Published in Künstliche Intelligenz: Volume 26, Issue 4 (2012), Page 357-363

Künstliche Intelligenz manuscript No.
(will be inserted by the editor)

Deep Learning

Layer-wise Learning of Feature Hierarchies

Hannes Schulz · Sven Behnke

Received: date / Accepted: date

Abstract Hierarchical neural networks for object recognition have a long history. In recent years, novel methods for incrementally learning a hierarchy of features from unlabeled inputs were proposed as good starting point for supervised training. These deep learning methods—together with the advances of parallel computers—made it possible to successfully attack problems that were not practical before, in terms of depth and input size. In this article, we introduce the reader to the basic concepts of deep learning, discuss selected methods in detail, and present application examples from computer vision and speech recognition.

Keywords hierarchical feature learning · unsupervised learning · object categorization

1 Introduction

Supervised learning tasks, such as assigning a class label to images, are given as a set of example input-output pairs where the output must be predicted. Different learning architectures, such as support vector machines, neural networks, decision trees, and memory-based methods (e.g. k nearest neighbors) can be used to approximate the desired classification function not only for the given examples, but also for unseen test images.

Frequently, classification is not done directly on the raw pixel input, but an intermediate representation—a vector of *features*—is extracted first which is then classified, resulting in a two-stage computation. This approach performs very well if the features represent the essential information needed for classification. Obviously, feature extraction is not free of parameters and depends on the type of data and the task. For example, we might expect a different set of features is useful for detecting cars in images than for detecting people. Perhaps surprisingly, however, some types of features, like localized edges for natural images, can be adopted to a range of tasks (Bottou, 2011). These features seem to be generic representations of the input signal. Finding these generic features is difficult though—often feature extractors are hand-crafted and selected by testing them on many similar learning problems.

Other methods, such as feed-forward neural networks with a single hidden layer, learn features from

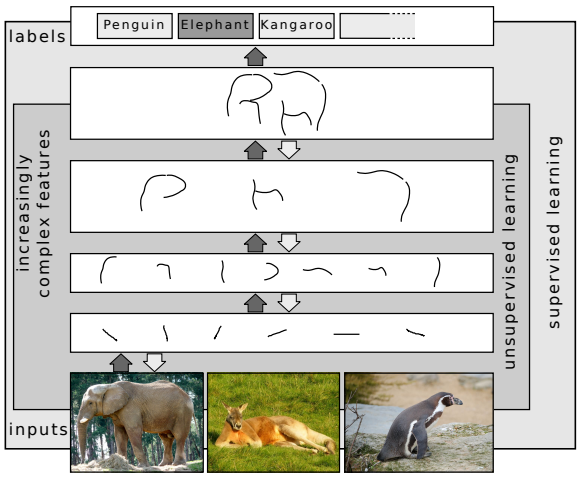

Fig. 1: Schematic overview of layer-wise learning of feature hierarchies. Increasingly complex features are determined from input using unsupervised learning. The features can be used for supervised task learning.

University Bonn, Autonomous Intelligent Systems
Institut für Informatik VI
Friedrich-Ebert-Allee 144, 53113 Bonn
Tel: +49 (0) 228 73-4422
Fax: +49 (0) 228 73-4425
E-mail: {schulz, behnke}@ais.uni-bonn.de

the training set by optimizing the parameters of feature extraction and the classifier simultaneously. While such networks can in principle represent almost any function (Cybenko, 1989), the number of required feature detectors in the hidden layer can be exponential in the number of inputs. This property is a generalization of the circuit complexity result that any Boolean function can be represented by two layers of conjunction and disjunction of inputs (Shannon, 1949). Often however, Boolean functions can be represented more space-efficiently by multi-stage binary decision diagrams that are less wide and hence need less logic units. The gain in efficiency is made possible by reusing the results of lower-level circuits at higher levels. Applying this finding to the feed-forward neural network context, we can save space—and time in sequential processing—by combining lower-level features to more abstract features when we allow multiple hidden layers and create a *feature hierarchy*.

Hierarchical neural networks for object categorization in images have a long history. They are motivated by the hierarchical structure of the ventral stream of the visual cortex, where a sequence of retinotopic representations is computed that represents the field-of-view with decreasing spatial resolution, but increasing feature complexity. This network structure reflects the typical hierarchical structure of images, where edges can be grouped to parts, which form objects (Figure 1).

One of the earliest hierarchical neural networks for object recognition was the Neocognitron proposed by Fukushima (1980). In a sequence of feature extracting and pooling layers—which create invariance to local shifts—the network was able to recognize handwritten digits and letters, even if they had been deformed. Other prominent hierarchical neural networks for object recognition include LeNet by LeCun et al (1989), the HMAX network by Riesenhuber and Poggio (1999), and the Neural Abstraction Pyramid by Behnke (2003b). Hierarchical features are also learned in the hierarchy of parts proposed by Fidler and Leonardis (2007). Finally, hierarchy is a key feature in state-of-the-art architectures for object recognition. While shallow, extremely wide architectures perform very well (Coates et al, 2010), the best-performing classifiers for standard datasets are currently very deep with up to ten levels of features (Cireşan et al, 2012).

Despite the above examples, difficulties in learning the feature hierarchies often prevented better performance of deep architectures, as compared to the convex optimization of shallow models like the Support Vector Machine. To overcome these difficulties, Hinton et al.—who coined the term deep learning—proposed to initialize supervised training of deep networks with a feature hierarchy that was learned in an unsupervised way, layer by layer from the data (Hinton and Salakhutdinov, 2006; Hinton et al, 2006). The impressive performance of this method—together with massively parallel computation by GPUs—triggered a revival of neural networks research.

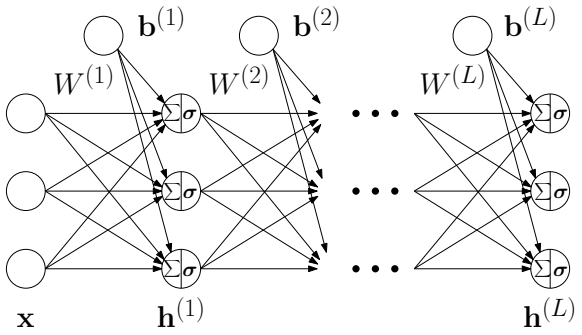

Fig. 2: Visualization of a deep architecture

2 Notation

To ease the following discussion, let us first define the terms. A deep architecture (as shown in Figure 2) with depth L is a function $\mathbf{h}^{(L)}$ with

$$\mathbf{h}^{(l)}(\mathbf{x}) = \boldsymbol{\sigma}\left(W^{(l)}\mathbf{h}^{(l-1)}(\mathbf{x}) + \mathbf{b}^{(l)}\right), \quad \mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x}. \quad (1)$$

Here, $\mathbf{x} \in \mathbb{R}^N$ is the input vector, $\boldsymbol{\sigma}(\cdot)$ is an elementwise sigmoid function such as $\sigma_i(\mathbf{z}) = (1 + \exp(-z_i))^{-1}$. The weight matrices $W^{(l)} \in \mathbb{R}^{N_l \times N_{l-1}}$ and the biases $\mathbf{b}^{(l)} \in \mathbb{R}^{N_l}$ constitute the layer parameters $\theta^{(l)}$. Intermediate $\mathbf{h}^{(l)}(\mathbf{x}) \in \mathbb{R}^{N_l}$ are referred to as hidden layers. When unambiguous, we omit the dependency on $\mathbf{x}$ and write $\mathbf{h}^{(l)}$. While Eq. (1) is based on dot products of input and weight vectors, it is also possible to rely on differences between the two vectors instead and set $\boldsymbol{\sigma}(\cdot)$ to a radial basis function.

Our features correspond to the rows of $W^{(l)}$ and can be determined by learning. We first formalize the task using a loss function which is minimal when the task is solved. *Learning* is then to find parameters such that the loss function is minimal on some training data $\mathcal{D}$. For example, we might choose the mean square loss

$$\ell_{\text{MSE}}\left(\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(L)}, \mathcal{D}\right) = \sum_{d=1}^{D}\sum_{n=1}^{N}\left(h_n^{(L)}(\mathbf{x}^d) - x_n^d\right)^2 \quad (2)$$

for an i.i.d. dataset $\mathcal{D} = \left\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^D\right\}$ with Gaussian noise model. This is an unsupervised task where the input is reconstructed from the features. If for some $l < L$ we have $N_l < N_0$, this requires learning to compress and

decompress the input. Supervised tasks provide some desired output or label in addition to the input data. If we assume binary labels $\mathbf{y} \in \{0,1\}^M$ with $\sum_{m=1}^M y_m = 1$, our dataset is $\mathcal{D} = \left\{(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), \ldots, (\mathbf{x}^D, \mathbf{y}^D)\right\}$, and we learn a classification task by minimizing the cross-entropy loss

$$\ell_{\text{CE}}\left(\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(L)}, \mathcal{D}\right) = -\sum_{d=1}^{D}\sum_{m=1}^{M}\left[y_m^d \cdot \log \frac{\exp\left(h_m^{(L)}(\mathbf{x}^d)\right)}{\sum_{m'=1}^{M}\exp\left(h_{m'}^{(L)}(\mathbf{x}^d)\right)}\right]. \quad (3)$$

In networks with at least one non-linear hidden layer, both losses are non-convex in the parameters. They are typically minimized by gradient descent, where the gradients are efficiently (linearly in the number of parameters) computed with the backpropagation algorithm.

3 Difficulties in Learning Deep Architectures

The principles of learning architectures with many levels have been known since the proposal of the backpropagation algorithm for multi-layer perceptrons (Rumelhart et al, 1986). In practice, however, using more than one hidden layer was neither common nor successful.

Erhan et al (2009) investigated the reasons for the often unsatisfying performance of deep architectures. They conclude that the problem stems from the learning phase, as

1. increasing depth increases the probability of finding poor local minima of the non-convex loss, and
2. training the lower layers (close to the input) is more difficult than training the higher layers (close to the teacher). A reason might be vanishing gradients, comparable to gradient propagation problems in recurrent neural networks (Hochreiter et al, 2001).

To remedy both problems, various strategies can be employed, which we will now discuss.

4 Deep Learning Strategies

4.1 Greedy Layer-wise Training

In their influential work on data reduction with neural networks, Hinton and Salakhutdinov (2006) introduced a first solution to the problems stated in Section 3. Before minimizing the loss of the deep network with L levels, they optimized a sequence of $L-1$ singe-layer problems using restricted Boltzmann machines (RBM).

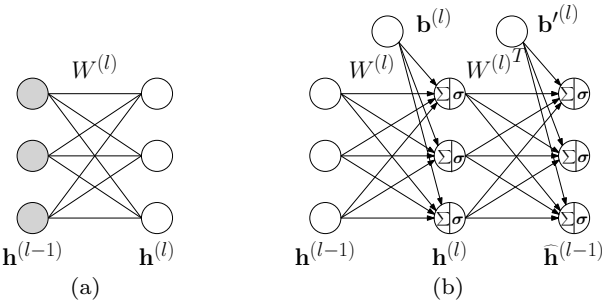

Fig. 3: Building blocks for greedy pre-training. (a) A restricted Boltzmann machine (RBM) is an undirected graphical model where variables are depicted as circles. Gray circles signify *observed* variables. (b) Auto-encoder network, reconstructing $\mathbf{h}^{(l-1)}$ with $\widehat{\mathbf{h}}^{(l-1)}$.

An RBM is a graphical model displayed in Figure 3 (a) that represents the log-linear probability distribution

$$p(\mathbf{h}^{(l-1)}, \mathbf{h}^{(l)}) = Z^{-1} \cdot \exp\left(\mathbf{b}^{(l-1)^T}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)^T}\mathbf{h}^{(l)} + \mathbf{h}^{(l)^T}W^{(l)}\mathbf{h}^{(l-1)}\right),$$

where Z is the *partition function* which ensures that $\iint p(\cdot,\cdot)\, d\mathbf{h}^{(l-1)}\, d\mathbf{h}^{(l)} = 1$. Here, $\mathbf{h}^{(l)}$ and $\mathbf{h}^{(l-1)}$ denote vectors of binary random variables. The parameters are chosen such that, when we marginalize out $\mathbf{h}^{(l)}$, they minimize the negative log-likelihood of the data distribution

$$\ell_{DD}(\theta^{(l)}, \mathcal{D}) = -\sum_{d=1}^{D}\log\sum_{\mathbf{h}^{(l)}} p\left(\mathbf{h}^{(l-1)}(\mathbf{x}^d), \mathbf{h}^{(l)}\right). \quad (4)$$

At first glance, we note that this loss is very different from Eqs. (2) and (3). Determining the gradient of Eq. (4) analytically is usually unfeasible since calculating the partition function Z in $p(\cdot,\cdot)$ scales exponentially with $\min(N_l, N_{l-1})$. Instead, approximations such as contrastive divergence (Hinton, 2002; Tieleman, 2008) are used. Due to the factorization of the graphical model, however, the expected conditional probability of $\mathbf{h}^{(l)}$ given $\mathbf{h}^{(l-1)}$ can be calculated analytically as

$$p\left(\mathbf{h}^{(l)}\,\middle|\,\mathbf{h}^{(l-1)}\right) = \boldsymbol{\sigma}\left(W^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right) \quad (5)$$

—which closely resembles Eq. (1)! After minimizing $\ell_{DD}(\theta^{(l)}, \mathcal{D})$, Hinton and Salakhutdinov transform $\mathcal{D}$ using Eq. (5) and iterate the process for $\theta^{(l+1)}$. Once all parameters of a feature hierarchy have been *pre-trained* in this way, they are *fine-tuned* with the original objective (e.g. (3)).

While the precise definitions of pre-training and fine-tuning loss vary, this general approach is prevalent in the deep learning literature. Most prominently,

auto-encoders (Bengio et al, 2006) are widely employed for pre-training, since their gradient can be calculated exactly. An auto-encoder (Figure 3 b) is a function $\widehat{\mathbf{h}}^{(l-1)} = {W^{(l)}}^T \mathbf{h}^{(l)}(\mathbf{x}) + {\mathbf{b}'}^{(l-1)}$. In its hidden layer $\mathbf{h}^{(l)}$, it creates a feature representation (encoding) of its input $\mathbf{h}^{(l-1)}$. The encoding is used for two purposes. Firstly, we optimize ℓ_{MSE} to reconstruct $\mathbf{h}^{(l-1)}$ from the features. Secondly, similar to RBMs, the features are used as input for $\mathbf{h}^{(l+1)}$, where the next-level auto-encoder is trained in the same fashion. The close connection between RBMs and auto-encoders has been investigated by Vincent (2011).

Why can we get away with changing the loss function between pre-training and fine-tuning seemingly at will? There are at least two reasons which have been identified:

1. The discussed pre-training methods identify generic features of the input, which resemble largely independent constituents. Higher-level features detect common co-occurrence patterns of lower level features. Intuitively, such features are likely to play a role in many objectives, termed the *structure assumption* in Weston et al (2008).
2. Pre-training can be seen as a *regularization* of fine-tuning. It moves the weights to a region in parameter space that has better generalization properties, and can therefore be seen as a form of semi-supervised learning (Erhan et al, 2010).

Additionally, by training only one layer at a time, we solve simpler problems with fewer local minima and cleaner gradients (as discussed in Section 3)—and postpone dealing with the complete, hard problem to the fine-tuning phase.

Other local learning methods that have been applied to learn feature hierarchies include competitive learning (Fukushima, 1980; Behnke, 1999), slow feature analysis (Wiskott and Sejnowski, 2002), non-negative matrix factorization (Behnke, 2003a), and deconvolutions (Zeiler et al, 2011).

4.2 Regularization

There are some restrictions on the type of building block we can use in deep learning. An important requirement is that we should not learn trivial features of the input. Commonly, data is normalized by centering and sometimes whitening. Still, if the number of features is large enough, the identity function can be learned, which does not yield new insights to be used in higher layers.

One way to enforce learning of novel features in auto-encoders is to keep the number of features small with respect to the number of inputs, such that for all $l < L : N_{l+1} < N_l$. Intuitively, we learn to represent the input using fewer bits and minimize the information loss.

Sometimes it is useful to learn highly overcomplete ($N_l \gg N_{l-1}$) feature hierarchies, e.g. for decomposing the signal for use in linear classifiers (Boureau et al, 2010). In this case, we can amend the auto-encoder loss function by approximately minimizing the number of non-zero entries of the hidden representation, resulting in

$$\ell_{MSE+S}(\theta^{(l)}, \mathcal{D}) = \ell_{MSE}(\theta^{(l)}, \mathcal{D}) + \lambda \sum_{d=1}^{D} \left\| \mathbf{h}^{(l)}(\mathbf{x}^d) \right\|_1 .$$

Optimizing this objective is related to sparse coding (Kavukcuoglu et al, 2010).

Large datasets allow better generalization to unseen test data from the same distribution. We can artificially introduce new data to profit from this effect, even if the generated data is not strictly i.i.d. In auto-encoders, this is achieved by adding noise to the input, and reconstructing the original, noise-free data (Vincent et al, 2010). This also requires to learn non-trivial features, since any single input is likely to be corrupted by noise. The learning algorithms for RBMs have built-in random sampling, which has a similar effect (Vincent, 2011).

If inputs are natural images, we can further exploit their two-dimensional structure to regularize learning. Pixels have much stronger correlation with their immediate neighbors than with far-away ones (Huang and Mumford, 1999). As a consequence, image features are strongly localized, i.e. mostly zero, and local weight matrices can be used which assign each feature a set of non-zero weights only in a small image region.

Even with local weight matrices, we can anticipate that the filters will be very redundant. All edge features, to choose a simple example, need to be learned at all image locations. LeCun et al (1998) introduced *weight sharing* to eliminate this redundancy. Every feature detector is applied at all image locations, which can be efficiently implemented using convolutions instead of the matrix multiplication in Eq. (1). Absolute image position can still be incorporated by adding non-convolutional layers at later stages in the hierarchy.

5 Applications

Deep learning techniques have been applied to numerous domains and spurred a plethora of extensions. In this section, we highlight a few approaches selected to give an impression of the broad domain of applications and the vitality of deep learning.

In their seminal work, Hinton and Salakhutdinov (2006) showed that the deep learning approach with

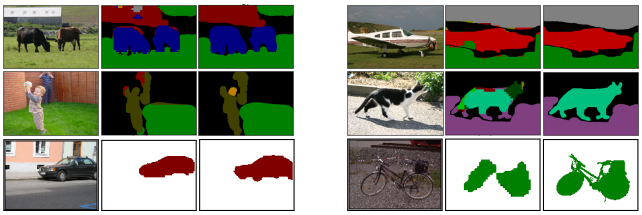

Fig. 4: Example segmentations by Schulz and Behnke (2012) for three datasets: MSRC-9 (top row), MSRC-21 (center row), and IG02 (bottom row). Left column shows original image, center column our output, right column ground truth. MSRC has ground mask of unlabeled pixels from ground truth superimposed.

unsupervised RBM pre-training is useful for compression of handwritten digits and faces. For this purpose, the authors pre-trained a feature hierarchy with depth $L = 4$ using RBMs and then fine-tuned for a reconstruction task. The same unsupervised learning technique was also successful in embedding text documents in a low-dimensional space where distances then corresponded to given labels better than for other methods.

Lee et al (2009a) extended the RBM to convolutional weights for image recognition. To increase translational invariance, the authors generalized maximum-pooling of convolutional feed-forward neural networks (Scherer et al, 2010) to probabilistic max-pooling, which can be used in generative models. The authors used their model to generate missing parts in images of faces and for classification.

Varying the type of the learned feature, Ranzato and Hinton (2010) extended the restricted Boltzmann machine to model not only pixel means, but also covariances in images. To generate samples from the model distribution, block Gibbs sampling of standard RBMs cannot be used. Instead the authors rely on a hybrid Monte Carlo technique. After training their model on natural image patches, knowledge of few pixels was sufficient to fill out large image regions. The mean-covariance features can also be used as inputs to an RBM stack, resulting in improved performance on an image classification task.

Using purely supervised pre-training of a convolutional deep neural network, Grangier et al (2009), as well as Schulz and Behnke (2012) achieved very good performance on multiple object-class segmentation benchmarks. Here, the task is to label each pixel in an image with one of a given set of object class labels. Schulz and Behnke improved the method by reusing the not-yet-optimal pre-training results of lower layers as inputs for higher layers. Figure 4 shows sample segmentations.

For natural language processing, Collobert and Weston (2008) proposed to use a deep neural network to simultaneously perform part-of-speech tagging, determining semantic roles, and semantic similarity. Similar to images, the authors use convolutions to become independent of sentence length. By repeatedly marking single input words as "to be labeled", whole sentences can be interpreted by the network. To optimize all objectives at once, the authors share parameters between the various classifiers and observe that task performances improve when multi-task learning is used.

Lee et al (2009b) apply deep learning principles to the audio domain. While in images, learned features correspond to edges, here the features represent phones and phonemes and improve the performance in multiple audio recognition tasks. More recently, Dahl et al (2012) built upon previous work on the combination of artificial neural networks for feature extraction and hidden Markov models (HMM) for context. By choosing deep (up to five layer) neural networks instead of shallow ones and by employing RBM pre-training, they significantly out-performed state-of-the-art on large-vocabulary speech recognition. They further boosted their results by enlarging their set of labels for supervised fine-tuning using the posterior probabilities of senomes (learned tied triphone HMM states), which significantly outnumber the set of classical phonemes.

On a variety of benchmark datasets for image classification, the currently best-performing approach is the multi-column deep neural network by Cireşan et al (2012). Here, very deep convolutional networks are trained on random variations of the training set. By carefully handcrafting a model of transformations conforming with the input distribution, ten-layer architectures could be learned even without pre-training. Additional gains were achieved by training multiple networks ("columns"), each with a different preprocessing procedure, and averaging the results.

6 Conclusion

We gave a brief overview of the ideas behind deep learning, a field of machine learning creating and analyzing the building blocks of feature hierarchies.

Feature hierarchies provide a space and time-efficient decomposition of inputs, which can be useful in various tasks such as classification, denoising, and compression. For a long time it was not clear how feature hierarchies can be learned. We discussed the problems encountered during learning—the large number of local minima and gradient dilution.

The deep learning solution to learning feature hierarchies is to solve a sequence of simple shallow problems first. In each step, deep methods learn a new level of features—gaining new insights into the input data distribution on the way. The resulting feature hierarchy can

finally be adopted to an arbitrary (usually supervised) task.

While deep learning has progressed tremendously over the last years, many challenges remain. The described building blocks of deep learning are restricted and cannot represent arbitrary features, since the encoder has no hidden layer. We believe that unsupervised pre-training of two-layer encoders is more promising for the future. The set of learned invariances should be extended to include transformations (Taylor et al, 2010; Memisevic, 2011) and hierarchies of transformations, so that image sequences can be modeled. Another open problem is to model the 3D structure of scenes to deal with occlusions. Finally, it is a challenge to scale up the proposed techniques to large real-world datasets such as ImageNet, and investigate whether the results can compete with current computer vision approaches there.

References

Behnke S (1999) Hebbian learning and competition in the neural abstraction pyramid. In: Proceedings of International Joint Conference on Neural Networks (IJCNN), Washington, DC, USA, vol 2, pp 1356–1361

Behnke S (2003a) Discovering hierarchical speech features using convolutional non-negative matrix factorization. In: Proceedings of International Joint Conference on Neural Networks (IJCNN), Portland, Oregon, USA, vol 4, pp 2758–2763

Behnke S (2003b) Hierarchical Neural Networks for Image Interpretation, Lecture Notes in Computer Science, vol 2766. Springer

Bengio Y, Lamblin P, Popovici D, Larochelle H (2006) Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems (NIPS), Vancouver, Canada, pp 153–160

Bottou L (2011) From machine learning to machine reasoning. Arxiv preprint arXiv:11021808

Boureau Y, Bach F, LeCun Y, Ponce J (2010) Learning mid-level features for recognition. In: Proceedings of Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, pp 2559–2566

Cireşan DC, Meier U, Masci J, Schmidhuber J (2012) Multi-column deep neural networks for image classification. Proceedings of Computer Vision and Pattern Recognition (CVPR) in press

Coates A, Lee H, Ng AY (2010) An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS), Chia Laguna, Italy

Collobert R, Weston J (2008) A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of International Conference on Machine Learning (ICML), Helsinki, Finland, pp 160–167

Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems (MCSS) 2(4):303–314

Dahl G, Yu D, Deng L, Acero A (2012) Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. IEEE Transactions on Audio, Speech, and Language Processing 20(1):30–42

Erhan D, Manzagol P, Bengio Y, Bengio S, Vincent P (2009) The difficulty of training deep architectures and the effect of unsupervised pre-training. In: Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS), Clearwater Beach, FL, USA, pp 153–160

Erhan D, Bengio Y, Courville AC, Manzagol PA, Vincent P, Bengio S (2010) Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research 11:625–660

Fidler S, Leonardis A (2007) Towards scalable representations of object categories: Learning a hierarchy of parts. In: Proceedings of Computer Vision and Pattern Recognition (CVPR), Minneapolis, MN, USA

Fukushima K (1980) Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36(4):193–202

Grangier D, Bottou L, Collobert R (2009) Deep convolutional networks for scene parsing. In: ICML Deep Learning Workshop, Montreal, Canada

Hinton G (2002) Training products of experts by minimizing contrastive divergence. Neural Computation 14(8):1771–1800

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

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

Hochreiter S, Bengio Y, Frasconi P, Schmidhuber J (2001) Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: Kremer SC, Kolen JF (eds) A Field Guide to Dynamical Recurrent Neural Networks, Wiley-IEEE Press

Huang J, Mumford D (1999) Statistics of natural images and models. In: Proceedings of Computer Vision and Pattern Recognition (CVPR), Ft. Collins, CO, USA

Kavukcuoglu K, Ranzato M, LeCun Y (2010) Fast inference in sparse coding algorithms with applications to object recognition. CoRR abs/1010.3467

LeCun Y, Boser B, Denker J, Henderson D, Howard R, Hubbard W, Jackel L (1989) Backpropagation applied to handwritten zip code recognition. Neural computation 1(4):541–551

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. vol 86, pp 2278–2324

Lee H, Grosse R, Ranganath R, Ng A (2009a) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of International Conference on Machine Learning (ICML), New York, NY, USA, pp 609–616

Lee H, Pham P, Largman Y, Ng A (2009b) Unsupervised feature learning for audio classification using convolutional deep belief networks. In: Advances in Neural Information Processing Systems (NIPS), Vancouver, Canada, pp 1096–1104

Memisevic R (2011) Gradient-based learning of higher-order image features. In: Proceedings of International Conference on Computer Vision (ICCV), Barcelona, Spain, pp 1591–1598

Ranzato M, Hinton G (2010) Modeling pixel means and covariances using factorized third-order Boltzmann machines. In: Proceedings of Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, pp 2551–2558

Riesenhuber M, Poggio T (1999) Hierarchical models of object recognition in cortex. Nature Neuroscience 2:1019–1025

Rumelhart D, Hinton G, Williams R (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536

Scherer D, Müller A, Behnke S (2010) Evaluation of pooling operations in convolutional architectures for object recognition. In: Proceedings of International Conference on Artificial Neural Networks (ICANN), Thessaloniki, Greece, pp 92–101

Schulz H, Behnke S (2012) Learning object-class segmentation with convolutional neural networks. In: Proceedings of the European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium

Shannon C (1949) The synthesis of two-terminal switching circuits. Bell System Technical Journal 28(1):59–98

Taylor G, Fergus R, LeCun Y, Bregler C (2010) Convolutional learning of spatio-temporal features. Computer Vision–ECCV 2010 pp 140–153

Tieleman T (2008) Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proceedings of International Conference on Machine Learning (ICML), pp 1064–1071

Vincent P (2011) A connection between score matching and denoising autoencoders. Neural Computation 23(7):1661–1674

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

Weston J, Ratle F, Collobert R (2008) Deep learning via semi-supervised embedding. In: Proceedings of International Conference on Machine Learning (ICML), Helsinki, Finland, pp 1168–1175

Wiskott L, Sejnowski T (2002) Slow feature analysis: Unsupervised learning of invariances. Neural Computation 14(4):715–770

Zeiler M, Taylor G, Fergus R (2011) Adaptive deconvolutional networks for mid and high level feature learning. In: Proceedings of International Conference on Computer Vision (ICCV), Barcelona, Spain, pp 2018–2025