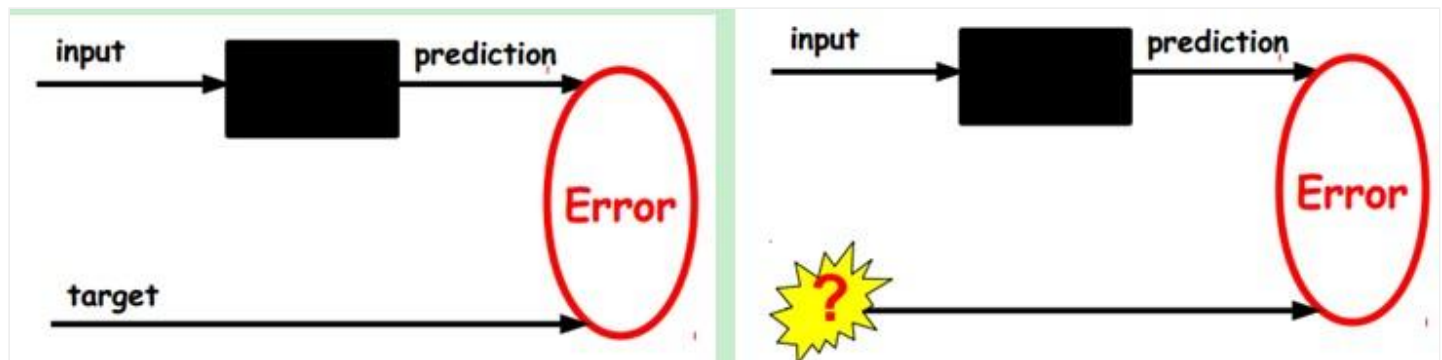


## AutoEncoder autocoder

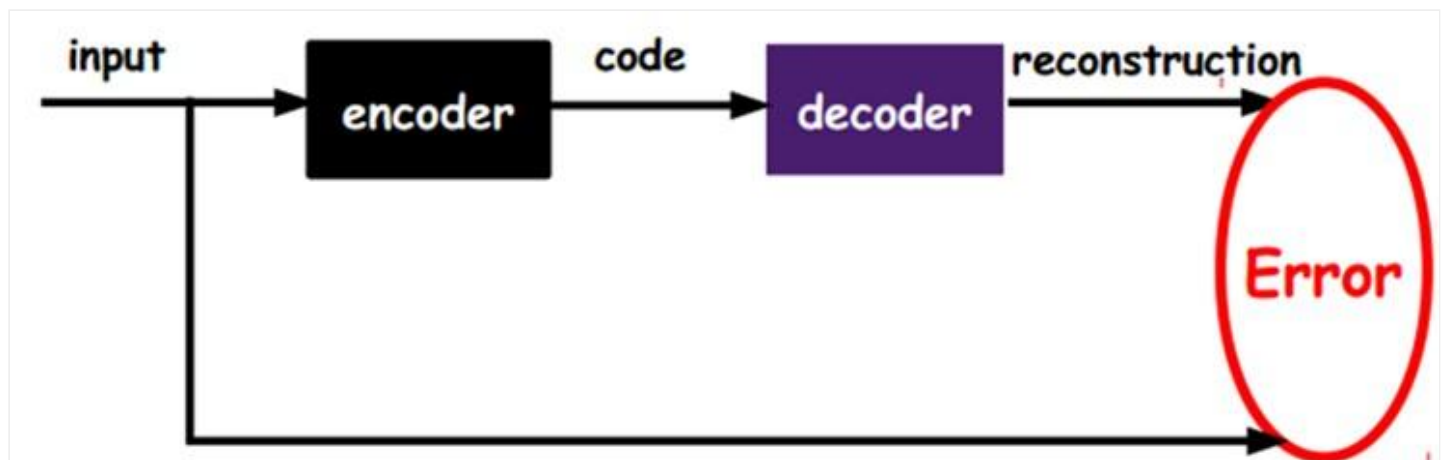
The simplest is to use a method Deep Learning characteristics of artificial neural networks, artificial neural network (ANN) having a system hierarchy itself, if given a neural network, we assume that the input and output are the same, then adjust the training its parameters, to obtain the weights in each layer. Naturally, we get the input I represent several different (each layer representing a representation), these representations is characteristic. Autoencoder neural network is a kind of reproducing the input signal as possible. In order to achieve this reproduction, the encoder must automatically capture can represent the most important factor of the input data, like the PCA as the main component can be found on behalf of the original information.

Specific process briefly described as follows:

### 1) Given unlabeled data using unsupervised learning learning feature:

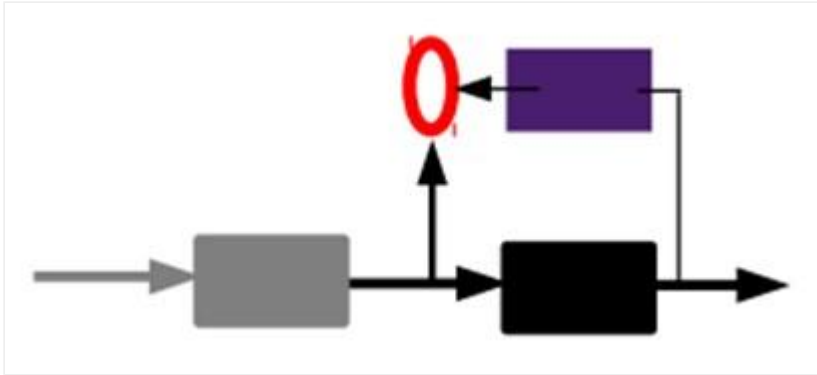


In our previous neural network, such as the first figure, we entered sample is tagged, i.e. (input, target), the layers that we have to change the difference between the previous and the current output target (label) parameters until convergence. But now we have only unlabeled data, which is the view on the right. So how do get this error?



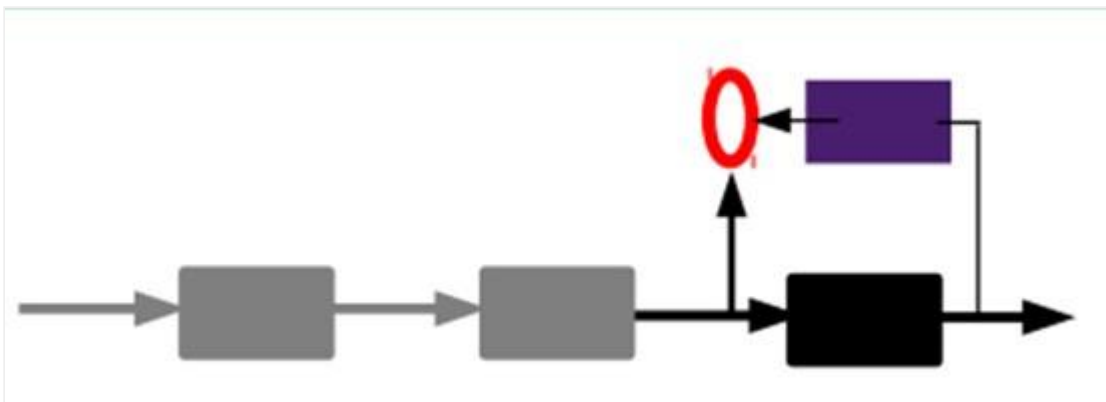
As shown above, we will enter an encoder input encoder, you get a code, enter the code that is a representation, then how do we know this is the input code represent it? We add a decoder decoder, this time the decoder will output a message, then the message if the

output of the input signal and a start input is much like (the ideal situation is the same), it is very clear that we have reason to believe this code is tricky. Therefore, we adjust the parameters and the decoder of the encoder, such that the minimum reconstruction error, this time we get a first input signal input indicates, the code is encoded. Because no label data, the source of error is obtained directly after the reconstruction compared to the original input.



## 2) generated by the encoder characteristics, and training the next layer. In this way layer by layer Training:

That we get above the first layer of code, our reconstruction error is minimized to convince us that this code is the original input signal is well expressed, or far-fetched point that it is exactly the same and the original signal (the expression is not the same, reflecting the It is a thing). That second layer and the training mode of the first layer is no difference, the output will be the first layer of the code as an input signal of the second layer, the same reconstruction error is minimized, it will obtain the parameters of the second layer, and the resulting the second layer code input, i.e. the second original input information expressed. Other layers on the same way cooked on the line (training this layer, the front layer parameters are fixed, and their decoder has been useless, are no longer necessary).



## 3) supervised fine-tuning:

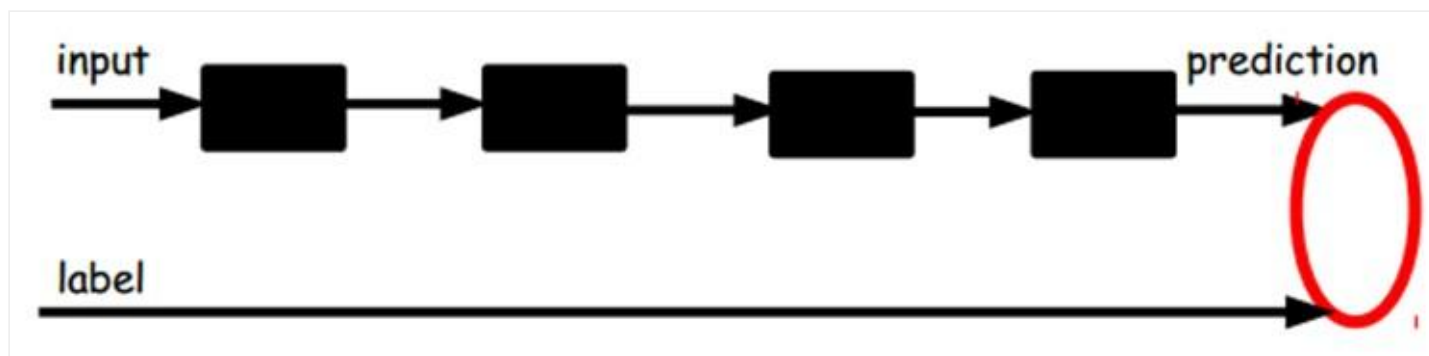
After the above method, we can get a lot of layers. As for how many layers (or how much depth, this itself is not at present a scientific evaluation methods) needs its own test tune. Each layer will get a different original input expression. Of course, we feel it is more abstract, the better, just as the human visual system.

Here, this AutoEncoder can not be used to classify the data, because it has not learned how to link one input and one class. It's just learned how to reconstruct or reproduce its input only. Or, it's just learning to get a good representative of the characteristics that can be entered, this feature can represent the original input signal to the maximum extent. So, in order to achieve the classification, we can add at most top AutoEncoder coding layer of a classifier (eg Roger Lancaster regression, SVM, etc.), then supervised training method by standard multilayer neural networks (gradient descent) to train .

In other words, at this time, we need to feature the final layer of code input to the final classifier by label samples, fine-tuning through supervised learning, which can be divided into two types, one is the only adjustment classifier (black section):



Another: The label samples, fine-tuning the system :( If there is enough data, this is the best .end-to-end learning end-learning)



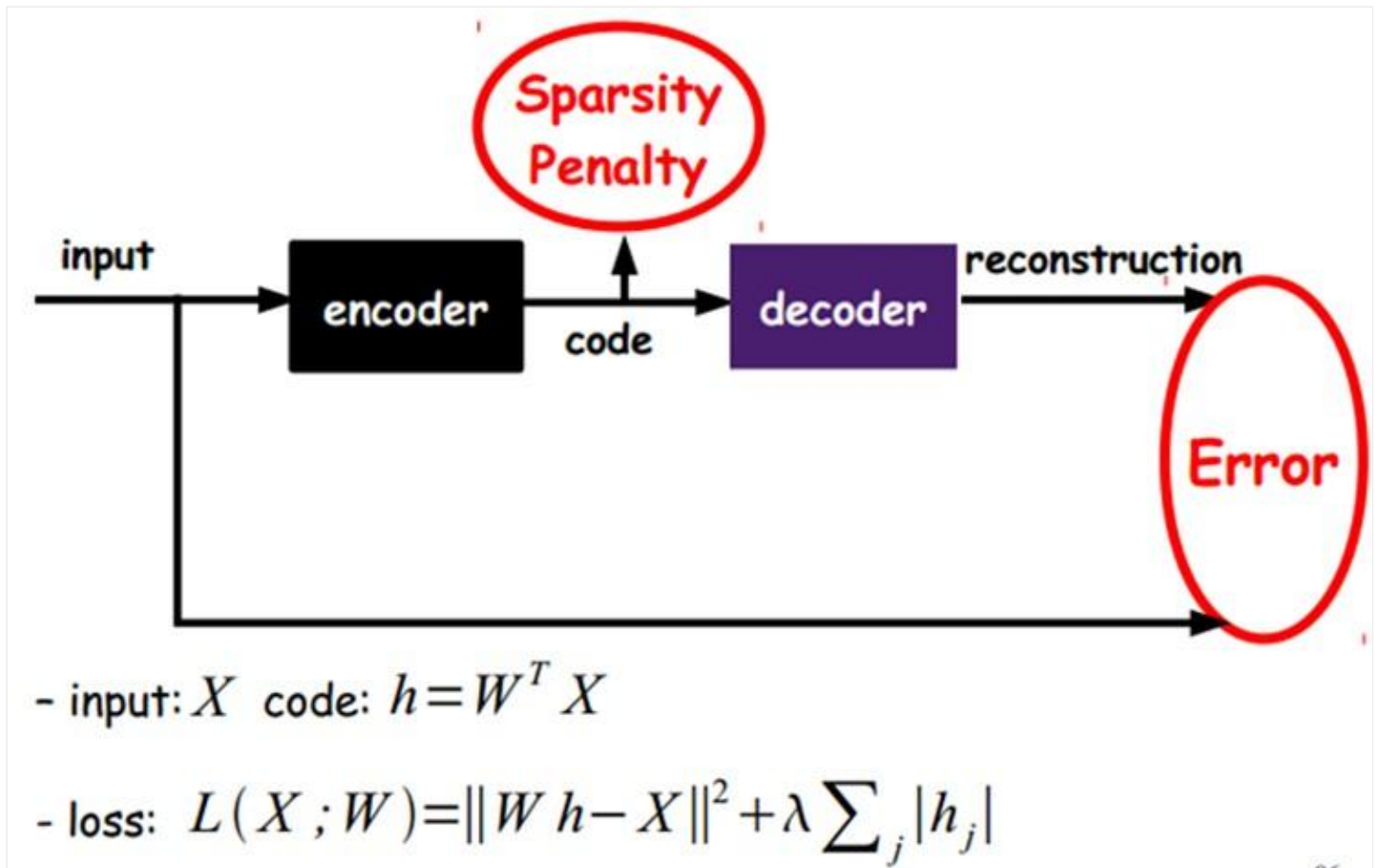
Once the training is completed supervision, this can be used to classify the network. Topmost neural networks can be used as a linear classifier, then we can use a classifier with better performance to replace it.

Can be found in the study, if the addition of these features automatic learning obtained in the original feature can greatly improve accuracy, even in the classification problem even better than the current best classification algorithm results!

AutoEncoder some variants, the two introduced briefly here:

### **Sparse AutoEncoder sparse automatic encoder:**

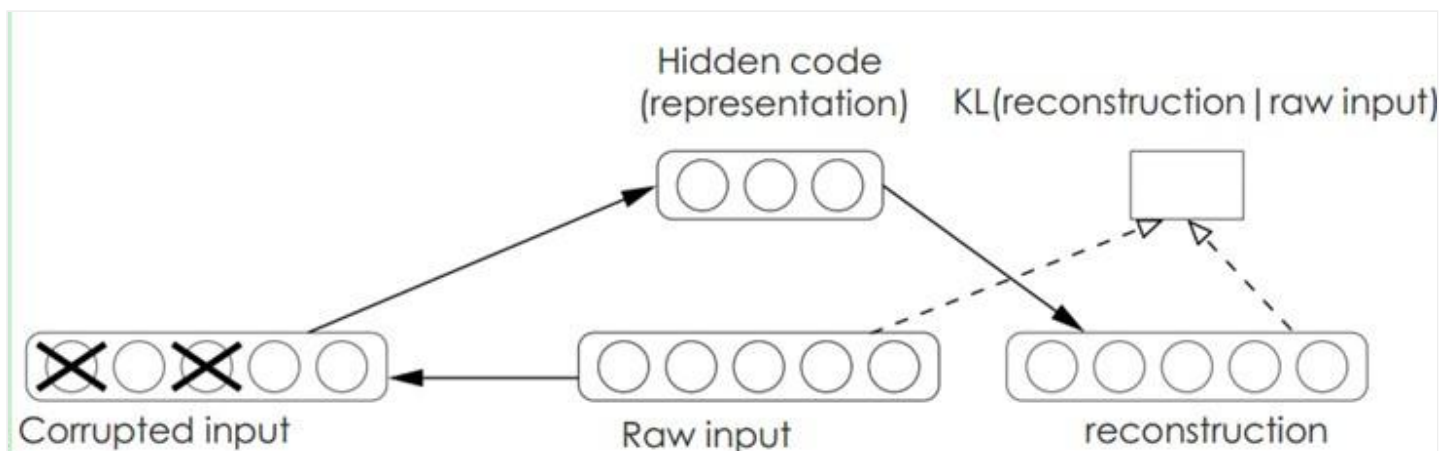
Of course, we can continue to add some constraints to get the new Deep Learning method, such as: if Plus L1 of Regularity restrictions on the basis of AutoEncoder (L1 main constraint nodes in each layer should be mostly 0 only a small number is not zero, which is the source Sparse name), we can get Sparse AutoEncoder law.



As shown above, in fact, express code to limit as much as possible every time to get sparse. Because the sparse expression tend to be more effective than other expression (like the human brain is the case, an input only stimulate certain neurons, most other neurons are inhibited in).

### Denoising AutoEncoders noise autocoder:

Automatic noise reduction encoder DA is based on the auto-encoder, the training data added noise, the auto-encoder must learn to remove this noise and get real noise pollution had not been entered. Therefore, forcing the encoder to learn more robust expression of the input signal, which is its generalization ability than the average encoder reasons. DA may gradient descent algorithm go through training.



## Sparse Coding sparse coding

If we put the concept of input and output must be equal to the constraint relaxation, while using linear algebra group, i.e.,  $O = a_1 * \Phi_1 + a_2 * \Phi_2 + \dots + a_n * \Phi_n$ ,  $\Phi_i$  is a group,  $a_i$  is a factor, we can get such an optimization problem:

$$\text{Min } \|I - O\|, \text{ where } I \text{ represents the input, } O \text{ for output.}$$

By solving the optimization equation, we can find the coefficients of  $a_i$  And base  $\Phi_i$ , Another of these approximate expressions is input and the base of the coefficients.

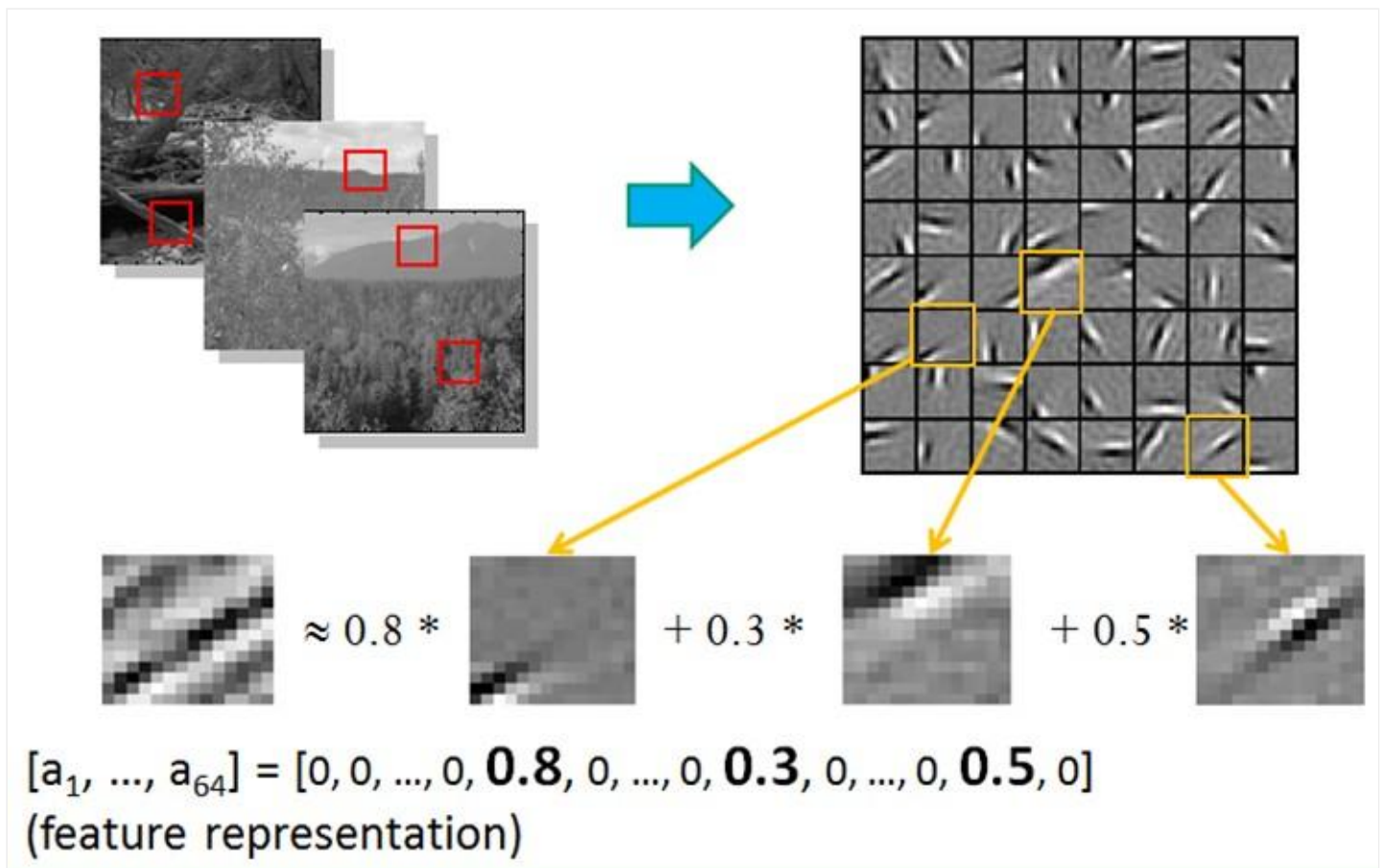
$$x = \sum_{i=1}^k a_i \phi_i$$

Thus, they can be used to express the input  $I$ , the learning process is automatically obtained. If we add Regularity limit  $L1$  in the above equation to obtain:

$$\text{Min } \|I - O\| + u * (|a_1| + |a_2| + \dots + |a_n|)$$

This method is called Sparse Coding. In layman's terms, it is a signal represented as a linear combination of a set of groups, and may require only a relatively small number signal group can be represented. "Sparsity" is defined as: only a few non-zero elements or only a few elements is much greater than zero. It requires a coefficient; Sparse meant to say: For a set of input vectors, we just want to have as little as a few much larger than the coefficient of zero. Use of sparse component selected to represent the input data is our reason, because the vast majority of sensory data, such as a natural image can be represented as a superposition of a small number of basic elements, these basic elements in the image plane or may be line. At the same time, such as the primary visual cortex analogy process thus been improved (the human brain has a large number of neurons, but for some image or only a small edge of neuronal excitability, others are in a state of inhibition).

Sparse coding algorithm is an unsupervised learning method, which is used to find a set of "over-complete" basis vectors more efficiently represent the data samples. Although the shape such as principal component analysis (PCA) enables us to easily find a set of "comprehensive" basis vectors, but here we want to do is find a group of "super complete" basis vectors to represent the input vector (that is, the number of basis vectors is larger than the dimension of the input vector). Benefits Overcomplete groups is that they can more effectively identify implicit in the structure and internal data input mode. However, for a complete super group, the coefficients  $a_i$  It is no longer uniquely determined by the input vector. Therefore, in the sparse coding algorithm, plus we have a criterion "sparsity" to solve degeneration (degeneracy) due to over-complete problems caused. ([Please refer to the detailed process: UFLDL Tutorial sparse coding](https://www.programmersought.com/article/56572394201/))



For example, in the bottom do Edge Detector generation of Feature Extraction of the image, so the work here is from the Natural Images in randomly selected small patch, the patch can be generated by describing their "base" that is the right of the  $8 * 8 = 64$  basis consisting basis, and then given a test patch, we can obtain a linear combination of basis according to the above formula, is a sparse matrix and, in the case of FIG. 64 have a dimension, wherein the non-zero entries only three, so called "sparse".

Here it may be in doubt, as the underlying Edge Detector why it? What is it the top? Here everyone will understand to be a simple explanation, the reason is because Edge Edge Detector in different directions will be able to describe the whole image, so naturally it is the Edge in different directions image ..... and on the basis of the basis layer combinations as a result, the upper layer is a layer combination basis ..... (that is, part of the top of the fourth when we put it)

Sparse coding is divided into two parts:

**1) Training phases:** Given a set of sample images  $[x_1, x_2, \dots]$ , we need to learn to get a set of base  $[\Phi_1, \Phi_2, \dots]$ , which is a dictionary.

Sparse coding is a variant of k-means algorithm, the training process is almost (EM algorithm idea: If you want to optimize the objective function of two variables, such as  $L(W, B)$ , then we can first fixed  $W$ ,  $B$  adjustment  $L$  makes minimum, and then fixed  $B$ ,  $W$  to adjust the minimum  $L$ , so iterative alternately, will continue to push the minimum  $L$ . EM algorithm can see my blog: "[From the maximum likelihood algorithm to shallow EM Solutions](https://www.programmersought.com/article/56572394201/)") .



Training process is a process of repeated iteration, according to the above mentioned, we alternate a change and  $\Phi$  minimizes the following objective function.

$$\min_{a, \phi} \sum_{i=1}^m \left\| x_i - \sum_{j=1}^k a_{i,j} \phi_j \right\|^2 + \lambda \sum_{i=1}^m \sum_{j=1}^k |a_{i,j}|$$

Each iteration of two steps:

a) Fixed Dictionary  $\Phi$  [k], then adjust  $a$  [k], such that (i.e., solutions of the problem LASSO) formula, i.e., the minimum objective function.

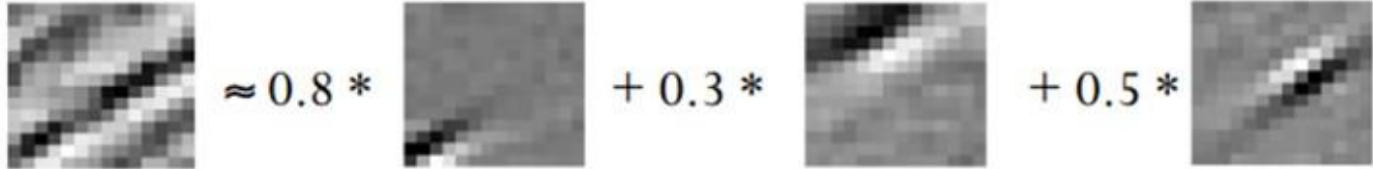
b) subsequently fixed  $a$  [k], adjusting  $\Phi$  [k], such that the above equation, i.e., the minimum objective function (i.e. CONVEX QP problem).

Continue iteration until convergence. In this way you can get a good base set can represent this series of  $x$ , which is a dictionary.

**2) Coding phase:** Given a new picture  $x$ , obtained from the above dictionary, obtained by a sparse vector LASSO problem solution  $a$ . The sparse vector is the input of a sparse vector  $x$  expressed.

$$\min_a \sum_{i=1}^m \left\| x_i - \sum_{j=1}^k a_{i,j} \phi_j \right\|^2 + \lambda \sum_{i=1}^m \sum_{j=1}^k |a_{i,j}|$$

E.g:

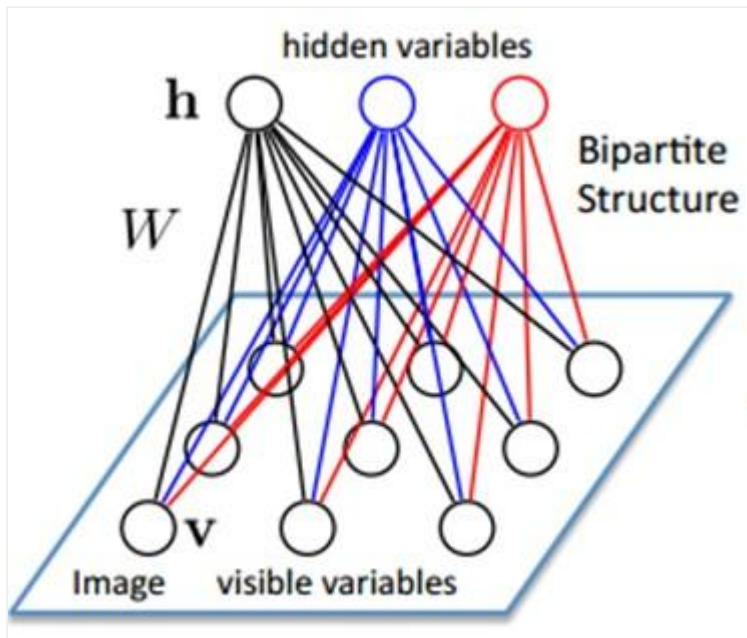


Represent  $x_i$  as:  $a_i = [0, 0, \dots, 0, \mathbf{0.8}, 0, \dots, 0, \mathbf{0.3}, 0, \dots, 0, \mathbf{0.5}, \dots]$

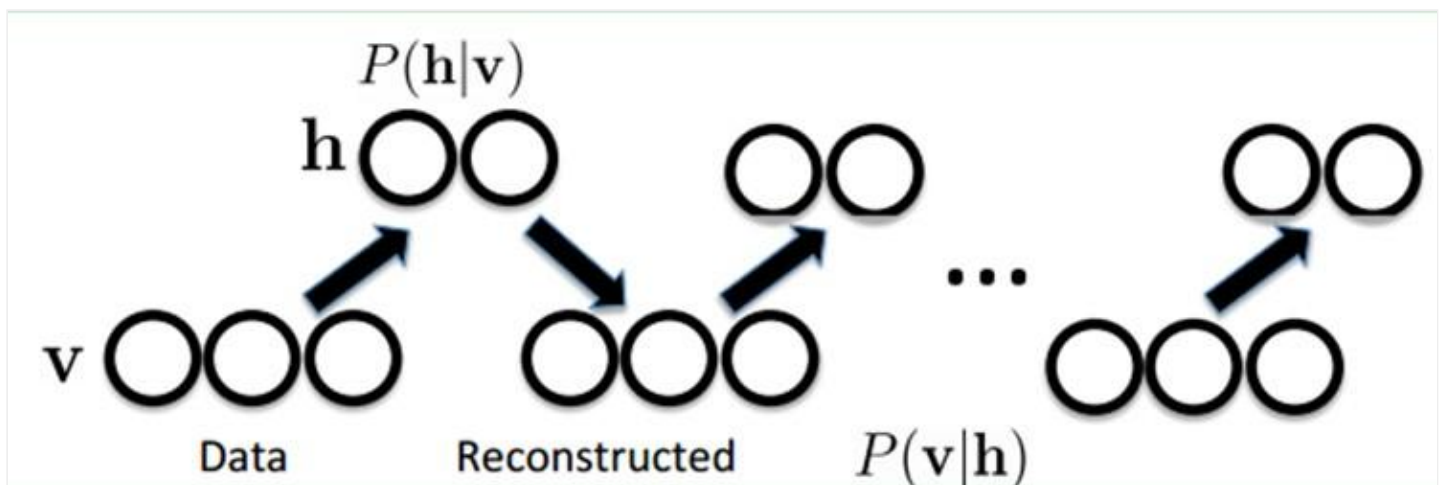
## Restricted Boltzmann Machine (RBM) limit Boltzmann machine

Suppose there is a bipartite graph, there is no link between the nodes in each layer, the layer is visible layer, i.e., the input data layer (v), one hidden layer (H), if it is assumed

that all nodes are random binary value of the variable nodes (only take value 0 or 1), and assuming full probability distribution  $p(v, h)$  meet the Boltzmann distribution, we call this model is Restricted BoltzmannMachine (RBM).

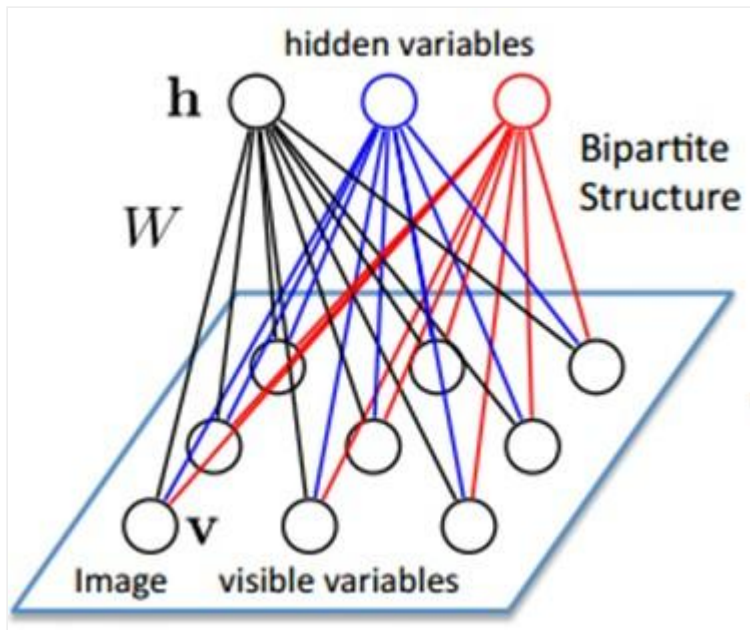


Let us look at why it is Deep Learning method. First, because this model is bipartite, so in the case of  $v$  is known, among all the hidden node independent conditions (because there is no connection between the nodes), i.e.  $p(h | v) = p(h_1 | v) \dots p(h_n | v)$ . Similarly, in the case of the known  $h$  hidden layer, all the nodes are visible independent conditions. You might get the hidden layer  $h | (v, h)$ , to obtain the hidden layer  $h$ , by  $p(v | h)$  at the same time as all of  $v$  and  $h$  satisfy the Boltzmann distribution, therefore, when the input  $v$ ,  $p$  can get through visible layer, by adjusting the parameters, we are such that the visible layer to the original visible layer  $v_1$   $v$  obtained from the hidden layer, if the same, then the hidden layer is obtained expressed otherwise visible layer, hidden layer and therefore can be used as It characterized visible layer of input data, so it is a kind of Deep Learning method.



How to train it? That is right between the visible and hidden layer node node values of how to determine it? We need to do some mathematical analysis. That is the model.





United configuration (joint configuration) energy can be expressed as:

$$E(\mathbf{v}, \mathbf{h}; \theta) = - \sum_{ij} W_{ij} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j$$

$\theta = \{W, a, b\}$  **model parameters.**

The configuration of a joint probability distribution can be determined by Boltzmann distribution (and the energy of this configuration):

$$P_{\theta}(\mathbf{v}, \mathbf{h}) = \frac{1}{Z(\theta)} \exp(-E(\mathbf{v}, \mathbf{h}; \theta)) = \frac{1}{Z(\theta)} \underbrace{\prod_{ij} e^{W_{ij} v_i h_j}}_{\text{partition function}} \underbrace{\prod_i e^{b_i v_i} \prod_j e^{a_j h_j}}_{\text{potential functions}}$$

$$Z(\theta) = \sum_{\mathbf{h}, \mathbf{v}} \exp(-E(\mathbf{v}, \mathbf{h}; \theta))$$

Because it is independent conditions between hidden nodes (since there is no connection between the nodes), namely:

$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v})$$

Then we can be relatively easily (for factoring formula Factorizes) obtained on the basis of  $\mathbf{v}$  is given on the visible layer, the  $j$ -th hidden layer node is the probability of 0 or 1 is:

$$P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - a_j)}$$

Similarly, in the hidden layer is given based on  $\mathbf{h}$ , visible layer of the  $i$ -th node is the probability of 0 or 1 may also be easily obtained:

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij}h_j - b_i)}$$

Given a set of samples satisfy iid:  $D = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(N)}\}$ , We need to learn the parameters  $\theta = \{W, a, b\}$ .

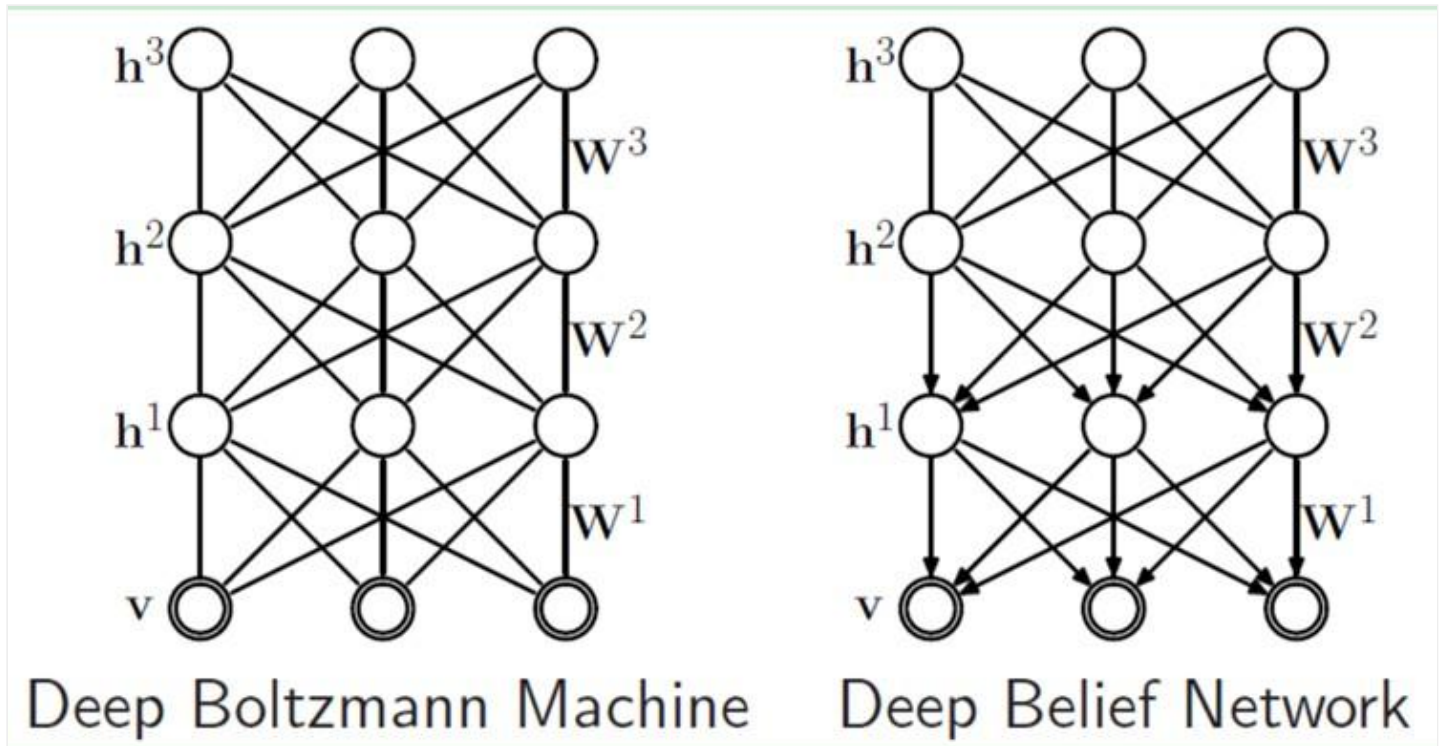
We maximize the log-likelihood function (maximum likelihood estimation: For some probability model, we need to select a parameter, so that the probability of observing a sample of our current maximum):

$$L(\theta) = \frac{1}{N} \sum_{n=1}^N \log P_{\theta}(\mathbf{v}^{(n)}) - \frac{\lambda}{N} \|W\|_F^2$$

I.e. maximum logarithmic likelihood function derivative,  $L$  can be obtained corresponding to the maximum of the parameter  $W$ .

$$\frac{\partial L(\theta)}{\partial W_{ij}} = E_{P_{data}}[v_i h_j] - E_{P_{\theta}}[v_i h_j] - \frac{2\lambda}{N} W_{ij}$$

If we increase the number of layers of hidden layer, we can get Deep Boltzmann Machine (DBM); near the visible part if we use the Bayesian belief network layer (i.e., directed graph model, of course, there is still confinement layer node no link between), Restricted Boltzmann Machine is used in a portion farthest from the visible layer, we can get DeepBelief Net (DBN).



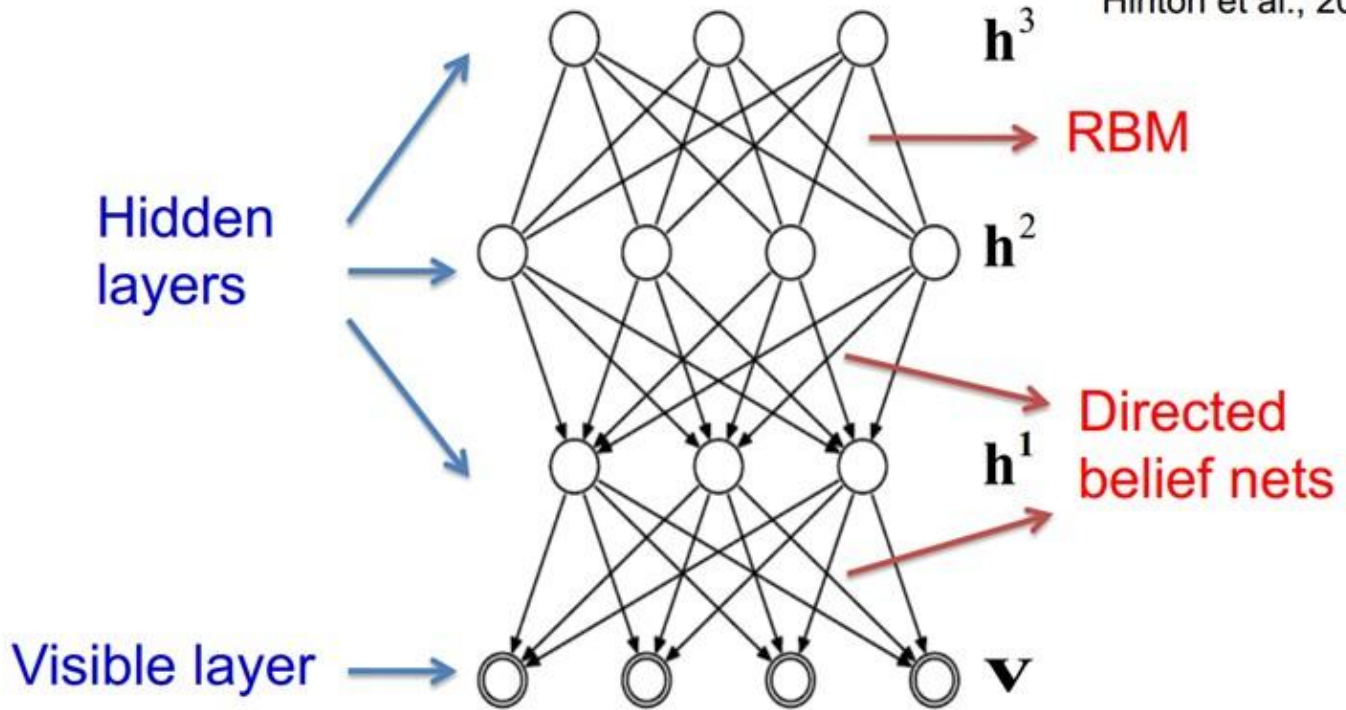
### Deep Belief Networks are convinced of the network

DBNs is a probability generation model, and neural network model of the traditional discrimination relatively generated model is to establish a joint distribution between the observed data and labels for the  $P(\text{Observation} | \text{Label})$  and  $P(\text{Label} | \text{Observation})$  have done assessments, but only just discriminant model to assess the latter, that is,  $P(\text{Label} | \text{Observation})$ . When the application for the traditional BP neural network algorithms in depth, DBNs encountered the following problems:

- (1) The need to provide a training sample set labels;
- (2) the learning process is slow;
- (3) improper selection can lead to learning parameters converge to local optima.

# DBN structure

Hinton et al., 2006



$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \dots, \mathbf{h}^l) = P(\mathbf{v} | \mathbf{h}^1) P(\mathbf{h}^1 | \mathbf{h}^2) \dots P(\mathbf{h}^{l-2} | \mathbf{h}^{l-1}) P(\mathbf{h}^{l-1}, \mathbf{h}^l)$$

A plurality of limiting DBNs Boltzmann machine (Restricted Boltzmann Machines) component layers, a typical type of neural network shown in Figure III. These networks are "restricted" to a visible layer and a hidden layer, interlayer connection is present, but no connection between the units within the layer. Hidden layer unit is trained to capture the correlation demonstrated in the visible layer of the high order data.

First, not considered constituting a topmost associative memory (associative memory) of the two layers are connected by a self-generated DBN weight down to a top of a guide to determine, as a building block RBMs as compared to traditional and depth layered sigmoid belief networks, it is easy to learn the connection weights.

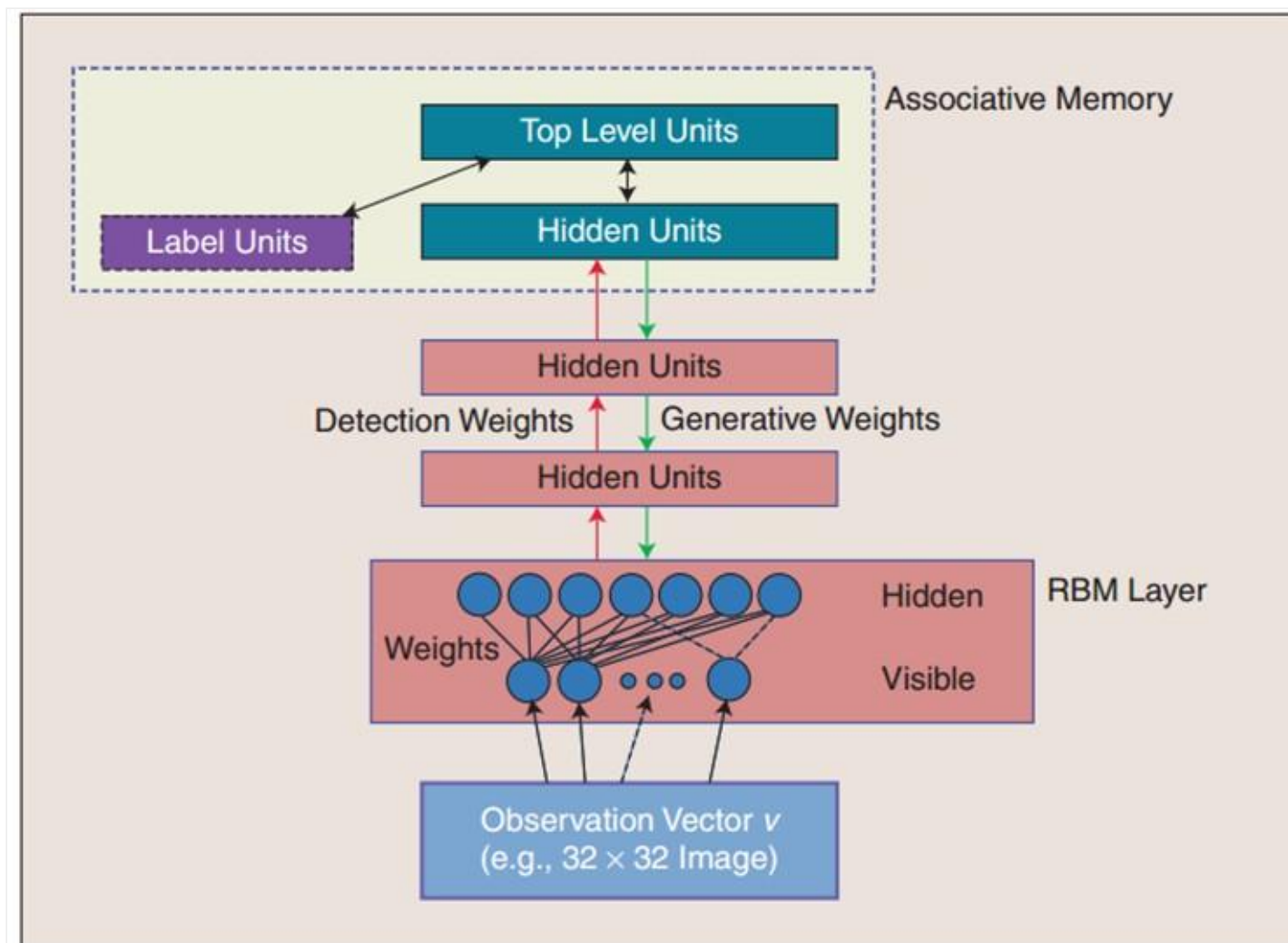
The very beginning, layer by layer by a greedy unsupervised pre-training methods to get the right value generation model, unsupervised greedy Hinton layer by layer method proved to be effective, and is referred to as contrast differences (contrastive divergence).

In the training phase, the visible layer generates a vector  $\mathbf{v}$ , it is transmitted to the hidden layer through the values. In turn, the input layer will be visible randomly selected to attempt to reconstruct the original input signal. Finally, the new visual neurons to activate the unit before transmitting the reconstructed hidden layer activation unit obtaining  $\mathbf{h}$  (in the training process, the magnitude of the first visible to the implicit mapping unit; then visualized by the hidden layer unit reconstruction; new visual elements again mapped to the hidden units, thus obtaining a new hidden units perform such repeated step is called Gibbs sampling). These steps backward and forward is that we are familiar with Gibbs



sampling, and the relevance of the difference between the hidden layer activation unit and visual input as the main basis on which weights update.

Training time will be significantly reduced, since only a single step closer to the maximum likelihood learning. Increase into the network of each layer will improve the training data for the number of probability, we can understand the true expression closer to energy. This significant expansion, and the use of unlabeled data, is a decisive factor in any depth learning applications.



**FIGURE 3** Illustration of the Deep Belief Network framework.

In the top two weights are connected together, so that the output of the lower layer will provide a reference to the associated top or clue, so that its top will be to contact the contents of its memory. And we are most concerned about, the last conceivable that discrimination performance, such as classification tasks inside.

After the pre-training, DBN by using tagged data BP algorithm to make adjustments to the discrimination performance. Here, a tag set will be attached to the top layer (to promote associative memory), to obtain a classified network by a surface from the bottom up, learned the identification weights. This performance will be better than the simple BP algorithm to train the network. This can be very intuitive interpretation, DBNs BP algorithm only needs the value of the parameter space for the right to conduct a local



search, which compared to the previous forward neural networks, training is much faster, and less time convergence.

DBNs flexibility makes it easier to expand. Expansion is a convolution DBNs (Convolutional Deep Belief Networks (CDBNs)). DBNs does not consider the two-dimensional image information structure, because the input is from a simple quantization of the one-dimensional image matrix. CDBNs while taking into account that this problem by using a spatial relationship neighborhood pixel through a region called the convolution model RBMs reaches invariance generated model, and can be readily converted into a high-dimensional image obtained. DBNs did not explicitly deal with the study of observed variables contact time, although there are already research in this area, such as stacking time RBMs, as a promotion, there dubbed temporal convolution machines sequence learning, the application of such a sequence of learning, to the voice signal processing problems caused by an exciting future research directions.

Currently, research and related DBNs automatic encoder comprises a stack, which is replaced by a conventional encoder with automatic stacking DBNs inside RBMs. This makes it possible to produce trained by the same rules as the depth of multilayer neural network architecture, but it lacks strict requirements parameterized layer. And DBNs different, the encoder uses a discriminant model automatically, so this configuration is difficult to sample an input sample space, which makes the network more difficult to capture its internal representation. However, noise autoencoder well able to avoid this problem, and better than conventional DBNs. It is by adding random contamination during the training field and stacking of generalization. Single training noise reduction process and the automatic encoder RBMs training process generates the same model.