Learning Invariant Feature Hierarchies

Yann LeCun

Courant Institute, New York University

Abstract. Fast visual recognition in the mammalian cortex seems to be a hierarchical process by which the representation of the visual world is transformed in multiple stages from low-level retinotopic features to high-level, global and invariant features, and to object categories. Every single step in this hierarchy seems to be subject to learning. How does the visual cortex learn such hierarchical representations by just looking at the world? How could computers learn such representations from data? Computer vision models that are weakly inspired by the visual cortex will be described. A number of unsupervised learning algorithms to train these models will be presented, which are based on the sparse auto-encoder concept. The effectiveness of these algorithms for learning invariant feature hierarchies will be demonstrated with a number of practical tasks such as scene parsing, pedestrian detection, and object classification.

1 Introduction

The age-old architecture of pattern recognition systems is composed of two parts: a feature extractor and a classifier. The feature extractor is generally built "by hand", and transforms the raw input into a representation suitable for classification. The classifier, on the other hand, is fairly generic, and is the only trainable part of the system. Much effort and ingenuity has been devoted to designing appropriate feature extractors for particular problems and input modalities (see [1,2] for the most popular methods in vision).

In contrast, the ventral pathway of the visual cortex appears to comprise more than just two stages: the lateral geniculate nucleus which performs a sort of multi-scale high-pass filtering and contrast normalization, V1 which consists mainly of pooled oriented edge detectors with local receptive fields [3], V2 and V4 which seem to detect larger and more complex local motifs [4], and the inferro-temporal cortex in which object categories are encoded [5]. Although the visual cortex contains numerous feedback connections, fast object recognition appears to be an essentially feed-forward affair [6]. Every stage in the system seem subject to plasticity and learning, and the function of each area seems largely determined by its afferent signals through learning, as demonstrated by brain rewiring experiments [7].

Building a multi-stage recognition system in which all the stages are trained has been a long-term interest of mine since I started working on pattern recognition in the early 1980's. The obvious advantage is that this would reduce the amount of "manual" labor by leaving the design of the feature extractor to the learning algorithm. The feature extractor could be the optimally tuned to the task at hand. But the idea of *feature learning* has encountered a surprisingly large amount of resistance from the computer vision and

A. Fusiello et al. (Eds.): ECCV 2012 Ws/Demos, Part I, LNCS 7583, pp. 496–505, 2012.

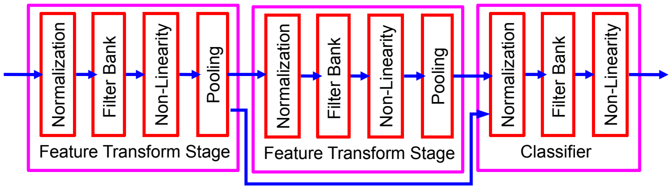

Fig. 1. A general multi-stage architecture for hierarchical feature learning and extraction

machine learning communities. The machine learning (ML) community has had a good handle on how to build classifiers for quite a while now. But the problem of *learning internal representations of the perceptual world* has received little attention until very recently. How do we train multi-stage systems that include the feature extractor, classifier, and high-level contextual post-processor in an integrated fashion? This question, which was already identified in the early 60's, has recently seen an explosive resurgence of interest, and has come to be known as the *deep learning* problem.

In the mid 1980's, there was a hope that neural nets with hidden units (Boltzmann machines or feed-forward neural nets) would offer a solution to the feature learning problem. But training generic neural networks was a complex and time consuming affair with the computers and software tools of the time, and given the rarity of large datasets. Furthermore, the basic architectural components used in traditional neural nets didn't seem appropriate for image recognition. One must design the architecture of the system to take advantage of the properties of the signal.

1.1 A General Architecture for Hierarchical Processing

A particularly important question is how to design the architecture so that the system can easily learn representations that are *invariant* (or robust) to irrelevant variations of the input. In the case of images, that includes translation, scaling, mild rotation, illumination, etc. A number of early researchers were inspired by Hubel and Wiesel's seminal work on the primary visual cortex, and their simple-cell/complex-cell model [3].

A general multi-stage architecture for hierarchical feature learning is shown in figure 1 (a 3-stage system is shown here). Each stage conforms to the simple-cell/complex-cell idea: (1) a normalization layer, which can be a whitening operation (e.g. ZCA), or in the case of spatial signals a high-pass filtering with local energy normalization at a single scale or multiple scales (Laplacian pyramid); (2) a linear filtering layer, which can be seen as a matrix or as a bank of convolution filters; (3) a fixed non-linear transformation layer, which can be a point-wise non-linear mapping (e.g. logistic, $\tanh$, shrinking function, or half-rectifier), or something harsher such as a multinomial logistic or a winner-take-all; (4) a pooling layer, which aggregates a subset of values from the previous layer, and generally reduces the dimensionality of the representation though subsampling. Layer 1 is analogous to the LGN, layers 2 and 3 to groups of simple cells, and layer 4 to groups of complex cells. Multiple such 4-layer stages can be cascaded (typically 2 to 5). The last stage may often dispenses with the pooling layer and can be viewed as a classifer (or predictor), operating on the features extracted by the previous stages.

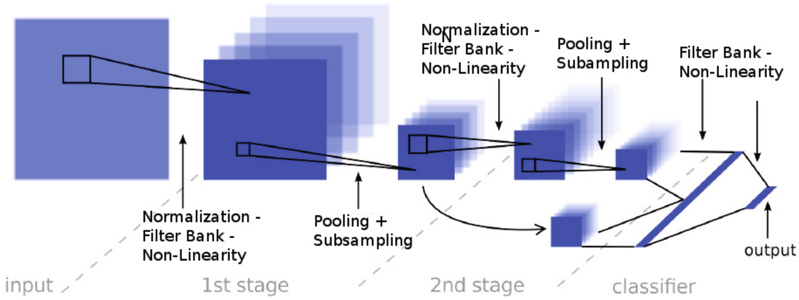

Fig. 2. A convolutional network architecture, which is a particular instance of the multi-stage architecture shown above

The role of layer 1 is to decorrelate variables and accentuate the differences (or ratios) between them, while eliminating variations of the absolute energy so that the non-linearity of layers 3 can always operate at its sweet spot. Decorrelation (and mean removal) has the additional advantage of accelerating gradient-based learning [8].

Layer 2 and 3 detect conjunctions of features or motifs on the previous stage. Its role is to non-linearly embed the input into a higher-dimensional space, so that inputs that are semantically different are likely to be represented by different patterns of activity. This expansion plays a similar role as using a non-linear kernel functions in a kernel machine: in high-dimensional spaces, categories are easier to separate. More generally, a function of interest is more likely to be linear when its input variable is embedded in a high dimensional space. The difference with kernel machine is that our filter banks will be trained from data, rather than simply selected from the training set.

Layer 4 serves to *merge semantically similar things* that have been partitioned into different patterns of activity by the simple cells. This is where invariance is built. Rather than producing invariance in the mathematical sense, the pooling layer merely "smoothes out" the input-output mapping so that irrelevant variations in the input affect the output smoothly, and in ways that can be easily dealt with (eliminated, if necessary). The pooling operation can consist of any symmetric aggregation function, such as an average, a max, a log-mixture ($\log \sum_i e^{x_i}$), or an L_p norm ($\sqrt[p]{\sum_i |x_i|^p}$), particularly with $p = 1, 2$, or ∞ (max). A theoretical analysis of pooling operations suggests that L_∞ is best when the features are sparse and the number of pooled variable is small, while average, L_1 or L_2 are best when the features are less sparse or the pooling area is large [9]. In practice L_2 pooling is a good tradeoff.

One may interpret the filter bank and non-linearity as conjunction operators (similar to logical `AND` or `NAND` in the boolean case) and the pooling operation as a sort of disjunction operator (similar to a logical `OR`), making a single stage a kind of non-boolean Disjunctive Normal Form.

1.2 Convolutional Architectures

Data from natural sensors often comes to us as multi-dimensional arrays in which local group of values are correlated, and the local statistics are invariant to the particular location in the array. For example, images can be seen as a series of 2D slices where each slice is a color channel, and the dimensions are spatial. The statistics of images are translation invariant, which means that if one particular filter is useful on one part of an image, it is probably useful on other parts of the image as well. This leads to the *convolutional network architecture* shown in figure 2 [10,11,12]. The filter bank in each stage is, in fact, a bank of convolution kernels applied to slices of the input. Filter outputs applied to multiple input channels can be combined additively to form a slice of the output (known as a feature map). The idea applies to other modalities than image, including audio, a (1+1)D array with one dimension being frequency channels and one being time, video, a (1+3)D array with color channels, time, and space. Other modalities such as RGB+Depth, sonar, radar, lidar, multi-spectral images, etc can be handled similarly.

The architecture shown in figure 2 has a structure reminiscent of the LGN-V1-V2-V4-IT hierarchy in the ventral pathway of the visual cortex. The simple cells have local receptive fields and are organized in a retinotopic fashion. The pooling layers are subsampled spatially, which reduces the spatial resolution of the representation and makes the representation vary smoothly with translations and small distortions of the input. Units have local receptive fields whose size increases as we move up in the hierarchy [13].

Supervised training of convolutional nets is performed using a form of stochastic gradient descent to minimize the discrepancy between the desired output and the actual output of the network. All the coefficient of all the filters in all the layers are updated simultaneously by the learning procedure. The gradients are computed with the back-propagation method.

A number of similar models have been proposed with different learning algorithms, starting with Fukushima's Neocognitron [14], and several others in the last 10 years [15,16]. It is important to notice that the most popular feature extraction methods in computer vision, SIFT [17] and HoG [18], are very much inspired by the simple-cell/complex-cell concept and conform to the first stage of figure 1. Moreover, the most popular standard recognition pipelines [19,20] conforms to the 3-stage architecture of figure 1: the first stage is SIFT, densely extracted over the input (similar to a convolutional network), the second stage filters are trained in an unsupervised manner with K-means (the non-linearity is a winner-take-all), and the pooling is an average over multiple scales. Finally the classifier is a support vector machine.

There have been numerous applications of convolutional networks going back to the mid 1990's for such applications as OCR and handwriting recognition, face, person, and license plate detection, age and gender estimation, video surveillance, etc (see [21] and references therein). They have been deployed by companies such as Google, Microsoft, AT&T, and NEC. In recent months, deep learning systems have broken long-held record on a number of benchmarks in a number of areas including speech recognition [22,23,24], object recognition vision [25,26,27], scene parsing [28], action recognition in video [29,30], natural language processing [31,32], and musical genre recognition [33].

2 Unsupervised Feature Learning

The resurgence of interest in deep architectures was caused in part by the appearance of unsupervised learning algorithms to train (or pre-train) the layers of a multi-stage system. The basic procedure is to pre-train each stage of a network in an unsupervised manner one after the other. After all the stages have been pre-trained, the entire network is fine-tuned using supervised learning (with gradient back-propagation). This procedure has two advantages: (1) unsupervised pre-training seems to place the system in a favorable starting point for supervised fine-tuning that will produce better performance results; (2) unsupervised learning leverages the availability of massive amounts of unlabeled data. Unsupervised pre-training on unlabeled data seems to "consume" a large amount of free parameters in the model, and allows us to use very large and flexible networks that would be hopelessly over-parameterized in a purely-supervised setting.

There are many methods to pre-train the filter banks of a multi-stage system in unsupervised mode, including (from the simple to the complicated): simply using randomly picked samples as filters; using K-means to produce prototypes and using them as filters [34]; using dictionary learning in sparse coding and use the basis functions as filters [35,36,37,38,12,39,34]; train some version of regularized auto-encoder such as *sparse auto-encoder*, denoising auto-encoder [40] or contracting auto-encoder [41]; train a restricted Boltzmann machine and use the weight matrix as filters [42] (see [43] for a recent review).

We will concentrate on the concept of sparse auto-encoder, as implemented in the Predictive Sparse Decomposition (PSD) method [37,38,39]. PSD is based on the classical sparse coding and sparse modeling algorithm [44]: a vector (e.g. an image patch) Y is encoded as a feature vector Z^* by minimizing the following energy function:

$$E(Y,Z,W_d) = ||Y - W_d Z||^2 + \alpha \sum_k |Z_k| \qquad Z^* = \text{argmin}_Z E(Y,Z,W) \tag{1}$$

where W_d is a so-called *dictionary matrix* whose columns W_{dk} are called atoms or basis functions. The vector Z has generally higher dimension than Y (overcomplete representation). The minimization procedure will find a small number of column of W_d that can be linearly combined to reconstruct Y. The coefficients are the components of Z, many of which will be zero, due to the sparsity-inducing L_1 regularization term. Given a training set of input vectors $Y^i,\ i = 1 \ldots P$, learning W_d can be performed using stochastic gradient descent on W_d to minimize the reconstructive loss function $L_R(W) = \sum_{i=1}^P \min_z E(Y^i, Z, W)$, under the constraints that the columns of W_d be within the unit sphere.

Extracting features with sparse coding work beautifully as a feature extraction method when the dictionary matrix is trained with sparse modeling, particularly for learning mid-level features for object recognition [45,20]. But inferring Z^* from a particular Y is slow, because it involves minimizing the L1/L2 energy function above.

One solution to this problem is the PSD method, which consist in training an *parameterized non-linear encoder function* $g(Y, W_e)$ where W_e is *encoder matrix or filter matrix*, so as to best predict the optimal sparse feature vector Z^* for all training samples Y^i. The rows of W_e can be interpreted as a filter bank. Once the decoder and the encoder are trained, the decoder can be dispensed with, and the encoder used as a feture extractor (filter bank + non-linearity). The training can be performed by minimizing the predictive loss function

$$L_P(W_e) = \sum_{i=1}^{p} || \min_Z E(Y^i, Z, W_d) - g(Y^i, W_e)||^2.$$

Better yet, one can define a compound energy function

$$E_{PSD}(Y, Z, W_d, W_e) = ||Y - W_d Z||^2 + ||Z - g(Y, W_e)||^2 + \alpha \sum_k |Z_k|,$$

and train W_d and W_e to minimize the PSD loss function:

$$L_{PSD}(W_d, W_e) = \sum_{i=1}^{p} \min_Z E_{PSD}(Y^i, Z, W_d, W_e).$$

The encoder architecture can be a simple function such as $\text{shrink}_\beta(W_e Y)$, where shrink_β is the shrinking function applied indepently to each component of $W_e Y$:

$$\text{shrink}_\beta(x) = [0 \text{ if } -\beta < x < \beta;\ x - \beta \text{ if } x > \beta;\ x + \beta \text{ if } x < -\beta]$$

A more sophisticated version of PSD [39] uses a parameterized version of the unfolded flow graph of the FISTA algorithm for sparse coding [46]. This produces better sparse codes, but it's not clear whether the resulting feature vectors are better for recognition purpose (except perhaps in the highly over-complete case).

When trained on natural image patches, PSD (like sparse modeling) produces Gabor-like oriented filters (edgelets) at various orientation, frequencies, and positions within the patch (when trained to produce 2nd-stage features, the interpretation is considerably less clear). However, since the filters trained in this manner are meant to be used in a convolutional manner, it would seem appropriate to train them convolutionally. Training PSD (or sparse coding) at the patch level is inefficient, since shifted versions of each filter must be generated to reconstruct each image patch. Convolutional sparse coding [47] and convolutional PSD [48] solve that problem by viewing the reconstruction as multiple convolutions

$$E(Y, Z, W_d) = ||Y - \sum_k W_{dk} * Z_k||^2 + \alpha \sum_{kpq} |Z_{kpq}|$$

where Z_k is a *feature map* (an image about the same size as image Y) instead of a scalar, W_{dk} is a convolution kernel, and $*$ is the convolution operator (the predictive energy term was left out). Convolutional PSD produces much more diverse filters than patch-based PSD, as shown in figure 3.

A convolutional network pre-trained with Convolutional PSD generally yields better performance than if the network is trained purely supervised.

3 Unsupervised Invariant Feature Learning

The PSD training procedure is quite effective, but doesn't take into account the fact that the features it produces are aggregated by the pooling layer (complex cells). Invariant

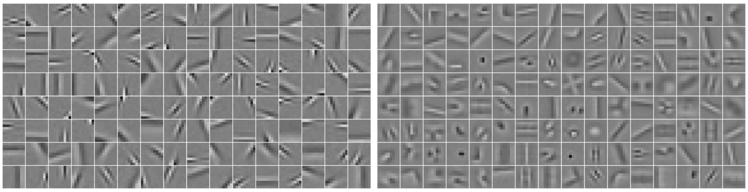

Fig. 3. Filters obtained with patch-based PSD (left) and convolutional PSD (right). Each square filter is a row of the encoder matrix W_e. The convolutional version produces much more diverse and less redundant filters.

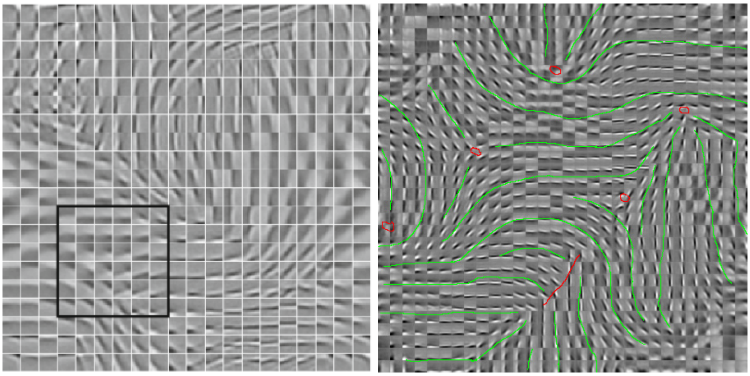

Fig. 4. Left: topographic maps of filter produced by Invariant PSD when the groups (complex cells) are sqaure blocks of feature components in this 2D topology; Right: pinwheel-like patterns obtained when training PSD with group sparsity on 4X overcomplete units with local 15x15 receptive fields without shared weights.

PSD is an attempt to integrate the complex cells in the unsupervised training. This can be done with the idea of group sparsity [38,49,50], which is reminiscent of sub-space ICA [51] and Product of Experts [52]. the main idea is to replace the sparsity penalty in the PSD energy function by a group sparsity penalty:

$$E_{PSD}(Y, Z, W_d, W_e) = ||Y - W_d Z||^2 + ||Z - g(Y, W_e)||^2 + \alpha \sum_g \sqrt{\sum_{k \in g} Z_k^2},$$

where g is an index over groups of components of Z (possibly overlapping groups) The output of each group can be seen as an L_2 pooling of the components of Z that belong to the group, and is akin to a complex cell. With this regularizer, the system tries to activate the smallest number of groups, but may allow multiple units within a group to be active. This causes filters within a group to be similar to each other because similar filters tend to be active simultaneously (e.g. edge detectors at similar orientations). When the components of Z are arragend in a particular topology (e.g. 2D torus) and the groups are overlaping regions in that topology, the filters organize themselves into topographic maps similar to what is observed in V1. Figure 4 shows examples of such topographic maps of filters.

4 Conclusion

Despite the ménagerie of unsupervised feature learning algorithms at our disposal, few would say that we have the perfect algorithm in our hands. Although some biologically-inspired algorithms, such as convolutional networks and their variations, produce record-breaking results on practical datasets such as scene parsing, the ability of these algorithms to learn is still far from what is observed in humans and animals. One must ask whether there exist a simple learning "algorithm" used by the cortex, or if there is a simple principle on which such an algorithm could be based. It is certainly worth looking for such a principle.

Acknowledgments. The author wishes to thank Y-Lan Boureau, Clément Farabet, Karol Gregor, Kevin Jarrett, Koray Kavukcuoglu, Marc'aurelio Ranzato, and Arthur Szlam. Most of the work described in this paper is theirs. This work was supported in part by ONR, NSF, and DARPA.

References

1. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV 60(4), 91–110 (2004)
2. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection, vol. 2, pp. 886–893 (June 2005)
3. Hubel, D.H., Wiesel, T.N.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J. Physiol. 160, 106–154 (1962)
4. Hansen, K.A., Kay, K.N., Gallant, J.L.: Topographic organization in and near human visual area v4. Journal of Neuroscience 27, 11896–11911 (2007)
5. Tanaka, K.: Inferotemporal cortex and object vision. Annual Review of Neuroscience 19, 109–139 (1996)
6. Thorpe, S., Fize, D., Marlot, C.: Speed of processing in the human visual system. Nature 381(6582), 520–522 (1996)
7. Sur, M., Garraghty, P.E., Roe, A.W.: Experimentally induced visual projections into auditory thalamus and cortex. Science 242(4884), 1437–1441 (1988)
8. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient BackProp. In: Orr, G.B., Müller, K.-R. (eds.) NIPS-WS 1996. LNCS, vol. 1524, pp. 9–50. Springer, Heidelberg (1998)
9. Boureau, Y., Ponce, J., LeCun, Y.: A theoretical analysis of feature pooling in vision algorithms. In: Proc. International Conference on Machine learning, ICML 2010 (2010)
10. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: Touretzky, D. (ed.) Advances in Neural Information Processing Systems (NIPS 1989), Denver, CO, vol. 2. Morgan Kaufman (1990)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
12. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: Proc. International Conference on Computer Vision (ICCV 2009). IEEE (2009)
13. Freeman, J., Simoncelli, E.P.: Metamers of the ventral stream. Nature Neuroscience 14(9), 1195–1201 (2011)

14. Fukushima, K., Miyake, S.: Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. Pattern Recognition 15, 455–469 (1982)
15. Serre, T., Wolf, L., Poggio, T.: Object recognition with features inspired by visual cortex. In: CVPR (2005)
16. Pinto, N., Cox, D.D., DiCarlo, J.J.: Why is real-world visual object recognition hard? PLoS Comput. Biol. 4(1), e27 (2008)
17. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision (2004)
18. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proc. of Computer Vision and Pattern Recognition (2005)
19. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR (2006)
20. Boureau, Y., Bach, F., LeCun, Y., Ponce, J.: Learning mid-level features for recognition. In: Proc. International Conference on Computer Vision and Pattern Recognition (CVPR 2010). IEEE (2010)
21. LeCun, Y., Kavukvuoglu, K., Farabet, C.: Convolutional networks and applications in vision. In: Proc. International Symposium on Circuits and Systems (ISCAS 2010). IEEE (2010)
22. Dahl, G., Ranzato, M., Mohamed, A., Hinton, G.E.: Phone recognition with the mean-covariance restricted boltzmann machine. In: Advances in Neural Information Processing Systems, vol. 23, pp. 469–477 (2010)
23. Dahl, G., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. IEEE Transactions on Audio, Speech, and Language Processing (2012)
24. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine (in press, 2012)
25. Boureau, Y., Le Roux, N., Bach, F., Ponce, J., LeCun, Y.: Ask the locals: multi-way local pooling for image recognition. In: Proc. International Conference on Computer Vision, ICCV 2011 (2011)
26. Le, Q., Monga, R., Devin, M., Corrado, G., Chen, K., Ranzato, M., Dean, J., Ng, A.: Building high-level features using large scale unsupervised learning. In: Proceedings of ICML 2012 (2012)
27. Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Improving neural networks by preventing co-adaptation of feature detectors. ArXiv e-prints (July 2012)
28. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Scene parsing with multiscale feature learning, purity trees, and optimal covers. In: Proc. International Conference on Machine learning, ICML 2012 (2012)
29. Taylor, G.W., Fergus, R., LeCun, Y., Bregler, C.: Convolutional Learning of Spatio-temporal Features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 140–153. Springer, Heidelberg (2010)
30. Le, Q., Zou, W., Yeung, S., Ng, A.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3361–3368. IEEE (2011)
31. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research 12, 2493–2537 (2011)
32. Glorot, X., Bordes, A., Bengio, Y.: Domain adaptation for large-scale sentiment classification: A deep learning approach. In: Proceedings of the Twenty-eight International Conference on Machine Learning (ICML 2011), vol. 27, pp. 97–110 (2011)

33. Henaff, M., Jarrett, K., Kavukcuoglu, K., LeCun, Y.: Unsupervised learning of sparse features for scalable audio classification. In: Proceedings of International Symposium on Music Information Retrieval, ISMIR 2011 (2011) (Best Student Paper Award)
34. Coates, A., Ng., A.Y.: Selecting receptive fields in deep networks. In: Neural Information Processing Systems 24, NIPS 2011 (2011)
35. Ranzato, M., Poultney, C., Chopra, S., LeCun, Y.: Efficient learning of sparse representations with an energy-based model. In: Platt, J., et al. (eds.) Advances in Neural Information Processing Systems (NIPS 2006), vol. 19. MIT Press (2006)
36. Ranzato, M., Boureau, Y., LeCun, Y.: Sparse feature learning for deep belief networks. In: Advances in Neural Information Processing Systems (NIPS 2007), vol. 20 (2007)
37. Kavukcuoglu, K., Ranzato, M., LeCun, Y.: Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU (2008), Tech Report CBLL-TR-2008-12-01, http://arxiv.org/abs/1010.3467
38. Kavukcuoglu, K., Ranzato, M., Fergus, R., LeCun, Y.: Learning invariant features through topographic filter maps. In: Proc. International Conference on Computer Vision and Pattern Recognition (CVPR 2009). IEEE (2009)
39. Gregor, K., LeCun, Y.: Learning fast approximations of sparse coding. In: Proc. International Conference on Machine learning, ICML 2010 (2010)
40. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 11
41. Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contracting auto-encoders: Explicit invariance during feature extraction. In: Proceedings of the Twenty-eight International Conference on Machine Learning, ICML 2011 (June 2011)
42. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
43. Bengio, Y., Courville, A.C., Vincent, P.: Unsupervised feature learning and deep learning: A review and new perspectives. CoRR abs/1206.5538 (2012)
44. Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: a strategy employed by v1? Vision Research 37, 3311–3325 (1997)
45. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR (2009)
46. Beck, A., Teboulle, M.: A fast iterative shrinkage thresholding algorithm with application to wavelet-based image deblurring. In: ISCASSP (2009)
47. Zeiler, M., Krishnan, D., Taylor, G., Fergus, R.: Deconvolutional networks. In: CVPR (2010)
48. Kavukcuoglu, K., Sermanet, P., Boureau, Y., Gregor, K., Mathieu, M., LeCun, Y.: Learning convolutional feature hierachies for visual recognition. In: Advances in Neural Information Processing Systems (NIPS 2010), vol. 23 (2010)
49. Mairal, J., Jenatton, R., Obozinski, G., Bach, F.: Convex and network flow optimization for structured sparsity. Journal of Machine Learning Research 12, 2681–2720 (2011)
50. Le, Q.V., Ranzato, M.A., Monga, R., Devin, M., Chen, K., Corrado, G.S., Dean, J., Ng, A.Y.: Building high-level features using large scale unsupervised learning. In: Proceedings of the Twenty-ninth International Conference on Machine Learning, ICML 2012 (2012)
51. Hyvarinen, A., Hoyer, P.: A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images. Vision Research 41(8), 2413–2423 (2001)
52. Osindero, S., Welling, M., Hinton, G.E.: Topographic product models applied to natural scene statistics. Neural Comput. 18(2), 381–414 (2006)