classification accuracy on the test images. Many works have been done to improve the classification performance on this dataset [33, 34].

For different datasets and tasks, the network architectures are also different. For MNIST dataset, the full connecting architecture like DBN [7] and convolutional architecture like CNN [9] can be used. For classification, the label can be combined with the data and fed to the network simultaneously using the AA architecture or separately using the HA architecture. For CIFAR-10 dataset, since the input image is much larger than that of MNIST dataset, convolutional architecture is used and the label is directly fed into the perception layer. In the experiments on image recovery, the convolutional AA architecture is used. Besides architecture, many other parameters that may influence the performance such as the learning rate, i.e., the step length in each update, the weight parameter λ, and the stop criterion. Those parameters will be introduced in detail in each experiment.

4.2. DANN for Image Classification

Image classification is an important application in pattern recognition. It has been greatly developed with the deep neural networks. Therefore, we test our DANN by image classification. MNIST and CIFAR-10 datasets are used in this experiment. For MNIST dataset, the full connecting and convolutional architectures are used. We compare the proposed DANN with feed-forward neural networks on different architectures, i.e., full connecting and convolution, and the dense associative memory (DAM) [25]. DAM is a type of associative memory model which stores and reliably retrieves many more patterns than the number of neurons in the network.

4.2.1. Classification Results on MNIST dataset

To better test the performance of DANN, we use 100, 1000, 10000, and 60000 training samples respectively to train the networks. The full connecting architecture is first evaluated. We set the architecture to be "794-500-300-200-200" for DANN and "784-500-300-200-10" for an end-to-end feed-forward neural network. The learning rate of updating network parameters and reconstruction is 0.01 and 1, respectively. Since the image is small and simple, the trick of neighborhood prior is not used in this experiment. After trained by 100, 1000, 10000, and 60000 training samples, respectively, the classification training and test errors of the three methods on the 10000 test data are shown in Table 1. For convenience, we use the same network architecture for different number of training samples. With consideration of overfitting or underfitting, training error is also exhibited. For fair comparison, we compare the test error by controlling the training errors to be similar to each other.

From the classification results, we can find that with less training samples, DANN achieves lower accuracy than the compared feed-froward neural network. For instance, with 100 training samples, all the three methods achieve the maximum training accuracy, however, the test error of DANN is much larger than that of feed-forward network. But with more training

Table 1: Training and test errors (%) of different methods on the MNIST dataset

Methods		100 samples		1000 samples		10000 samples		60000 samples	
		train	test	train	test	train	test	train	test
DAM		0	45.17	0.40	18.06	0.61	5.01	0.29	3.31
Full Connecting	feed-forward	0	26.63	0.20	11.14	0.22	4.23	0.28	2.81
	DANN	0	39.97	0.50	14.89	0.23	3.03	0.28	2.48
Convolution	feed-forward	3.00	21.36	8.40	8.51	2.33	2.54	0.14	1.98
	DANN	2.00	25.04	7.65	8.22	1.65	2.17	0.13	1.21

samples, i.e., over 10000 samples, DANN begin to outperform the feed-forward network. With all of the training samples, DANN can achieve the best performance among the three methods. It deserves to be noted that sometimes the test error is even lower than the training error during the learning process of DANN, for instance, when DANN achieves the training error of 17.43%, the test error is 16.99%. Such a phenomenon indicates that with less training samples, the proposed probability model cannot capture the observed space well and with a new sample that the model have not ever encountered, the model is failed to accurately recover the missed elements. However, with numerous training samples, the model is able to estimate the whole observed space with higher accuracy. In general, with enough training samples, the proposed DANN is able to achieve comparable results compared with the widely used feed-forward neural network.

The full connecting network is sometimes clumsy to deal with images because of the huge input dimension. Convolutional neural network is designed to overcome this defect by convolutional kernels. Since the top perception layer is the processing core of DANN, it can be easily reconstructed by convolutional layers. Therefore, we compare the convolutional DANN with CNN which is suitable to deal with images. The basic architecture we use here is the LeNet [9] and we set the perception layer being 50 units following the full connecting layer with 84 units. The learning rate of perception layer, propagation layers, and reconstruction process is 0.01, 0.1, and 1 respectively. The classification performance comparison between DANN and feed-forward neural network with the convolutional architecture is shown in Table 1. It is obvious that the convolutional architecture achieves much better performance than the full connecting architecture. The same phenomenon that DANN achieves lower performance with less training samples can also be found in the experiments on the convolutional architecture. With 100 training samples, DANN achieves lower training error while the test error is much larger. However, DANN is able to adapt to the convolutional architecture well when there are enough training samples. As a consequence, DANN has the ability of recovering correct labels which can be used as classifiers.

4.2.2. Classification Results on CIFAR-10 dataset

A more complex dataset is also used here, i.e., the CIFAR-10 dataset which is a challenge for image classifiers. Since the data is much more complex than that in MNIST dataset, the convolutional architecture is used. There are two convolutional layers with each layer followed by a pooling layer and two full connecting layers in the architecture. The first convolutional layer has 64 5×5 kernels and the second convolutional layer has 64 5×5

Table 2: Training and test errors (%) of different methods on the CIFAR-10 dataset

Methods	100 samples		1000 samples		10000 samples		50000 samples	
	train	test	train	test	train	test	train	test
DAM	40.0	41.1	32.2	33.8	28.3	29.0	22.6	24.8
feed-forward	37.0	37.3	18.8	22.2	13.4	19.1	14.1	18.6
DANN	37.0	37.8	25.5	27.1	13.5	18.6	11.3	16.3

kernels. The first full connecting layer has 384 units and the second full connecting layer has 192 units. For the feed-forward neural network, the last layer is the output layer with 10 units. For DANN, the last layer is the perception layer with 150 units. The learning rate of perception layer, propagation layers, and reconstruction process is 0.01, 0.1, and 1 respectively and $\lambda = 0.01$. Similarly, 100, 1000, 10000, and 50000 training samples are used respectively to evaluate the performance of DAM, feed-forward neural network, and the proposed DANN. The training and test classification errors of them are shown in Table 2. With the more complex data, DAM achieves much larger errors because there are less neurons and connections to store and processing the whole data space. With the deep architecture, unsupervised learning models is able to represent whole observed space by enough training samples. Therefore, DANN outperforms the convolutional neural network with the same propagating architecture when trained by enough samples.

In conclusion, experiments on MNIST and CIFAR-10 datasets demonstrate the effectiveness of the proposed DANN on the classification task. As introduced above, a unique and interesting function of associative memory is image recovery. Next we test the DANN on the task of image recovery.

4.3. DANN for Image Recovery

Image recovery can be achieved by AA memory. First the MNIST dataset is used, and we compare the proposed DANN with DAM [25] and the hierarchical self-organizing associative memory (HSAM) [23] which is a probability based one. In theory, DAM and HSAM can only recover the images by which they are trained. They have no ability to draw inferences. DAM stores the patterns and with a corrupted data, it recovers the most similar pattern. HSAM is composed of hierarchical processing elements, but each processing element only stores the probability of four status of two input units. Therefore for MNIST dataset, we recover the manually corrupted images in the test set after trained by training images which is different from previous work [23].

4.3.1. Recovering Digits with Labels

In image classification, the classifier can tell you what it is given an image. However, a smarter learning machine can also depict the image when told what the image is. Associative memory is convenient to achieve this task by taking the image as the missing elements. So in this experiment, we recover the whole image by a label after trained by the training set. The network parameters trained in the experiments on image classification can also be used to recover the images. Figure 6 shows the 10 digits recovered from 10 labels of each method. HSAM and DANN recover all the digits but HSAM introduces more noise because some

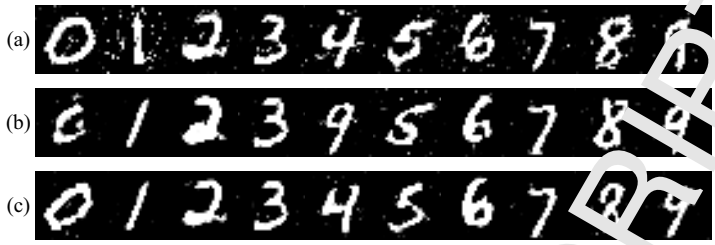


Figure 6: Recovered digits from 10 labels by different methods. (a) HSAM (b) DAM (c) DANN

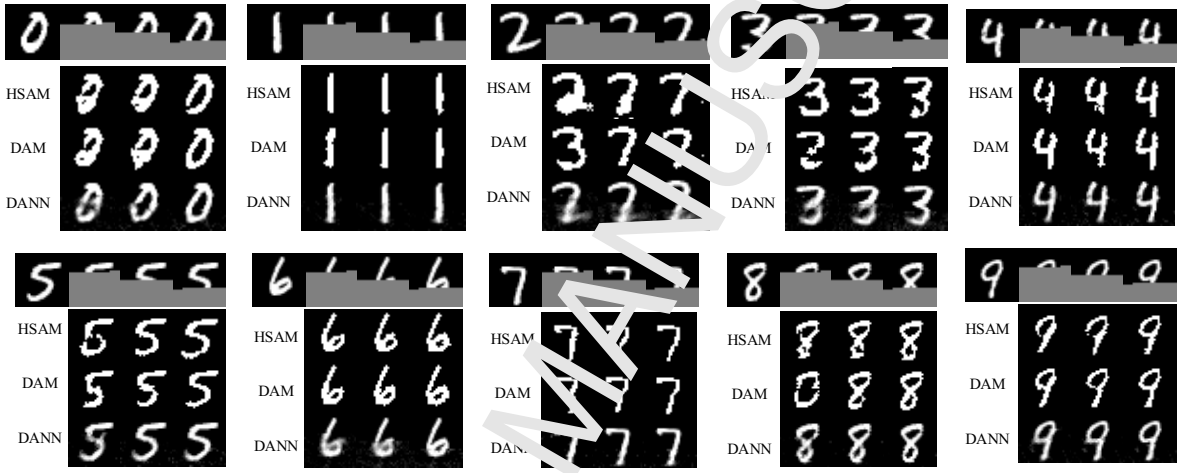
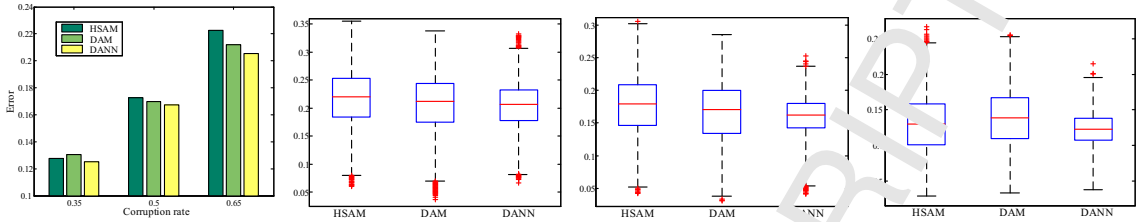


Figure 7: Recovered test images by the three methods. In each image group, the first line shows the original image and the corrupted images by different corruption rates, i.e., 65%, 50%, and 35%. The rest images are the images recovered from the corresponding corrupted images by HSAM, DAM, and DANN, respectively.

processing elements may not well estimate the probability. DAM wrongly recover 4 as 9 because these two digits are similar and DAM may recover the wrong pattern when the units are updated iteratively. DANN recovers the 10 digits with the best performance because the deep architecture can well represent the digits after trained by all the 60000 images. This experiment demonstrates that DANN is able to depict a digit when told what the digit is after learning. Such an interesting ability cannot be achieved by "end-to-end" deep neural networks.

4.3.2. Recovering Corrupted Digits

Another interesting capability of associative memory is to recover the whole image by providing part of the image which also cannot be achieved by "end-to-end" networks. DANN is trained by 60000 training images with the same full connecting architecture used in image classification. Two methods HSAM and DAM are used for comparison. Figure 7 shows some reconstructed test images obtained by the three methods. We corrupt the test images by 35%, 50%, and 65% respectively as shown in Figure 7 followed by the recovered images from the corresponding corrupted images. When the corruption rate is 65%, it is difficult to recognize what the original image is even by human. Take the digit 2 for example, it is more likely to



(a) (b) (c) (d)

Figure 8: Test recovery errors. (a) Average errors achieved by HSAM, DAM, and DANN with corruption rates of 35%, 50%, and 65%. (b) Box plots on test errors obtained by HSAM, DAM, and DANN when the corruption rate is 65%. (c) Box plots on test errors obtained by HSAM, DAM, and DANN when the corruption rate is 50%. (d) Box plots on test errors obtained by HSAM, DAM, and DANN when the corruption rate is 35%.

be recognized as 7 even only 35% images are corrupted. Therefore, DAM first recognizes it as 3 and recovers an image of 3. With more information, it cannot be a 3 and DAM recovers a 7. At last, the most important part of the digit 2, i.e., the tail, is still hidden. Then DAM fails to recover the correct digit. HSAM first correctly recognize the digit. But with more clew, it changes its result regrettably. The corrupted images is really like 7 and it cheated DAM and HSAM successfully. DANN is for continuous data and it is a probability model. Therefore the recovered part is between 0 and 1. DANN is also not sure what the digit is and the tail is ambiguous. DAM also recovers the wrong digits when the corrupted digits are 3 and 8. From this figure, DANN performs well on the digit image recovery with different corruption rates. Then we implement on all the test digits and compute the recovery error RE for each test digit $RE = \|x_{rec} - x\|_2^2$ where x_{rec} denotes the recovered digit. The average recovery errors are shown in bar graph in Figure 8 (a). It is obvious that DANN achieves the best performance even with different corruption rates. To better evaluate the results, we draw the box plots for the errors of the test set in Figure 8 (b), (c), and (d) with the corruption rates of 65%, 50%, and 35% respectively. Since DANN achieves ambiguous digits as shown in Figure 7, it is difficult to achieve exactly correct digits. Therefore the bottom of the box is a little higher than that of HSAM and DAM. However, HSAM and DAM take the risk of recover totally wrong digits. So the top of the boxes is much higher than that of DANN. This experiment demonstrates the effectiveness of DANN for recovering digit images. Next we attempt to recover color images by DANN which cannot be achieved by HSAM and DAM because the input units of them are defined to be binary.

4.3.3. Recovering Color Images

This experiment is implemented on the CIFAR-10 dataset. The convolutional AA architecture which is the same as that in image classification is used and trained by all the images in the training set. Then we recover the trained images from images corrupted by Gaussian noise, salt&pepper noise, and blank patch. Figure 9, 10, and 11 respectively shows some recovered images from these corrupted images. In each figure, the first column contains the corrupted images and then the recovered images are list in the second column. The original images and the visualized recovery errors are listed in the third and fourth columns. The visualization of the errors is to exhibit the error distribution of the recovered images.

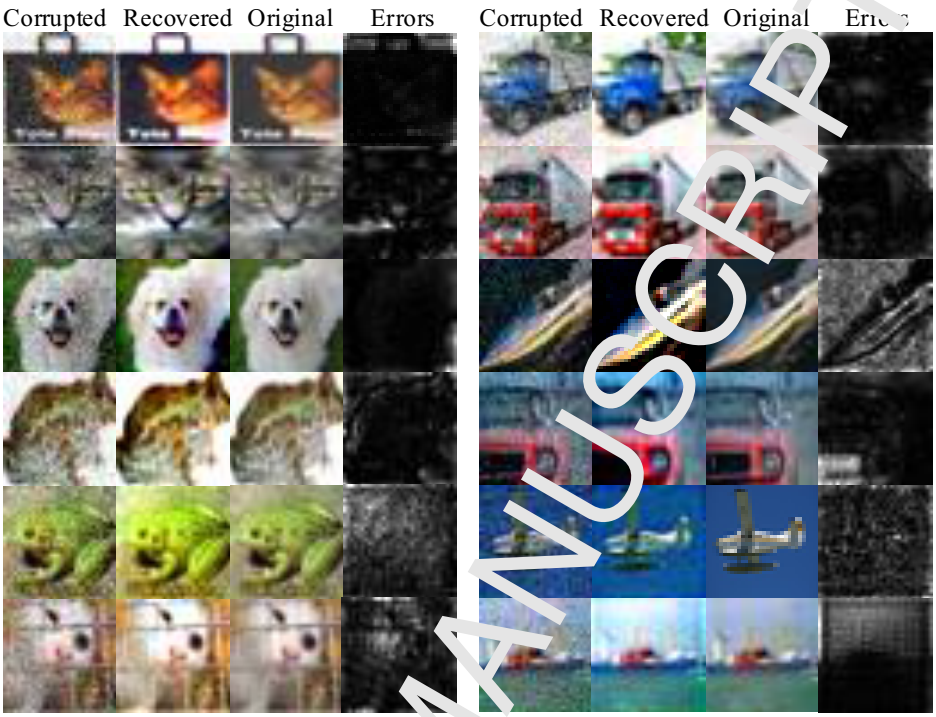


Figure 9: Image recovery from Gaussian noise corrupted CIFAR-10 images.

In Figure 9, the gaussian noise blurs the images but DANN recovers most details of original images. For example, the motif on the frogs is destroyed by noise, but DANN clearly recovers the motif. Moreover, DANN even increases the image contrast. Then in Figure 10, most of the images are corrupted by noise. DANN can still recover the correct images. Some recovered images are even the same to the original images, such as the gray cat, blue truck, and speedboat. Last we recover the images from partly corrupted images. The image details will be recovered from part of images. DANN well recovers the patch corrupted images. Experiments on the three types of corruptions demonstrate the effectiveness of DANN for recovering color images which cannot be achieved by most associative memory models because they are usually defined based on binary input. However, since in DANN, the images are recovered from a probabilistic model, the regions with pure color are not well painted with exactly the same color as shown in the visualization of errors.

5. Concluding Remarks

A deep associative neural network (DANN) for associative memory has been proposed. It mimics the associative learning process of human and both data and label are input of the network. A hierarchical architecture consisting of a perception layer and several propagation layers is constructed. To learn the network parameters, a probabilistic model inspired from unsupervised representation learning is defined. The single layer unsupervised learning model is extended into a deep one and a learning procedure is designed to optimize this model. Contrastive divergence algorithm is used and a novel sampling method is proposed to sample

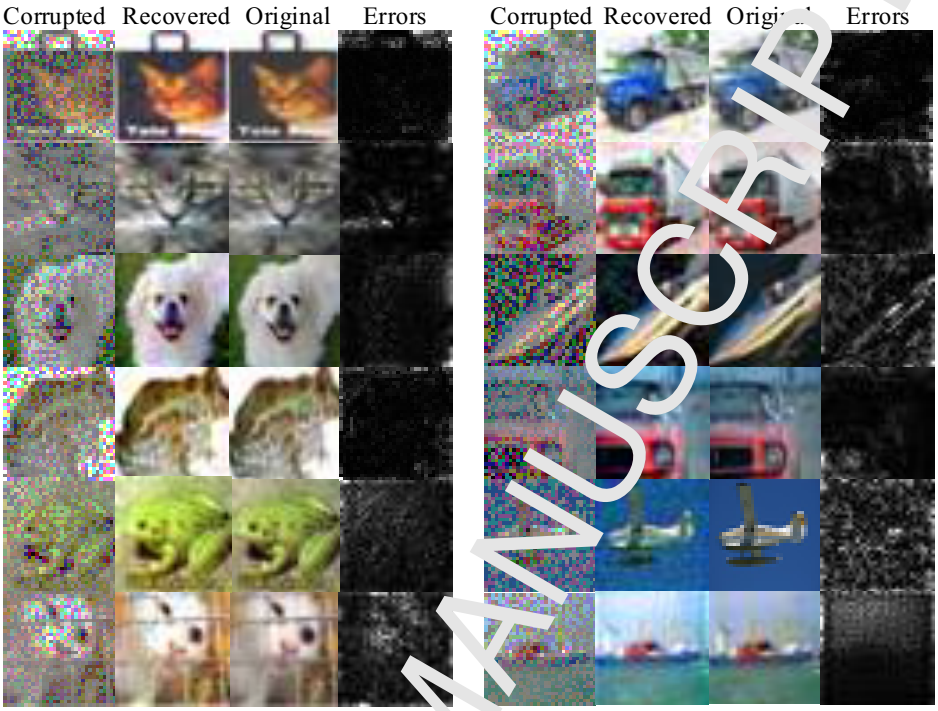


Figure 10: Image recovery from salt&pepper noise corrupted CIFAR-10 images.

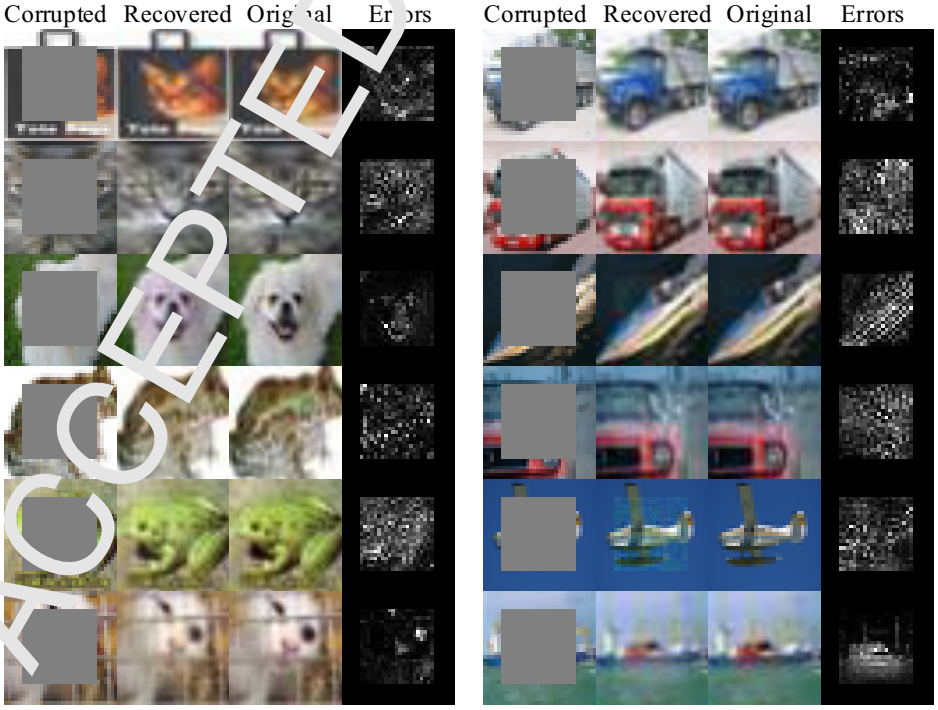


Figure 11: Image recovery from blank patch corrupted CIFAR-10 images.

from the deep network. Finally, some learning tricks are used to further increase the learning efficiency. Compared with existing associative memory models, DANN is superior due to the propagation layers which learns abstract features for better processing the core information. Therefore, DANN is able to deal with much more complex data such as color images which is difficult for many associative memory models because the neurons or processing elements in them are defined in binary. Experiments are implemented for evaluating image classification and image recovery which demonstrate the effectiveness of the DANN.

However, due to the fantasy data in the denominator of the probabilistic model, it is difficult to optimize the model ideally. Therefore, in the future work, we will focus more on the model and learning process and attempt to train the parameters of DANN more efficiently. Moreover, we will try other applications such as image inpainting, nature language processing, and video retrieval by DANN.

References

[1] I. Arel, D. C. Rose, T. P. Karnowski, Deep machine learning-a new frontier in artificial intelligence research [research frontier], IEEE Computational Intelligence Magazine 5 (4) (2010) 13–18.

[2] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[3] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[4] J. Liu, M. Gong, K. Qin, P. Zhang, A deep convolutional coupling network for change detection based on heterogeneous optical and radar images, IEEE Transactions on Neural Networks and Learning Systems 29 (3) (2018) 545–559.

[5] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[6] A. A. Awan, K. Hamidouche, J. M. Hashmi, D. K. Panda, S-Caffe: Co-designing MPI runtimes and Caffe for scalable deep learning on modern GPU clusters, in: Proceedings of the ACM Signal Symposium on Principles and Practice of Parallel Programming, 2017, pp. 193–205.

[7] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18 (7) (2006) 1527–1554.

[8] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[9] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[10] F. A. Gers, N. N. Schraudolph, J. Schmidhuber, Learning precise timing with lstm recurrent networks, Journal of Machine Learning Research 3 (8) (2002) 115–143.

[11] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (8) (2013) 1798–1828.

[12] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Proceedings of Advances in Neural Information Processing Systems, 2007, pp. 153–160.

[13] Y.-l. Boureau, Y. L. Cun, et al., Sparse feature learning for deep belief networks, in: Proceedings of Advances in Neural Information Processing Systems, 2008, pp. 1185–1192.

[14] H. Lee, C. Ekanadham, A. Y. Ng, Sparse deep belief net model for visual area v2, in: Proceedings of Advances in Neural Information Processing Systems, 2008, pp. 873–880.

[15] N.-N. Ji, J.-S. Zhang, C.-X. Zhang, A sparse-response deep belief network based on rate distortion theory, Pattern Recognition 47 (9) (2014) 3179–3191.

[16] M. Gong, J. Liu, H. Li, Q. Cai, L. Su, A multiobjective sparse feature learning model for deep neural networks, IEEE Transactions on Neural Networks and Learning Systems 26 (12) (2015) 3263–3277.

[17] J. Liu, M. Gong, Q. Miao, Neuron learning machine for representation learning, in: Proceedings of AAAI Conference on Artificial Intelligence, 2017.

[18] J. Liu, M. Gong, Q. Miao, Modeling hebb learning rule for unsupervised learning, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2017, pp. 2315–2321.

[19] G. Palm, Neural associative memories and sparse coding, Neural Networks 37 (2013) 165–171.

[20] D. D. Vogel, Auto-associative memory produced by disinhibition in a sparsely connected network, Neural Networks 11 (5) (1998) 897–908.

[21] L. Wang, Heteroassociations of spatio-temporal sequences with the bidirectional associative memory, IEEE Transactions on Neural Networks 11 (6) (2000) 1503–1505.

[22] J. A. Starzyk, H. He, Spatio–temporal memories for machine learning: A long-term memory organization, IEEE Transactions on Neural Networks 20 (5) (2009) 768–780.

[23] J. A. Starzyk, H. He, Y. Li, A hierarchical self-organizing associative memory for machine learning, in: Proceedings of the International Symposium on Neural Networks, Springer, 2007, pp. 413–423.

[24] S. Hu, Y. Liu, Z. Liu, T. Chen, J. Wang, Q. Yu, L. Deng, Y. Yin, S. Hosaka, Associative memory realized by a reconfigurable memristive hopfield neural network, Nature Communications 6 (2015) 7522.

[25] D. Krotov, J. J. Hopfield, Dense associative memory for pattern recognition, in: Proceedings of Advances in Neural Information Processing Systems, 2016, pp. 1172–1180.

[26] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences of the United States of America 79 (8) (1982) 2554.

[27] E. Kuriscak, P. Marsalek, J. Stroffek, P. G. Toth, Biological context of hebb learning in artificial neural networks, a review, Neurocomputing 152 (2015) 27–35.

[28] G. E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Computation 14 (8) (2002) 1771–1800.

[29] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, Journal of Machine Learning Research 11 (Dec) (2010) 3371–3408.

[30] M. Kobayashi, Decomposition of rotor Hopfield neural networks using complex numbers, IEEE Transactions on Neural Networks and Learning Systems 29 (4) (2018) 1366–1370.

[31] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.

[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[34] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in: Proceedings of Advances in neural information processing systems, 2015, pp. 2377–2385.

Highlights

> We propose a deep associative neural network for associative memory.

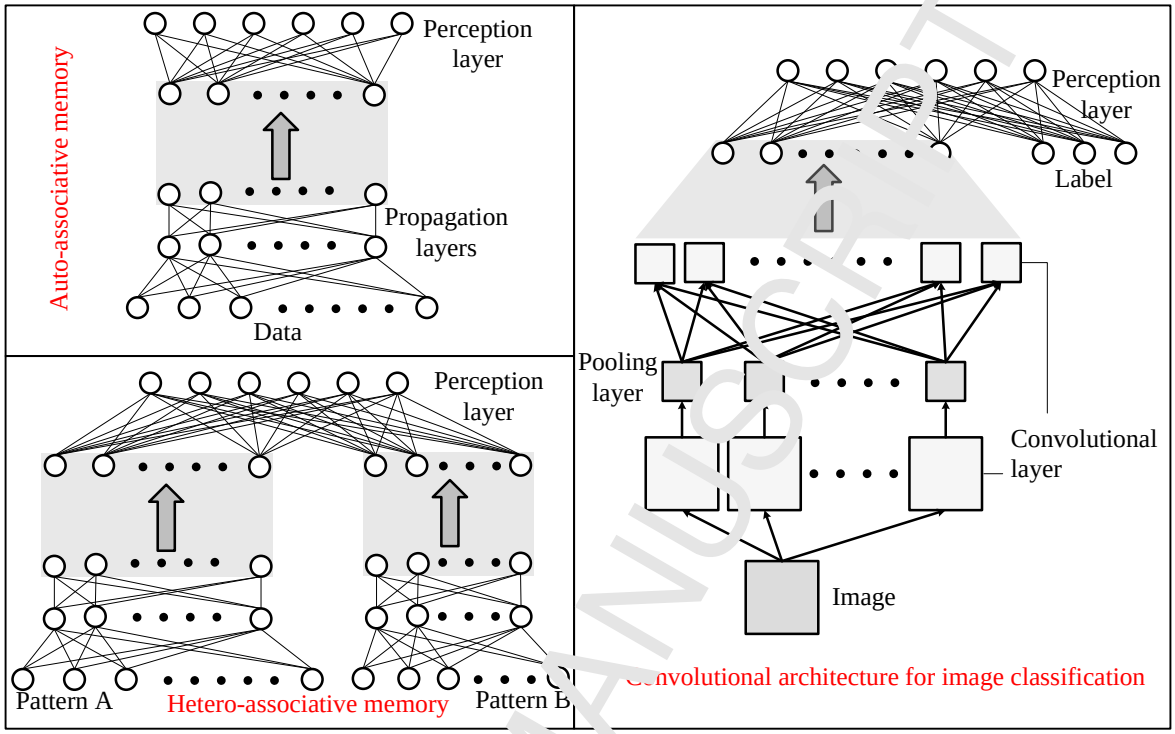
> The network parameters are learnt by optimizing the log-likelihood of a probability distribution.

> The probability model is established to make the network best capture the distributions of the observed data according to unsupervised representation learning.

> The contrastive divergence algorithm is improved with a novel iterated sampling method to optimize the proposed model.

> The proposed network can achieve many difficult pattern recognition tasks, including image classification and image recovery from labels and corrupted images

Graphical abstract



The architectures of deep associative neural network on different application scenarios.