Convolutional Networks and Applications in Vision

Yann LeCun, Koray Kavukcuoglu and Clément Farabet
Computer Science Department, Courant Institute of Mathematical Sciences, New York University
{yann,koray,cfarabet}@cs.nyu.edu

***Abstract*— Intelligent tasks, such as visual perception, auditory perception, and language understanding require the construction of good internal representations of the world (or "features"), which must be invariant to irrelevant variations of the input while, preserving relevant information. A major question for Machine Learning is how to *learn* such good features automatically. Convolutional Networks (ConvNets) are a biologically-inspired trainable architecture that can learn invariant features. Each stage in a ConvNets is composed of a filter bank, some non-linearities, and feature pooling layers. With multiple stages, a ConvNet can learn multi-level hierarchies of features. While ConvNets have been successfully deployed in many commercial applications from OCR to video surveillance, they require large amounts of labeled training samples. We describe new unsupervised learning algorithms, and new non-linear stages that allow ConvNets to be trained with very few labeled samples. Applications to visual object recognition and vision navigation for off-road mobile robots are described.**

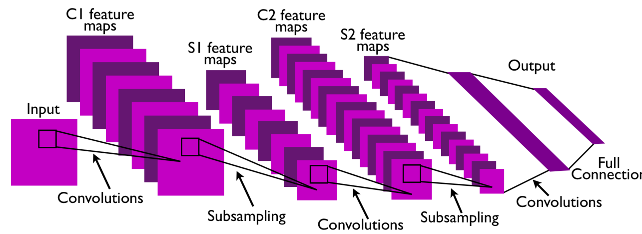

Fig. 1. A typical ConvNet architecture with two feature stages

I. Learning Internal Representations

One of the key questions of Vision Science (natural and artificial) is how to produce good internal representations of the visual world. What sort of internal representation would allow an artificial vision system to detect and classify objects into categories, independently of pose, scale, illumination, conformation, and clutter? More interestingly, how could an artificial vision system *learn* appropriate internal representations automatically, the way animals and human seem to learn by simply looking at the world? In the time-honored approach to computer vision (and to pattern recognition in general), the question is avoided: internal representations are produced by a hand-crafted feature extractor, whose output is fed to a trainable classifier. While the issue of learning features has been a topic of interest for many years, considerable progress has been achieved in the last few years with the development of so-called *deep learning* methods.

Good internal representations are hierarchical. In vision, pixels are assembled into edglets, edglets into motifs, motifs into parts, parts into objects, and objects into scenes. This suggests that recognition architectures for vision (and for other modalities such as audio and natural language) should have multiple trainable stages stacked on top of each other, one for each level in the feature hierarchy. This raises two new questions: what to put in each stage? and how to train such *deep, multi-stage architectures*? Convolutional Networks (ConvNets) are an answer to the first question. Until recently, the answer to the second question was to use gradient-based supervised learning, but recent research in *deep learning* has produced a number of unsupervised methods which greatly reduce the need for labeled samples.

Convolutional Networks

Convolutional Networks [1], [2] are trainable multistage architectures composed of multiple stages. The input and output of each stage are sets of arrays called *feature maps*. For example, if the input is a color image, each feature map would be a 2D array containing a color channel of the input image (for an audio input each feature map would be a 1D array, and for a video or volumetric image, it would be a 3D array). At the output, each feature map represents a particular feature extracted at all locations on the input. Each stage is composed of three layers: a *filter bank layer*, a *non-linearity layer*, and a *feature pooling layer*. A typical ConvNet is composed of one, two or three such 3-layer stages, followed by a classification module. Each layer type is now described for the case of image recognition.

Filter Bank Layer - F: the input is a 3D array with n_1 2D *feature maps* of size $n_2 \times n_3$. Each component is denoted x_{ijk}, and each feature map is denoted x_i. The output is also a 3D array, y composed of m_1 feature maps of size $m_2 \times m_3$. A trainable filter (kernel) k_{ij} in the filter bank has size $l_1 \times l_2$ and connects input feature map x_i to output feature map y_j. The module computes $y_j = b_j + \sum_i k_{ij} * x_i$ where $*$ is the 2D discrete convolution operator and b_j is a trainable bias parameter. Each filter detects a particular feature at every location on the input. Hence spatially translating the input of a feature detection layer will translate the output but leave it otherwise unchanged.

Non-Linearity Layer: In traditional ConvNets this simply consists in a pointwise $\tanh()$ sigmoid function applied to each site (ijk). However, recent implementations have used more sophisticated non-linearities. A useful one for natural image recognition is the rectified sigmoid R_{abs}: $\mathrm{abs}(g_i.tanh())$ where g_i is a trainable gain parameter. The rectified sigmoid is sometimes followed by a subtractive and divisive local normalization N, which enforces local competition between adjacent features in a feature map, and between features at the same spatial location. The subtractive normalization operation for a given site x_{ijk} computes: $v_{ijk} = x_{ijk} - \sum_{ipq} w_{pq}.x_{i,j+p,k+q}$, where w_{pq} is a normalized truncated Gaussian weighting window (typically of size 9x9). The divisive normalization computes $y_{ijk} = v_{ijk}/\max(mean(\sigma_{jk}), \sigma_{jk})$ where $\sigma_{jk} = (\sum_{ipq} w_{pq}.v^2_{i,j+p,k+q})^{1/2}$. The local contrast normalization layer is inspired by visual neuroscience models [3], [4].

Feature Pooling Layer: This layer treats each feature map separately. In its simplest instance, called P_A, it computes the average values over a neighborhood in each feature map. The neighborhoods are stepped by a stride larger than 1 (but smaller than or equal the pooling neighborhood). This results in a reduced-resolution output feature map which is robust to small variations in the location of features in the previous layer. The average operation is sometimes replaced by a max P_M. Traditional ConvNets use a pointwise $tanh()$ after the pooling layer, but more recent models do not. Some ConvNets dispense with the separate pooling layer entirely, but use strides larger than one in the filter bank layer to reduce

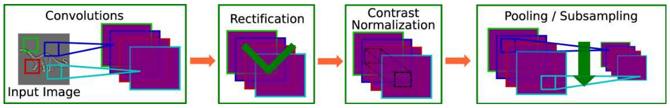

Fig. 2. An example of feature extraction stage of the type $F-R_{abs}-N-P_A$. An input image (or a feature map) is passed through a filter bank, followed by $\text{abs}(g_i.\tanh())$, local subtractive and divisive contrast normalization, and spatial pooling/sub-sampling.

the resolution [5], [6]. In some recent versions of ConvNets, the pooling also pools similar feature at the same location, in addition to the same feature at nearby locations [7].

Supervised training 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. Details of the procedure are given in [2], and methods for efficient training are detailed in [8].

History and Applications

ConvNets can be seen as a representatives of a wide class of models that we will call *Multi-Stage Hubel-Wiesel Architectures*. The idea is rooted in Hubel and Wiesel's classic 1962 work on the cat's primary visual cortex. It identified orientation-selective *simple cells* with local receptive fields, whose role is similar to the ConvNets filter bank layers, and *complex cells*, whose role is similar to the pooling layers. The first such model to be simulated on a computer was Fukushima's Neocognitron [9], which used a layer-wise, unsupervised competitive learning algorithm for the filter banks, and a separately-trained supervised linear classifier for the output layer. The innovation in [5], [1] was to simplify the architecture and to use the back-propagation algorithm to train the entire system in a supervised fashion. The approach was very successful for such tasks as OCR and handwriting recognition. An operational bank check reading system built around ConvNets was developed at AT&T in the early 1990's [2]. It was first deployed commercially in 1993, running on a DSP board in check-reading ATM machines in Europe and the US, and was deployed in large bank check reading machines in 1996. By the late 90's it was reading over 10% of all the checks in the US. This motivated Microsoft to deploy ConvNets in a number of OCR and handwriting recognition systems [6], [10], [11] including for Arabic [12] and Chinese characters [13]. Supervised ConvNets have also been used for object detection in images, including faces with record accuracy and real-time performance [14], [15], [16], [17], Google recently deployed a ConvNet to detect faces and license plate in StreetView images so as to protect privacy [18]. NEC has deployed ConvNet-based system in Japan for tracking customers in supermarket and recognizing their gender and age. Vidient Technologies has developed a ConvNet-based video surveillance system deployed in several airports in the US. France Télécom has deployed ConvNet-based face detection systems for video-conference and other systems [15]. Other experimental detection applications include hands/gesture [19], logos and text [20]. A big advantage of ConvNets for detection is their computational efficiency: even though the system is trained on small windows, it suffices to extend the convolutions to the size of the input image and replicate the output layer to compute detections at every location. Supervised ConvNets have also been used for vision-based obstacle avoidance for off-road mobile robots [21]. Two participants in the recent DARPA-sponsored LAGR program on vision-based navigation for off-road robots used ConvNets for long-range obstacle detection [22], [23]. In [22], the system is pre-trained off-line using a combination of unsupervised learning (as described in section II) and supervised learning. It is then adapted on-line, as the robot runs, using labels provided by a short-range stereovision system (see videos at `http://www.cs.nyu.edu/ yann/research/lagr`). Interesting new applications include image restoration [24] and image segmentation, particularly for biological images [25]. The big advantage over MRFs is the ability to take a large context window into account. Stunning results were obtained at MIT for reconstructing neuronal circuits from an stack of brain slice images a few nanometer thick. [26].

Over the years, other instances of the Multi-Stage Hubel-Wiesel Architecture have appeared that are in the tradition of the Neocognitron: unlike supervised ConvNets, they use a combination of hand-crafting, and simple unsupervised methods to design the filter banks. Notable examples include Mozer's visual models [27], and the so-called HMAX family of models from T. Poggio's lab at MIT [28], [29], which uses hard-wired Gabor filters in the first stage, and a simple unsupervised random template selection algorithm for the second stage. All stages use point-wise non-linearities and max pooling. From the same institute, Pinto et al. [4] have identified the most appropriate non-linearities and normalizations by running systematic experiments with a a single-stage architecture using GPU-based parallel hardware.

II. Unsupervised Learning of ConvNets

Training deep, multi-stage architectures using supervised gradient back propagation requires many labeled samples. However in many problems labeled data is scarce whereas unlabeled data is abundant. Recent research in deep learning [30], [31], [32] has shown that *unsupervised learning* can be used to train each stage one after the other using only unlabeled data, reducing the requirement for labeled samples significantly. In [33], using abs and normalization non-linearities, unsupervised pre-training, and supervised global refinement has been shown to yield excellent performance on the Caltech-101 dataset with only 30 training samples per category (more on this below). In [34], good accuracy was obtained on the same set using a very different unsupervised method based on sparse Restricted Boltzmann Machines. Several works at NEC have also shown that using *auxiliary tasks* [35], [36] helps regularizing the system and produces excellent performance.

Unsupervised Training with Predictive Sparse Decomposition

The unsupervised method we propose, to learn the filter coefficients in the filter bank layers, is called Predictive Sparse Decomposition (PSD) [37]. Similar to the well-known sparse coding algorithms [38], inputs are approximated as a sparse linear combination of dictionary elements. In conventional sparse coding for any given input X, one needs to run an expensive optimization algorithm to find Z^* (the "basis pursuit" problem). PSD trains a feed-forward regressor (or *encoder*) $C(X, K)$ to quickly approximate the sparse solution Z^*. During training, the feature vector Z^* is obtained by minimizing:

$$E(Z, W, K) = \|X - WZ\|_2^2 + \lambda\|Z\|_1 + \|Z - C(X, K)\|_2^2$$

where W is the matrix whose columns are the dictionnary elements and K are the filters. For each training sample X, one first finds Z^* that minimizes E, then W and K are

TABLE I

AVERAGE RECOGNITION RATES ON CALTECH-101.

	$\mathbf{R_{abs} - N - P_A}$	$\mathbf{R_{abs} - P_A}$	$\mathbf{N - P_M}$	$\mathbf{P_A}$
$\mathbf{U^+}$	65.5%	60.5%	61.0%	32.0%
$\mathbf{R^+}$	64.7%	59.5%	60.0%	29.7%
$\mathbf{U}$	63.7%	46.7%	56.0%	9.1%
$\mathbf{R}$	62.9%	33.7%	37.6%	8.8%

adjusted by stochastic gradient descent to lower E. Once training is complete, the feature vector for a given input is simply obtained with $Z^* = C(X, K)$, hence the process is extremely fast (feed-forward).

Results on Object Recognition

In this section, various architectures and training procedures are compared to determine which non-linearities are preferable, and which training protocol makes a difference.

Generic Object Recognition using Caltech 101 Dataset: We use a two-stage system where, the first stage is composed of an F layer with 64 filters of size 9×9, followed by different combinations of non-linearities and pooling. The second-stage feature extractor is fed with the output of the first stage and extracts 256 output features maps, each of which combines a random subset of 16 feature maps from the previous stage using 9×9 kernels. Hence the total number of convolution kernels is $256 \times 16 = 4096$.

Table I summarizes the results for the experiments, where U and R denotes unsupervised pre-training and random initialization respectively, and $^+$ denotes supervised fine-tuning of the whole system.

1. Excellent accuracy of 65.5% is obtained using unsupervised pre-training and supervised refinement with abs and normalization non-linearities. The result is on par with the popular model based on SIFT and pyramid match kernel SVM [39]. It is clear that abs and normalization are cruciala for achieving good performance. This is an extremely important fact for users of convolutional networks, which traditionally only use $\tanh()$.

2. Astonishingly, *random filters without any filter learning whatsoever achieve decent performance* (62.9% for R), as long as abs and normalization are present ($R_{abs} - N - P_A$). A more detailed study on this particular case can be found in [33].

3. Comparing experiments from rows R vs R^+, U vs U^+, we see that supervised fine tuning consistently improves the performance, particularly with weak non-linearities.

4. It seems that unsupervised pre-training (U, U^+) is crucial when newly proposed non-linearities are not in place.

Handwritten Digit Classification using MNIST Dataset: Using the evidence gathered in previous experiments, we used a two-stage system with a two-layer fully-connected classifier. The two convolutional stages were pre-trained unsupervised, and refined supervised. An error rate of 0.53% was achieved ont he test set. To our knowledge, *this is the lowest error rate ever reported on the original MNIST dataset, without distortions or preprocessing*. The best previously reported error rate was 0.60% [32].

Connection with Other Approaches in Object Recognition

Many recent successful object recognition systems can also be seen as single or multi-layer feature extraction systems followed by a classifier. Most common feature extraction systems like SIFT [40], HoG [41] are composed of filterbanks (oriented edge detectors at multiple scales) followed by non-linearities (winner take all) and pooling (histogramming). A Pyramid Match Kernel (PMK) SVM [39] classifer can also be seen as another layer of feature extraction since it performs a K-means based feature extraction followed by local histogramming.

III. HARDWARE AND SOFTWARE IMPLEMENTATIONS

Implementing ConvNets in software is best achieved using the modular, object-oriented approach suggested in [2]. Each basic module (convolution, pooling, etc) is implemented as a class with three member functions `module.fprop(input,output)`, which computes the output from the input, `module.bprop(input,output)`, which back-propagates gradients from the outputs back to the inputs and the internal trainable parameters, and optionally `module.bbprop(input,output)`, which may back-propagate second diagonal derivatives for the implementation of second-order optimization algorithms [8].

Several software implementations of ConvNets are built around this concept, and have four basic capabilities: 1. a flexible multi-dimensional array library with basic operations such as dot products, and convolutions, 2. a class hierarchy of basic learning machine building blocs (e.g. multiple convolutions non-linear transforms, cost functions, ...), 3. a set of classes for energy-based inference [42], gradient-based optimization, and performance measurement.

Three available ConvNet implementations use this concept. The first one is part of the Lush system, a Lisp dialect with an interpreter and compiler with an easy interface to C [43] . The second one is EBlearn, a C++ machine learning library with class hierarchy to the Lush implementation [44]. Third is Torch-5 [45] a C library with an interpreter front end based on Lua. All three systems come with facilities to manipulate large datasets, images, and videos.

The first hardware implementations of ConvNets date back to the early 90's with Bell Labs' ANNA chip, a mixed analog-digital processor that could compute 64 simultaneous 8×8 convolutions at a peak rate of 4.10^9 multiply-accumulate operations per second [46], [47], with 4 bit resolution on the states and 6 bits on the weights. More recently, a group from the Canon corporation developed a prototype ConvNet chip for low-power intelligent cameras [48]. Some current approaches rely on Addressed-Event Representation (AER) convolvers, which present the advantage of not requiring multipliers to compute the convolutions. CAVIAR is the leading such project, with a performance of 12G connections/sec [49].

FPGA implementations of ConvNets appeared in the mid-90's with [50], which used low-accuracy arithmetic to avoid implementing full-fledged multipliers. Fortunately, recent DSP-oriented FPGAs include large numbers of hard-wired MAC units, which allow extremely fast and low power implementations of ConvNets. The CNP developed in our group [51] achieves 10GOPS for 7x7 kernels, with an architecture that implements entire ConvNets, including pre/post-processing, and is entirely programmable. An actual face detection application was demonstrated on this system, achieving 10fps on VGA images [52].

IV. CONCLUSION

The Convolutional Network architecture is a remarkably versatile, yet conceptually simple paradigm that can be applied to a wide spectrum of perceptual tasks. While traditional ConvNet trained with supervised learning are very effective, training them require a large number of labeled training samples. We have shown that using simple architectural tricks such as rectification and contrast normalization, and using unsupervised pre-training of each filter bank, the need for labeled samples is considerably reduced. Because of their applicability to a wide range of tasks, and because of their relatively uniform architecture, ConvNets are perfect candidates for hardware implementations, and embedded applications, as demonstrated by the increasing amount of work in this area. We expect to see many new embedded vision systems based on ConvNets in the next few years.

Despite the recent progress in deep learning, one of the major challenges of computer vision, machine learning, and AI in general in the next decade will be to devise methods that can automatically learn good features hierarchies from unlabeled and labeled data in an integrated fashion. Current and future research will focus on performing unsupervised learning on multiple stages simultaneously, on the integration of unsupervised and unsupervised learning, and on using the feed-back path implemented by the decoders to perform visual inference, such as pattern completion and disambiguation.

References

[1] 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'89*.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[3] S. Lyu and E. P. Simoncelli, "Nonlinear image representation using divisive normalization," in *CVPR*, 2008.

[4] N. Pinto, D. D. Cox, and J. J. DiCarlo, "Why is real-world visual object recognition hard?" *PLoS Comput Biol*, vol. 4, no. 1, p. e27, 01 2008.

[5] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, 1989.

[6] Y. Simard, Patrice, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *ICDAR'03*.

[7] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun, "Learning invariant features through topographic filter maps," in *CVPR'09*.

[8] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, 1998.

[9] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, no. 6, pp. 455–469, 1982.

[10] K. Chellapilla, M. Shilman, and P. Simard, "Optimally combining a cascade of classifiers," in *Proc. of Document Recognition and Retrieval 13, Electronic Imaging, 6067*, 2006.

[11] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *IWFHR'06*.

[12] A. Abdulkader, "A two-tier approach for arabic offline handwriting recognition," in *IWFHR'06*.

[13] K. Chellapilla and P. Simard, "A new radical based approach to offline handwritten east-asian character recognition," in *IWFHR'06*.

[14] R. Vaillant, C. Monrocq, and Y. LeCun, "Original approach for the localisation of objects in images," *IEE Proc on Vision, Image, and Signal Processing*, vol. 141, no. 4, pp. 245–250, August 1994.

[15] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[16] M. Osadchy, Y. LeCun, and M. Miller, "Synergistic face detection and pose estimation with energy-based models," *Journal of Machine Learning Research*, vol. 8, pp. 1197–1215, May 2007.

[17] F. Nasse, C. Thurau, and G. A. Fink, "Face detection using gpu-based convolutional neural networks," pp. 83–90, 2009.

[18] A. Frome, G. Cheung, A. Abdulkader, M. Zennaro, B. Wu, A. Bissacco, H. Adam, H. Neven, and L. Vincent, "Large-scale privacy protection in street-level imagery," in *ICCV'09*.

[19] S. Nowlan and J. Platt, "A convolutional neural network hand tracker." San Mateo, CA: Morgan Kaufmann, 1995, pp. 901–908.

[20] M. Delakis and C. Garcia, "Text detection with convolutional neural networks," in *International Conference on Computer Vision Theory and Applications (VISAPP 2008)*, 2008.

[21] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems (NIPS 2005)*. MIT Press, 2005.

[22] R. Hadsell, P. Sermanet, M. Scoffier, A. Erkan, K. Kavackuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, February 2009.

[23] M. Happold and M. Ollis, "Using learned features from 3d data for robot navigation," 2007.

[24] V. Jain and H. S. Seung, "Natural image denoising with convolutional networks," in *Advances in Neural Information Processing Systems 21 (NIPS 2008)*. MIT Press, 2008.

[25] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. Barbano, "Toward automatic phenotyping of developing embryos from videos," *IEEE Transactions on Image Processing*, 2005, special issue on Molecular and Cellular Bioimaging.

[26] V. Jain, J. F. Murray, F. Roth, S. Turaga, V. Zhigulin, K. Briggman, M. Helmstaedter, W. Denk, and H. S. Seung, "Supervised learning of image restoration with convolutional networks." in *ICCV'07*.

[27] M. Mozer, *The Perception of Multiple Objects, A Connectionist Approach*. MIT Press, 1991.

[28] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *CVPR*, 2005.

[29] J. Mutch and D. G. Lowe, "Multiclass object recognition with sparse, localized features," in *CVPR*, 2006.

[30] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks." *Science*, 2006.

[31] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *NIPS*, 2007.

[32] M. Ranzato, Y. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," in *NIPS'07*, 2007.

[33] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. International Conference on Computer Vision (ICCV'09)*. IEEE, 2009.

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

[35] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing, "Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks," in *ECCV*. Springer-Verlag, 2008.

[36] J. Weston, F. Rattle, and R. Collobert, "Deep learning via semi-supervised embedding," in *ICML*, 2008.

[37] K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "Fast inference in sparse coding algorithms with applications to object recognition," Tech. Rep., 2008, tech Report CBLL-TR-2008-12-01.

[38] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: a strategy employed by v1?" *Vision Research*, 1997.

[39] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006.

[40] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[41] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[42] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," in *Predicting Structured Data*, G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, Eds. MIT Press, 2006.

[43] Y. LeCun and L. Bottou, "Lush reference manual," Tech. Rep., 2002, code available at http://lush.sourceforge.net. [Online]. Available: http://lush.sourceforge.net

[44] P. Sermanet, K. Kavukcuoglu, and Y. LeCun, "Eblearn: Open-source energy-based learning in c++," in *Proc. International Conference on Tools with Artificial Intelligence (ICTAI'09)*. IEEE, 2009.

[45] R. Collobert, "Torch," presented at the Workshop on Machine Learning Open Source Software, NIPS, 2008.

[46] B. Boser, E. Sackinger, J. Bromley, Y. LeCun, and L. Jackel, "An analog neural network processor with programmable topology," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 2017–2025, December 1991.

[47] E. Säckinger, B. Boser, J. Bromley, Y. LeCun, and L. D. Jackel, "Application of the ANNA neural network chip to high-speed character recognition," *IEEE Transaction on Neural Networks*, 1992.

[48] O. Nomura and T. Morie, "Projection-field-type vlsi convolutional neural networks using merged/mixed analog-digital approach," in *ICONIP'07*.

[49] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camu nas-Mesa, R. Berner, M. Rivas-Pérez, T. Delbrück, S.-C. Liu, R. Douglas, P. Häfliger, G. Jiménez-Moreno, A. C. Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jiménez, and B. Linares-Barranco, "Caviar: a 45k neuron, 5m synapse, 12g connects/s aer hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking," *Trans. Neur. Netw.*, vol. 20, no. 9, pp. 1417–1438, 2009.

[50] J. Cloutier, E. Cosatto, S. Pigeon, F. Boyer, and P. Y. Simard, "Vip: An fpga-based processor for image processing and neural networks," in *MicroNeuro*, 1996.

[51] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "Cnp: An fpga-based processor for convolutional networks," in *International Conference on Field Programmable Logic and Applications*, 2009.

[52] C. Farabet, C. Poulet, and Y. LeCun, "An fpga-based stream processor for embedded real-time vision with convolutional networks," in *Fifth IEEE Workshop on Embedded Computer Vision (ECV'09)*. IEEE, 2009.