An Introduction to Deep Learning

Ludovic Arnold, Sébastien Rebecchi, Sylvain Chevallier, Hélène Paugam-Moisy

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

An Introduction to Deep Learning

Ludovic Arnold[1,2], Sébastien Rebecchi[1], Sylvain Chevallier[1], Hélène Paugam-Moisy[1,3]

1- Tao, INRIA-Saclay, LRI, UMR8623, Université Paris-Sud 11
F-91405 Orsay, France

2- LIMSI, UMR3251
F-91403 Orsay, France

3- Université Lyon 2, LIRIS, UMR5205
F-69676 Bron, France

Abstract. The deep learning paradigm tackles problems on which shallow architectures (e.g. SVM) are affected by the curse of dimensionality. As part of a two-stage learning scheme involving multiple layers of nonlinear processing a set of statistically robust features is automatically extracted from the data. The present tutorial introducing the ESANN deep learning special session details the state-of-the-art models and summarizes the current understanding of this learning approach which is a reference for many difficult classification tasks.

1 Introduction

In statistical machine learning, a major issue is the selection of an appropriate feature space where input instances have desired properties for solving a particular problem. For example, in the context of supervised learning for binary classification, it is often required that the two classes are separable by an hyperplane. In the case where this property is not directly satisfied in the input space, one is given the possibility to map instances into an intermediate feature space where the classes are linearly separable. This intermediate space can either be specified explicitly by hand-coded features, be defined implicitly with a so-called kernel function, or be automatically learned. In both of the first cases, it is the user's responsibility to design the feature space. This can incur a huge cost in terms of computational time or expert knowledge, especially with highly dimensional input spaces, such as when dealing with images.

As for the third alternative, automatically learning the features with deep architectures, i.e. architectures composed of multiple layers of nonlinear processing, can be considered as a relevant choice. Indeed, some highly nonlinear functions can be represented much more compactly in terms of number of parameters with deep architectures than with shallow ones (e.g. SVM). For example, it has been proven that the parity function for n-bit inputs can be coded by a feed-forward neural network with $O(\log n)$ hidden layers and $O(n)$ neurons, while a feed-forward neural network with only one hidden layer needs an exponential number of the same neurons to perform the same task [1]. Moreover, in the case of highly varying functions, learning algorithms entirely based on local generalization are severely impacted by the curse of dimensionality [2]. Deep architectures address this issue with the use of distributed representations and as such may constitute a tractable alternative.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

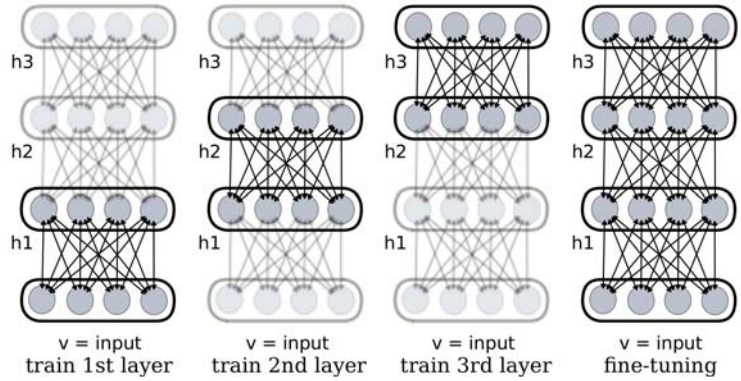

Figure 1: The deep learning scheme: a greedy unsupervised layer-wise pre-training stage followed by a supervised fine-tuning stage affecting all layers.

Unfortunately, training deep architectures is a difficult task and classical methods that have proved effective when applied to shallow architectures are not as efficient when adapted to deep architectures. Adding layers does not necessarily lead to better solutions. For example, the more the number of layers in a neural network, the lesser the impact of the back-propagation on the first layers. The gradient descent then tends to get stuck in local minima or plateaus [3], which is why practitioners have often preferred to limit neural networks to one or two hidden layers.

This issue has been solved by introducing an unsupervised layer-wise pre-training of deep architectures [3, 4]. More precisely, in a deep learning scheme each layer is treated separately and successively trained in a greedy manner: once the previous layers have been trained, a new layer is trained from the encoding of the input data by the previous layers. Then, a supervised fine-tuning stage of the whole network can be performed (see Fig. 1).

This paper aims at providing to the reader a better understanding of the deep learning through a review of the literature and an emphasis of its key properties. Section 2 details a widely used deep network model: the deep belief network or stacked restricted Boltzmann machines. Other models found in deep architectures are presented in Sect. 3, i.e. stacked auto-associators, deep kernel machines and deep convolutional networks. Section 4 summarizes the main results in the different application domains, points out the contributions of the deep learning scheme and concludes the tutorial.

2 Deep learning with RBMs

2.1 Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMs) are at the intersection of several fields of study and benefit from a rich theoretical framework [5, 6]. First, we will

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

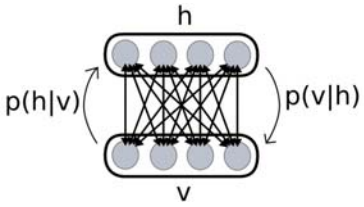

Figure 2: The RBM architecture with a visible ($\mathbf{v}$) and a hidden ($\mathbf{h}$) layers.

present them as a probabilistic model before showing how the neural network equations arise naturally.

An RBM defines a probability distribution p on data vectors $\mathbf{v}$ as follows:

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{u},\mathbf{g}} e^{-E(\mathbf{u},\mathbf{g})}}. \tag{1}$$

The variable $\mathbf{v}$ is the input vector and the variable $\mathbf{h}$ corresponds to unobserved features [7] that can be thought of as hidden causes not available in the original dataset. An RBM defines a joint probability on both the observed and unobserved variables which are referred to as visible and hidden units respectively (see Fig. 2). The distribution is then marginalized over the hidden units to give a distribution over the visible units only. The probability distribution is defined by an energy function E (RBMs are a special case of energy-based models [8]), which is usually defined over couples $(\mathbf{v}, \mathbf{h})$ of binary vectors by:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} w_{ij} v_i h_j, \tag{2}$$

with a_i and b_j the biases associated to the input variables v_i and hidden variables h_j respectively and w_{ij} the weights of a pairwise interaction between them. In accordance with (1), configurations $(\mathbf{v}, \mathbf{h})$ with a low energy are given a high probability whereas a high energy corresponds to a low probability.

The energy function above is crafted to make the conditional probabilities $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$ tractable. The computation is done using the usual neural network propagation rule (see Fig. 2) with:

$$\begin{aligned} p(\mathbf{v}|\mathbf{h}) &= \prod_i p(v_i|\mathbf{h}) \quad \text{and} \quad p(v_i = 1|\mathbf{h}) = \text{sigm}\left(a_j + \sum_j h_j w_{ij}\right), \\ p(\mathbf{h}|\mathbf{v}) &= \prod_j p(h_j|\mathbf{v}) \quad \text{and} \quad p(h_j = 1|\mathbf{v}) = \text{sigm}\left(b_j + \sum_i v_i w_{ij}\right), \end{aligned} \tag{3}$$

where $\text{sigm}(x) = 1/(1 + \exp(-x))$ is the logistic activation function.

The model with the energy function (2) defines a distribution over binary vectors and, as such, is not suitable for continuous valued data. To address this

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

issue, E can be appropriately modified to define the Gaussian-Bernoulli RBM by including a quadratic term on the visible units [3]:

$$E(\mathbf{v}, \mathbf{h}) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} w_{ij} \frac{v_i}{\sigma_i} h_j,$$

where σ_i represents the variance of the input variable v_i. Using this energy function, the conditional probability $p(\mathbf{h}|\mathbf{v})$ is almost unchanged but $p(\mathbf{v}|\mathbf{h})$ becomes a multivariate Gaussian with mean $a_i + \sigma_i \sum_j w_{ij} h_j$ and a diagonal covariance matrix:

$$\begin{aligned} p(v_i = x|\mathbf{h}) &= \frac{1}{\sigma_i \sqrt{2\pi}} \cdot e^{-\frac{\left(x - a_i - \sigma_i \sum_j w_{ij} h_j\right)^2}{2\sigma_i^2}}, \\ p(h_j = 1|\mathbf{v}) &= \text{sigm}\left(b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij}\right). \end{aligned} \tag{4}$$

In a deep architecture using Gaussian-Bernoulli RBM, only the first layer is real-valued whereas all the others have binary units. Other variations of the energy function are given in [3, 9, 10] to address the issue of continuous valued inputs.

2.2 Learning with RBMs and Contrastive Divergence

In order to train RBMs as a probabilistic model, the natural criterion to maximize is the log-likelihood. This can be done with gradient ascent from a training set $\mathcal{D}$ likewise:

$$\begin{aligned} \frac{\partial \log p(\mathcal{D})}{\partial w_{ij}} &= \sum_{\mathbf{x} \in \mathcal{D}} \frac{\partial \log p(\mathbf{x})}{\partial w_{ij}} \\ &= \sum_{\mathbf{x} \in \mathcal{D}} \frac{\sum_{\mathbf{g}} \frac{\partial E(\mathbf{x}, \mathbf{g})}{\partial w_{ij}} e^{-E(\mathbf{x}, \mathbf{g})}}{\sum_{\mathbf{g}} e^{-E(\mathbf{x}, \mathbf{g})}} - \sum_{\mathbf{x} \in \mathcal{D}} \frac{\sum_{\mathbf{u}} \sum_{\mathbf{g}} \frac{\partial E(\mathbf{u}, \mathbf{g})}{\partial w_{ij}} e^{-E(\mathbf{u}, \mathbf{g})}}{\sum_{\mathbf{u}} \sum_{\mathbf{g}} e^{-E(\mathbf{u}, \mathbf{g})}}, \\ &= E_{\text{data}}\left[\frac{\partial E(\mathbf{x}, \mathbf{g})}{\partial w_{ij}}\right] - E_{\text{model}}\left[\frac{\partial E(\mathbf{u}, \mathbf{g})}{\partial w_{ij}}\right], \end{aligned}$$

where the first term is the expectation of $\frac{\partial E(\mathbf{x}, \mathbf{g})}{\partial w_{ij}}$ when the input variables are set to an input vector $\mathbf{x}$ and the hidden variables are sampled according to the conditional distribution $p(\mathbf{h}|\mathbf{x})$. The second term is an expectation of $\frac{\partial E(\mathbf{u}, \mathbf{g})}{\partial w_{ij}}$ when $\mathbf{u}$ and $\mathbf{g}$ are sampled according to the joint distribution of the RBM $p(\mathbf{u}, \mathbf{g})$ and is intractable. It can however be approximated with a Markov chain Monte Carlo algorithm such as Gibbs sampling: starting from any configuration $(\mathbf{v}^0, \mathbf{h}^0)$, one

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

samples $\mathbf{h}^t$ according to $p(\mathbf{h}|\mathbf{v}^{t-1})$ and $\mathbf{v}^t$ according to $p(\mathbf{v}|\mathbf{h}^t)$ until the sample $(\mathbf{v}^t, \mathbf{h}^t)$ is distributed closely enough to the target distribution $p(\mathbf{v}, \mathbf{h})$.

In practice, the number of steps can be greatly reduced by starting the Markov chain with a sample from the training dataset and assuming that the model is not too far from the target distribution. This is the idea behind the Contrastive Divergence (CD) learning algorithm [11]. Although the maximized criterion is not the log-likelihood anymore, experimental results show that gradient updates almost always improve the likelihood of the model [11]. Moreover, the improvement to the likelihood tend to zero as the length of the chain increases [12], an argument which supports running the chain for a few steps only. Notice that the possibility to use only the sign of the CD update is explored in the present special session [13].

2.3 From stacked RBMs to deep belief networks

In an RBM, the hidden variables are independent conditionally to the visible variables, but they are not statistically independent. Stacking RBMs aims at learning these dependencies with another RBM. The visible layer of each RBM of the stack is set to the hidden layer of the previous RBM (see Fig. 3). Following the deep learning scheme, the first RBM is trained from the input instances and other RBMs are trained sequentially after that. Stacking RBMs increases a bound on the log-likelihood [14], which supports the expectation to improve the performance of the model by adding layers.

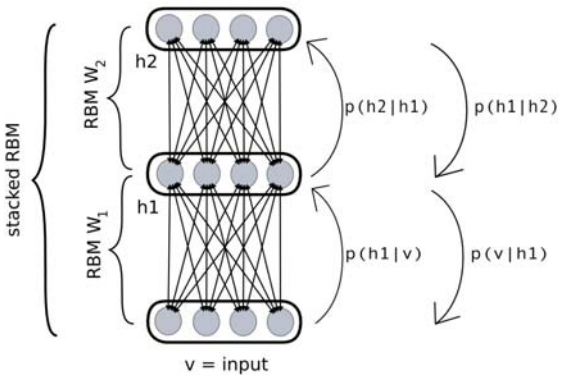

Figure 3: The stacked RBMs architecture.

A stacked RBMs architecture is a deep generative model. Patterns generated from the top RBM can be propagated back to the input layer using only the conditional probabilities as in a belief network. This setup is referred to as a Deep Belief Network [4].

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

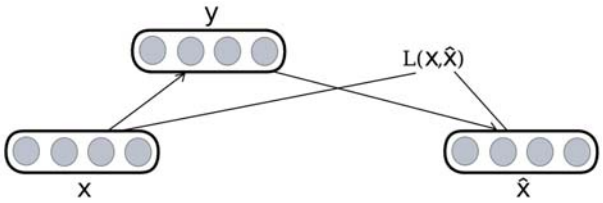

Figure 4: The training scheme of an AA.

3 Other models and variations

3.1 Stacked Auto-Associators

Another module which can be stacked in order to train a deep neural network in a greedy layer-wise manner is the Auto-Associator (AA) [15, 16].

An AA is a two-layers neural network. The first layer is the encoding layer and the second is the decoding layer. The number of neurons in the decoding layer is equal to the network's input dimensionality. The goal of an AA is to compute a code y of an input instance x from which x can be recovered with high accuracy. This models a two-stage approximation to the identity function:

$$f_{\text{dec}}(f_{\text{enc}}(x)) = f_{\text{dec}}(y) = \hat{x} \simeq x,$$

with f_{enc} the function computed by the encoding layer and f_{dec} the function computed by the decoding layer (see Fig. 4).

An AA can be trained by applying standard back-propagation of error derivatives. Depending on the nature of the input data, the loss function can either be the squared error L_{SE} for continuous values or the cross-entropy L_{CE} for binary vectors:

$$L_{\text{SE}}(x, \hat{x}) = \sum_i (\hat{x}_i - x_i)^2,$$
$$L_{\text{CE}}(x, \hat{x}) = \sum_i [x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i)].$$

The AA training method approximates the CD method of the RBM [14]. Another important fact is that an AA with a nonlinear f_{enc} differs from a PCA as it is able to capture multimodal aspects of the input distribution [17].

Similarly to the parametrization in an RBM, the decoder's weight matrix W_{dec} can be set to the transpose of the encoder's weight matrix, i.e. $W_{\text{dec}} = W_{\text{enc}}^{\intercal}$. In such a case, the AA is said to have tied weights. The advantage of this constraint is to avoid undesirable effects of the training process, such as encoding the identity function, i.e. $f_{\text{enc}}(x) = x$. This useless result is possible when the encoding dimensionality is not smaller than the input dimensionality.

An interesting variant of the AA is the Denoising Auto-Associator (DAA) [18]. A DAA is an AA trained to reconstruct noisy inputs. To achieve this

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

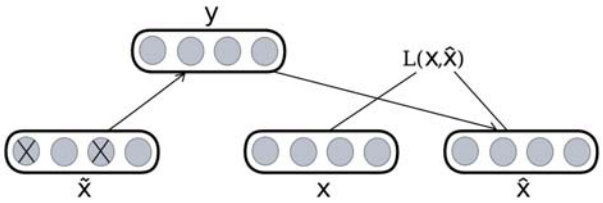

Figure 5: The training scheme of a DAA. Noisy components are marked with a cross.

goal, the instance fed to the network is not x but a corrupted version $\tilde{x}$. After training, if the network is able to compute a reconstruction $\hat{x}$ of x with a small loss, then it is admitted that the network has learned to remove the noise in the data in addition to encode it in a different feature space (see Fig. 5).

Finally, a Stacked Auto-Associator (SAA) [3, 19, 18, 20] is a deep neural network trained following the deep learning scheme: an unsupervised greedy layer-wise pre-training before a fine-tuning supervised stage, as explained in Sect. 2.3 (see also Fig. 1). Surprisingly, for d dimensional inputs and layers of size $k \geqslant d$, a SAA rarely learns the identity function [3]. In addition, it is possible to use different regularization rules and the most successful results have been reported with adding a sparsity constraint on the encoding unit activations [20, 21, 22]. This leads to learning very different features (w.r.t RBM) in the intermediate layers and the network performs a trade-off between reconstruction loss and information content of the representation [21].

3.2 Deep Kernel Machines

The Multilayer Kernel Machine (MKM) [23] has been introduced as a way to learn highly nonlinear functions with the iterative application of weakly nonlinear kernel methods.

The authors use the Kernel Principal Component Analysis (KPCA) [24] for the unsupervised greedy layer-wise pre-training stage of the deep learning scheme. From this method, the $\ell + 1^{\text{th}}$ layer learns a new representation of the output of the layer ℓ by extracting the n_ℓ principal components of the projection of the output of ℓ in the feature space induced by the kernel.

In order to lower as much as possible the dimensionality of the new representation in each layer, the authors propose to apply a supervised strategy devoted to selecting the best informative features among the ones extracted by the KPCA. It can be summarized as follows:

1. rank the n_l features according to their mutual information with the class labels;

2. for different values of K and $m_l \in \{1 \dots n_l\}$, compute the classification error rate of a K-NN classifier using only the m_l most informative features on a validation set;

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

3. the value of m_l with which the classifier has reached the lowest error rate determines the number of features to retain.

However, the main drawback of using KPCA as the building block of an MKM lies in the fact that the feature selection process must be done separately and thus requires a time-expensive cross validation stage. To get rid of this issue when training an MKM it is proposed in this special session [25] to use a more efficient kernel method, the Kernel Partial Least Squares (KPLS).

KPLS does not need cross validation to select the best informative features but embeds this process in the projection strategy [26]. The features are obtained iteratively, in a supervised manner. At each iteration j, KPLS selects the j^{th} feature the most correlated with the class labels by solving an updated eigenproblem. The eigenvalue λ_j of the extracted feature indicates the discriminative importance of this feature. The number of features to extract, i.e. the number of iterations to be performed by KPLS, is determined by a simple thresholding of λ_j.

3.3 Deep Convolutional Networks

Convolutional networks are the first examples of deep architectures [27, 28] that have successfully achieved a good generalization on visual inputs. They are the best known method for digit recognition [29]. They can be seen as biologically inspired architectures, imitating the processing of "simple" and "complex" cortical cells which respectively extract orientations information (similar to a Gabor filtering) and compositions of these orientations.

The main idea of convolutional networks is to combine local computations (convolution of the signal with weight sharing units) and pooling. The convolutions are intended to give translation invariance to the system, as the weights depend only on spatial separation and not on spatial position. The pooling allows to construct a more abstract set of features through nonlinear combination of the previous level features, taking into account the local topology of the input data. By alternating convolution layers and pooling layers, the network successively extracts and combines local features to construct a good representation of the input. The connectivity of the convolutional networks, where each unit in a convolution or a pooling layer is only connected to a small subset of the preceding layer, allows to train networks with as much as 7 hidden layers. The supervised learning is easily achieved, through an error gradient backpropagation.

On the one hand, convolutional framework has been applied to RBM and DBN [10, 30, 31]. In [31] the authors derive a generative pooling strategy which scales well with image size and they show that the intermediate representations are more abstract in the higher layer (from edges in the lower layers to object parts in the higher). On the other hand, the unsupervised pre-training stage of deep learning have been applied to convolutional networks [32] and can greatly reduce the number of labeled examples required. Furthermore, deep convolutional networks with sparse regularization [33] yield very promising results for difficult visual detection tasks, such as pedestrian detection.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

4 Discussion

4.1 What are the applicative domains for deep learning?

Deep learning architectures express their full potential when dealing with highly varying functions, requiring a high number of labeled samples to be captured by shallow architectures. Unsupervised pre-training allows, in practice, to achieve good generalization performance when the training set is of limited size by positioning the network in a region of the parameter space where the supervised gradient descent is less likely to fall in a local minimum of the loss function. Deep networks have been largely applied to visual classification databases such as handwritten digits[1], object categories[2] [3] [4], pedestrian detection [33] or off-road robot navigation [34], and also on acoustic signals to perform audio classification [35]. In natural language processing, a very interesting approach [36] gives a proof that deep architectures can perform multi-task learning, giving state-of-the-art results on difficult tasks like semantic role labeling. Deep architectures can also be applied to regression with Gaussian processes [37] and time series prediction [38]. In the latter, the conditional RBM have given promising results.

Another interesting application area is highly nonlinear data compression. To reduce the dimensionality of an input instance, it is sufficient for a deep architecture that the number of units in its last layer is smaller than its input dimensionality. In practice, limiting the size of a neuron layer can promote interesting nonlinear structure of the data. Moreover, adding layers to a neural network can lead to learning more abstract features, from which input instances can be coded with high accuracy in a more compact form. Reducing the dimensionality of data has been presented as one of the first application of deep learning [39]. This approach is very efficient to perform semantic hashing on text documents [22, 40], where the codes generated by the deepest layer are used to build a hash table from a set of documents. Retrived documents are those whose code differs only by a few bits from the query document code. A similar approach for a large scale image database is presented in this special session [41].

4.2 Open questions and future directions

A significant part of the ongoing research aims at improving the deep networks building blocks For RBMs, several propositions have been made to use real-valued units rather than binary ones either by integrating the covariance of the visible units in the hidden units update [42] or by approximating real-valued units by noisy rectified linear units [9, 10]. For AAs, the denoising criterion is particularly investigated since it achieves very good results on visual classification tasks [43].

[1] MNIST: http://yann.lecun.com/exdb/mnist/
[2] Caltech-101: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
[3] NORB: http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/
[4] CIFAR-10: http://www.cs.utoronto.ca/~kriz/cifar.html

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

Because the construction of good intermediate representations is a crucial part of deep learning, a meaningful approach is to study the response of the individual units in each layer. An effective strategy is to find the input pattern that maximizes the activation of a given unit, starting from a random input and performing a gradient ascent [44].

A major interrogation underlying the deep learning scheme concerns the role of the unsupervised pre-training. In [44], the authors argue that pre-training acts as an unusual form of regularization. By selecting a region in the parameter space which is not always better than a random one, pre-training systematically leads to better generalization. The authors provide results showing that pre-training does not act as an optimization procedure which selects the region of the parameter space where the basins of attraction are the deepest. Pre-training only modifies the starting point of supervised training and the regularization effect does not vanish when the number of data increases. Deep learning breaks down the problem of optimizing lower layers given that the upper layers have not yet been trained. Lower layers extract robust and disentangled representations of the factors of variations (e.g. on images: translation, rotation, scaling), whereas higher layers select and combine these representations. A fusion of the unsupervised and supervised paradigms in one single training scheme is an interesting way to explore.

Choosing the correct dimensions of a deep architecture is not an obvious process and the results shown in [29] open new perspectives on this topic. A convolution network with a random filter bank and with the correct nonlinearities can achieve near state-of-the-art results when there is few training labeled data (such as in the Caltech-101 dataset). It has been shown that the architecture of convolutional network has a major influence on the performance and that it is possible to achieve a very fast architecture selection using only random weights and no time-consuming learning procedure [45]. This, along with the work of [46], points toward new directions for answering the difficult question of how to efficiently set the sizes of layers in deep networks.

4.3 Conclusion

The strength of deep architectures is to stack multiple layers of nonlinear processing, a process which is well suited to capture highly varying functions with a compact set of parameters. The deep learning scheme, based on a greedy layer-wise unsupervised pre-training, allows to position deep networks in a parameter space region where the supervised fine-tuning avoids local minima. Deep learning methods achieve very good accuracy, often the best one, for tasks where a large set of data is available, even if only a small number of instances are labeled. This approach raises many theoretical and practical questions, which are investigated by a growing and very active research community and casts a new light on our understanding of neural networks and deep architectures.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

Acknowledgements

This work was supported by the French ANR as part of the ASAP project under grant ANR_09_EMER_001_04.

References

[1] Y. Bengio and Y. LeCun. Scaling learning algorithms towards AI. In *Large-Scale Kernel Machines*. 2007.

[2] Y. Bengio, O. Delalleau, and N. Le Roux. The curse of highly variable functions for local kernel machines. In *NIPS*, 2005.

[3] Y. Bengio, P. Lamblin, V. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, 2007.

[4] G. E. Hinton, S. Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neur. Comput.*, 18:1527–1554, 2006.

[5] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing*, pages 194–281. 1986.

[6] D. H . Ackley, G. E . Hinton, and T. J . Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[7] Z. Ghahramani. Unsupervised learning. In *Adv. Lect. Mach. Learn.*, pages 72–112. 2004.

[8] M. Ranzato, Y.-L. Boureau, S. Chopra, and Y. LeCun. A unified energy-based framework for unsupervised learning. In *AISTATS*, 2007.

[9] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, 2010.

[10] A. Krizhevsky. Convolutional deep belief networks on CIFAR-10. Technical report, Univ. Toronto, 2010.

[11] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neur. Comput.*, 14:1771–1800, 2002.

[12] Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neur. Comput.*, 21:1601–1621, 2009.

[13] A. Fischer and C. Igel. Training RBMs depending on the signs of the CD approximation of the log-likelihood derivatives. In *ESANN*, 2011.

[14] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2:1–127, 2009.

[15] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[16] G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.

[17] N. Japkowicz, S. J. Hanson, and M. A. Gluck. Nonlinear autoassociation is not equivalent to PCA. *Neur. Comput.*, 12:531–545, 2000.

[18] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[19] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, 2007.

[20] M. A. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS*, 2006.

[21] M. Ranzato, Y-L. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *NIPS*, 2008.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

[22] P. Mirowski, M. Ranzato, and Y. LeCun. Dynamic auto-encoders for semantic indexing. In *NIPS WS8*, 2010.

[23] Y. Cho and L. Saul. Kernel methods for deep learning. In *NIPS*, 2009.

[24] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neur. Comput.*, 10:1299–1319, 1998.

[25] F. Yger, M. Berar, G. Gasso, and A. Rakotomamonjy. A supervised strategy for deep kernel machine. In *ESANN*, 2011.

[26] R. Rosipal, L. J. Trejo, and B. Matthews. Kernel PLS-SVC for linear and nonlinear classification. In *ICML*, 2003.

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

[29] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

[30] G. Desjardins and Y. Bengio. Empirical evaluation of convolutional RBMs for vision. Technical report, Univ. Montréal, 2008.

[31] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009.

[32] K. Kavukcuoglu, M. A. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *CVPR*, 2009.

[33] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. In *NIPS*. 2010.

[34] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun. Learning long-range vision for autonomous off-road driving. *J. Field Robot.*, 26:120–144, 2009.

[35] H. Lee, Y. Largman, P. Pham, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*, 2009.

[36] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

[37] R. Salakhutdinov and G. E. Hinton. Using deep belief nets to learn covariance kernels for gaussian processes. In *NIPS*, 2008.

[38] M. D. Zeiler, G. W. Taylor, N. F. Troje, and G. E. Hinton. Modeling pigeon behaviour using a conditional restricted Boltzmann machine. In *ESANN*, 2009.

[39] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

[40] R. Salakhutdinov and G. E. Hinton. Semantic hashing. *Int. J. Approximate Reasoning*, 50:969–978, 2009.

[41] A. Krizhevsky and G. E. Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

[42] M. Ranzato, A. Krizhevsky, and G. E. Hinton. Factored 3-way restricted Boltzmann machines for modeling natural images. In *AISTATS*, 2010.

[43] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 2010.

[44] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, 2010.

[45] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Ng. On random weights and unsupervised feature learning. In *NIPS WS8*, 2010.

[46] L. Arnold, H. Paugam-Moisy, and M. Sebag. Unsupervised layer-wise model selection in deep neural networks. In *ECAI*, 2010.