Clustering Learning for Robotic Vision

Eugenio Culurciello, Jordan Bates, Jose Carrasco, Clement Farabet, Yann LeCun

November 20, 2012

1 Introduction

In the recent years the fusion of bio-inspired and neuromorphic vision models and machine learning has dominated the development of artificial vision system for the categorization of multiple objects in static frames. Bio-inspired deep networks are computer-vision and computational-neuroscience models of the mammalian visual system implemented in deep neural networks [1–6]. Most deep network architectures are composed of multiple layers (2, 3 typically), where each layer is composed of: linear two-dimensional filtering, non-linearity, pooling of data, output data normalization [8–10]. Recent machine learning research has focused on the task of training such deep networks from the abundant digital data in form of image frames and videos. In particular, deep networks need to learn good feature representations for complex visual tasks such as object categorization and tracking of objects in space and time, identifying object presence and absence. These representation usually involve learning the linear filter weight values from labeled and unlabeled input data. Since labeled data is costly and often ridden with human errors [19–21], the recent focus is on learning these features purely from unlabeled input data [11, 15–18]. These recent methods typically learn multiple layers of deep networks by training several layers of features, one layer at a time, with varying complexity of leaning models.

Recent techniques based on unsupervised clustering algorithms are especially promising because they use simple learning methods that quickly converge [11]. These algorithms are easy to setup and train and are especially suited for robotics research, because less complex knowledge of machine learning is needed, environment-specific data can be collected quickly with a few minutes of video, setup of custom size deep networks is quick and can be adapted to specific tasks. In addition, real-time operation with efficient networks can be obtained with only a few minutes of training and setup, leading to quick and direct experimentation in robotic experiments.

In this paper we present results obtained with unsupervised clustering algorithms on the training and operation of deep neural networks for real-time robotic vision systems. We provide simple techniques and open-source software that allows robotic researchers to use deep network in a short setup time and with little or no knowledge of machine learning necessary. The main goal of the paper is not to present state-of-art results on a specific dataset. Rather we use standard published datasets to evaluate the performance of prototype robotic vision system for general-purpose use, where no dataset is available. It is thus not useful to train the network to perform only on one dataset, when the levels of performance would not carry over to another dataset or real-world images. The goal is thus mainly to evaluate the use of unsupervised networks that can support at least ten frames-per-second operation on commercial hardware, such as recent laptop computers.

The paper presents the following key innovations: (1) use of clustering learning for quickly training robotic systems (section 2), (2) the use of distance-based filtering, as opposed to the standard convolution (section 2.2), (3) the experimental proof that clustering learning networks can outperform supervised networks on general purpose robotic tracking tasks (section 4).

2 Methods

In this paper we created and tested a model of unsupervised clustering algorithms (CL) that can quickly learn linear filters weight values, and also is amenable to real-time operation with conventional mobile hardware. We used the Torch7 software for all our experiments [22], since this software can reduce training and learning of deep networks by 5-10 times compared to similar Matlab and Python tools.

2.1 Input data

Input image data was obtained from the CIFAR10 [23] and the Street View House Numbers (SVHN) [26] datasets. The SVHN dataset has a training size of 73,257 32x32 images and test size of 26,032 32x32 images. the CIFAR10 dataset has a training size of 20,000 32x32 images and a test size of 2,000 32x32 images. Both datasets offer a 10 categories classification task on 32 x 32 size images. The train dataset was in all cases learned to 100% accuracy, and training was then stopped. Input data was contrast normalized separately on each RGB channel with a 9x9 gaussian filter using the Torch7 "nn.Spatial Contrastive Normalization" function.

As testing data for a general-purposed robotic vision system, we decided to test our networks in a tracking task, where the network has to be able to track an object of interest based on a single presentation. For this purpose we use the challenging benchmark TLD dataset [28]. From this dataset we selected multiple videos with different properties of occlusions, camera movement, pose, scale and illumination changes.

Even if other rgroups showed slight improvements using the YUV color space [8], we did not use it. Rather we kept the images in their original RGB to keep the model closer to biological human vision, where the retina is broadly sensitive to roughly RGB [27].

Also we did not use whitening of data (such as ZCA whitening) even if other groups have shown clear advantages of using it. We did not use whitening because of two main reasons: first it is not applicable for general-purpose vision system where an a-priori dataset cannot be obtained. Second whitening computation is very expensive (similar to the first layer of a convolutional neural network) and we instead replaced it with local contrast normalization, which is a bio-inspired technique to whiten the input data removing its mean bias and adapting the input dynamic range.

2.2 Network architecture

The deep neural network architecture is composed of 4 layers, not counting pooling and normalization operations. Two layers of linear two-dimensional filtering and two layers of output classifier in the form of a fully connected 2-layer neural network. The first two layers were composed of a two-dimensional convolutional linear filtering stage, a L2 norm pooling stage, and a subtractive normalization layer for removing the mean of all outputs. The filters of the first two layers are generated with unsupervised clustering algorithms, as explained below. Training of the last two fully-connected neural network layers was performed with approximately 50-100 epochs on the SVHN dataset on a quad-core Intel i7 laptop, or about 8 hours. Test data maximum precision usually only needed approximately 15 epochs.

The layers in the clustering learning network used the following sequence of operations:

1. SpatialSAD module: performing sum-abs-diff operation on images convolutionally with the learned CL filters

2. Spatial Contrastive Normalization: to zero the data mean and standard deviation

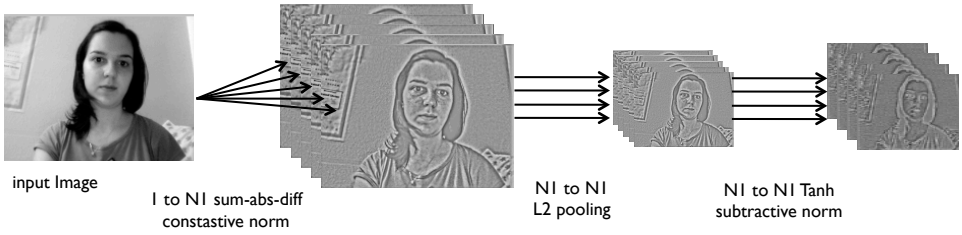


Figure 1: Architecture of one layer of the clustering learning (CL) network. Filters were applied with a sum-abs-diff operations, followed by contrastive normalization, L-2 pooling of features, nonlinearity tanh, and subtractive normalization.

3. L2 pooling over 2x2 regions
4. Tanh nonlinearity
5. Spatial Subtractive Normalization: to zero the data mean

In order to show the effectiveness of the learning techniques, we compared them to a standard 1-layer and a 2-layers convolutional neural network (CNN) [7, 9, 10], followed by the same 2-layers classifier. The layers in the convolutional neural network used the following sequence of operations:

1. SpatialConvolution module
2. L2 pooling over 2x2 regions
3. Tanh nonlinearity
4. Spatial Subtractive Normalization: to zero the data mean

All networks used 16 filters on the first layer, 128 filters on the second layer. The final classifier was fixed to 128 hidden units and 10 output classes for CIFAR and SVHN. Clustering learning networks used a fully connected input to 1st, and 1st to 2nd layer, while convolutional neural networks used 1-out-of-3 and 4-out-of-16 random connection table between the input and 1st layer, and 1st and 2nd layer respectively.

Notice that CL networks used sum-abs-diff metrics to correlate filters responses to inputs. This is different to the standard approach of deep networks [1, 8, 17, 23] where convolution operations are used. Our choice of sum-abs-diff operations was dictated by improved performance of the CL filters with respect to convolutions. In a standard modern computer the

difference between convolutions (multiplications by weights) and distance metrics (differences, subtractions) is not visible, as multipliers are optimized to perform as fast as the simpler difference operations. On the other hand, when programmable hardware as FPGA is used, the use of differences instead of multipliers can reduce the silicon area utilization, and power up to 15 times on a 16 bit operation, and more for larger number of bits. A full comparison of the hardware advantages of using distance operators instead of convolutions will be the subject of future publications.

2.3 Learning

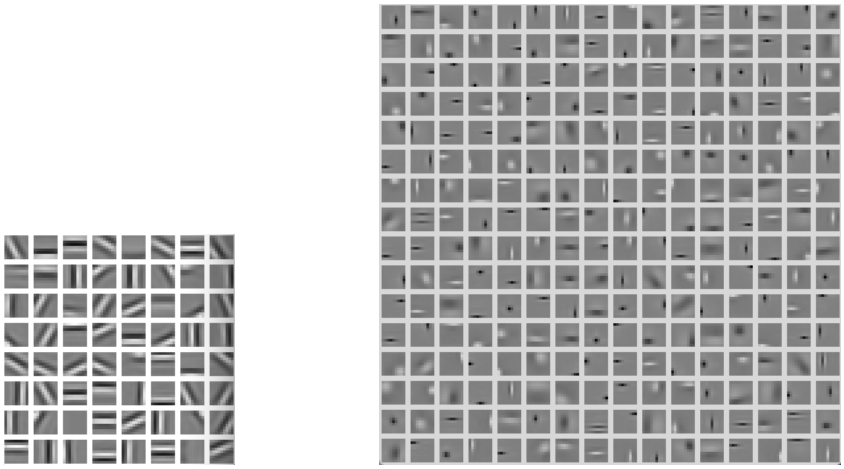
We use k-means clustering algorithm to learn a set of 16 filters in the first layer, and 128 filters in the second layer. The techniques and scripts are general and can quickly modified to learn any number of filters. The filter sizes on both layers was set to 5 x 5 pixels for the SVHN datasets. In CIFAR 1st lair we used 5 x 5 filters, and on the 2nd layer, we used 3 x 3 filters, as features were more spatially constrained in this dataset. Clustering used the same size patches of the normalized images, and we used 1 M patches from each dataset to train the first layer. The second layer training was performed by passing the entire dataset through the first layer of the deep neural network. The output dataset was then used again with the same script to train another set of linear filters, by using 1 M patches of the processed dataset.

Clustering learning filters learned on the 2nd layer used as many planes as the 1st layer (16 here). This was done to cluster different sets of features for each output of the 1st layer, and increased performance by an average of 5% or more.

Examples of the filters learned with CL techniques on a 1st and 2nd layer are given in figure 2. Both layer filters were obtained with training for 10 minutes on a modern quad-core Intel i7 laptop. A supervised network of this size would require several tens of thousands of image examples and several hours of training time.

3 Real-time network

The goal of this paper was to provide a simple and fast method to train unsupervised networks for general-purposed robotic vision system. For real-time experiments we used the TLD dataset [28]. We then compared the tracking performance of a real-time deep network both trained supervised [25], and with CL techniques described here.



(a) 1st layer filters (b) 2nd layer filters

Figure 2: Filters obtained with clustering learning on the 1st and 2nd layer. The filters obtained on the 1st layer are quite similar to elongated Gabor patches, and what can be obtained with more complex and numerically involved unsupervised techniques. Filter training with CL was obtained in 10 min time on a modern laptop.

The CL network for this task had the same two-layer network architecture used in [25]. This insured real-time operation of 6 frames/s on a quad-code Intel i7 laptop computer. We focused on this network and restricted ourselves to real-time operation because the goal of this project is the use of deep networks in mobile computers. The network operates on 46 x 46 input images, uses 16 filters with 7 x 7 receptive field on the first layer and 128 filters with 7 x 7 receptive field on the second layer. The 1st to 2nd layer fan-in was 8, and the 2nd to 3rd layer fan-in was 64. Both layers were connected with random tables. The network produces a 128 feature vector as output.

We trained the same size and number of filters through clustering algorithm for use in this network. We used patches from a contrast normalized version of a few images from the Berkeley image dataset [29]. Any set of natural-scene images can be used to train the general-purpose network presented in this section, and we on purpose chose not to sample patches from the target TLD dataset, in order to demonstrate the learning invariance properties of our technique.

4 Results

4.1 Static datasets

We report the results in the SVHN dataset in figure 3. Here we compared results of accuracy in the test set for 4 cases: clustering learning with 1 layer (CL 1 layer), clustering learning with 2 layers (CL 2 layers), a 1-layer and a 2-layers convolutional neural network (CNN 1l, 2l).

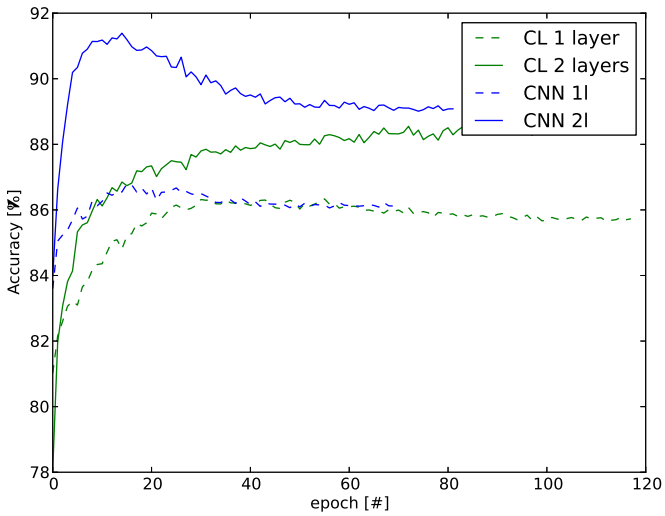


Figure 3: Test set accuracy comparison for a convolutional neural network (CNN) and a clustering learning (CL) network with 1 and 2 layers on the SVHN dataset

The data shows that clustering learning and CNN with 1 layer provide remarkably the same levels of accuracy on the dataset. This shows that at least on the 1st layer, the features learned with clustering learning are very effective. Adding a second layer brings the convnet to 91% levels of accuracy, which are standard without using any sophisticated tricks and with a small network with only 16 filters on the first layer. On the other had, the clustering learning network with 2 layers showed more than 2% increase in accuracy, from 86% to 89%. This increase is not as large as one would want and expect from adding a second layer, but is consistent with unsupervised learning results [11, 12].

It is interesting to note that with in the clustering learning 2 layers network accuracy was above 88%, and plateaued with the train set accuracy plateauing also at 92%. This shows that clustering learning filters also do not over fit, and present non-perfect, but almost identical results on both train and test sets. We note that CNN network trained supervised on SVHN report state-of-the-art performances of 96% and above.

The results above were all obtained with feed-forward hierarchical networks. We also tried to use multiple layers of clustering learning unsupervised networks in parallel, as recommended by other publications [9,11], but we did not obtain any benefits from that strategy, on the contrary parallel networks always reported losses of 3-5% accuracy with respect to a single layer. This is different from what reported in [11, 12].

We also report here the results in the CIFAR10 dataset in figure 4. As in the SVHN case, we compared results of accuracy in the test set for 4 cases: clustering learning with 1 layer (CL 1 layer), clustering learning with 2 layers (CL 2 layers), a 1-layer and a 2-layers convolutional neural network (convnet 1l, 2l).

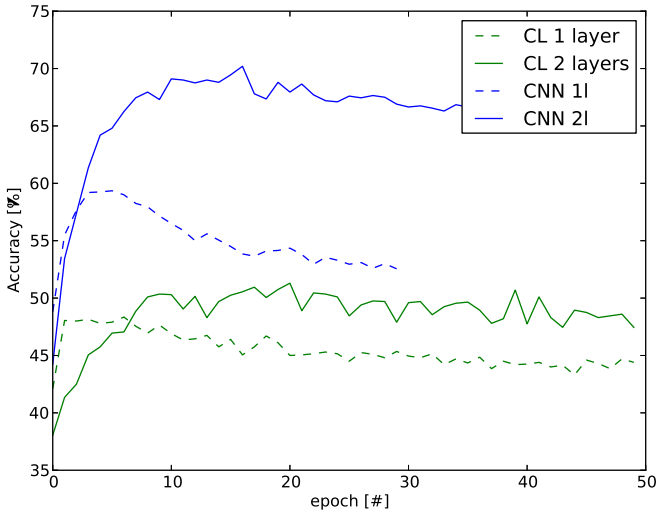


Figure 4: Test set accuracy comparison for a convolutional neural network (CNN) and a clustering learning (CL) network with 1 and 2 layers on the CIFAR10 dataset

The results in the CIFAR10 dataset show a gain of more than 10% from

using a single layer convnet to a 2 layer convnet. Clustering learning showed the same behavior as in the SVHN dataset: adding a second layer achieved 3-5% better accuracy on the test set. This dataset uses very small images and is notoriously difficult, and clustering learning only reported a 50% accuracy overall with 2 layers. All results are much lower than the current state of art in this dataset, which is close to 90% [13]. But again we stress that our goal is real-time implementation and other researcher have obtained record results in this dataset only with very large networks, most of which are not amenable to real-time operation.

We also used the filters obtained with CL on a 2-layers standard CNN network. But the use of convolution operation was giving results that were 5-10% less precise that the results obtained with sum-abs-diff operators.

4.2 Dynamic datasets

The results for the comparison for the real-time tracking network on the TLD dataset [28] are given in Table 1. Each video contains only one target. The metric used is the number of correctly tracked frames. More information and images on this dataset can be found here [30]. All network we tested were real-time networks performing at or above 5 frames/s. As can be seen in Table 1, the Clustering Learning (CL) network performs better than the convolutional neural network from [25] in all sequences, and is comparable only in one sequence ("Pedestrian 2"). These results are currently not state-of-the-art, which is currently obtained in references [30, 31].

Not only the perfomance of the CL network is higher than CNN on the TLD dataset, but also the CNN from [25] was trained in a week time, while the CL network was tried in 10 minutes. There is a clear advantage to using CL networks for general-purpose, dataset-free robotic vision tasks.

In addition we present in table 2 a comparison of running time of the most recent deep network work performing at the state-of-the-art in a variety of static datasets. We provide this comparison to show that although these networks perform extremely well on a specific dataset, we point out that similarly to the experimental results in 1, that performance might not carry over to robotic vision tasks such as tracking of previously unseen objects. We also want to point out that the satte-of-the-art networks in table 2 are far from real-time operation, and it is not clear in their publication how that can be achieved down-sizing the networks, and what performance they might attain.

The data provided in table 2 reports published and informally obtained data on the best results obtained in static datasets. Cl networks also re-

Table 1: Precision comparison between of a Clustering Learning (CL) network and a convolutional neural network (CNN) [25] used as trackers in the TLD dataset [28].

Sequence	Frames	Precision: **CL**	Precision: CNN [25]
David	761	**0.18**	0.08
Jumping	313	**0.37**	0.20
Pedestrian 1	140	**0.81**	0.69
Pedestrian 2	338	0.78	**0.79**
Pedestrian 3	184	**0.45**	0.44
Car	945	**0.67**	0.48
Carchase	9928	**0.38**	0.26

Table 2: Comparison of execution time of state-of-the-art networks as compared to the proposed CL network.

Publication	Frames/s	Precision	1st layer filters	Whitening
CL (this work)	5-10	89%, SVHN	16	none
[25]	1-2	79.5%, Stanford	16	none
[32]	?	80.8%, RGB-D	128	ZCA
[12]	?	82%, CIFAR10	1600	ZCA
[33]	?	98.5% ImageNet	96	none
[13]	?	88.8%, CIFAR10	300	none
[8]	?	99.5%, MNIST	32	none

ported 50% precision on CIFAR10, a low value compared to [12, 13]. As reference [32] precision we used the RGB only data. Most published results did not report computer time. We asked the author to provide us the data for this table, when available. We want to stress the importance of reporting computing time for deep networks, and the use of whitening, as it is the only way to compare large and small networks and try to find the optimal size for robotic vision system, or other tasks.

5 Discussion

We presented results on clustering learning algorithms for general-purpose vision system. These algorithms can be used to train multi-layer feed-forward neural networks in minutes, with fully automated scripts with non learning parameters. We show results on static dataset and dynamic tracking of objects in videos. We show results that prove that CL techniques are a viable and quick option to training deep networks. In static dataset, although they do not perform at the state-of-the-art because of their small network size, they can however be run in real-time because of the small network size. We also show that on tracking datasets the CL networks outperforms CNN networks. We believe the tracking dataset is a better approximation of robotic tasks, where locking on a target object is required for approach and manipulation.

We show that sum-abs-diff operators can be more effective than convolution operations for filtering in deep networks. We also point that custom hardware with these operators will be more efficient in power and space.

Many groups in the field of deep learning work on static dataset to demonstrate the best learning techniques. A lot of these techniques cannot be applied in robotic vision system with the current modern hardware because they cannot perform in real-time ad require too much computational load.

We argue that more research is needed in the field of applied and real-time deep networks, where shortcuts need to be taken in order to optimize performance for speedy operation. The use of custom hardware is recommended [34] but not necessary in many applications.

CL models are also very interesting to bio-inspired vision research because they provide a close connection between computational neuroscience and machine learning. In particular unsupervised clustering algorithms provide a simplistic model of Hebbian Learning methods, where neurons that respond to the same input are clustered [35–38] . Most of previous work

was mathematically complex and not efficient to implement. In this paper we thus present one of the first practical application of Hebbian-like learning applied to deep networks. Its fast and simple computation can help researcher quickly train complex neural networks.

References

[1] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[2] Raia Hadsell, Sumit Chopra and Yann LeCun: Dimensionality Reduction by Learning an Invariant Mapping, Proc. Computer Vision and Pattern Recognition Conference (CVPR'06), IEEE Press, 2006.

[3] Karol Gregor, Arthur Szlam and Yann LeCun: Structured Sparse Coding via Lateral Inhibition, Advances in Neural Information Processing Systems (NIPS 2011), 24, 2011.

[4] Riesenhuber, M. and Poggio, T. Hierarchical models of object recognition in cortex. Nature neuroscience, 2, 1999.

[5] T. Serre, A. Oliva, T. Poggio, A feedforward architecture accounts for rapid categorization Proceedings of the National Academy of Science, 2007.

[6] T. Serre, T. Poggio, A neuromorphic approach to computer vision, Communications of the ACM, 2010.

[7] Pierre Sermanet, Soumith Chintala and Yann LeCun: Convolutional Neural Networks Applied to House Numbers Digit Classification, Proceedings of International Conference on Pattern Recognition (ICPR'12), 2012.

[8] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato and Yann LeCun: What is the Best Multi-Stage Architecture for Object Recognition?, Proc. International Conference on Computer Vision (ICCV'09), IEEE, 2009.

[9] Yann LeCun, Koray Kavukvuoglu and Clment Farabet: Convolutional Networks and Applications in Vision, Proc. International Symposium on Circuits and Systems (ISCAS'10), IEEE, 2010.

[10] Y-Lan Boureau, Jean Ponce and Yann LeCun: A theoretical analysis of feature pooling in vision algorithms, Proc. International Conference on Machine learning (ICML'10), 2010.

[11] An Analysis of Single-Layer Networks in Unsupervised Feature Learning, Adam Coates, Honglak Lee, and Andrew Y. Ng. In AISTATS 14, 2011.

[12] A. Coates and A. Y. Ng. Selecting receptive fields in deep networks. In Advances in Neural Information Processing Systems, 2011.

[13] Dan Ciresan and Ueli Meier and Jürgen Schmidhuber, Multi-column Deep Neural Networks for Image Classification, CoRR, abs/1202.2745, 2012, http://arxiv.org/abs/1202.2745

[14] Eugenio Culurciello, neuFlow synthetic vision systems, http://www.neuflow.org/, 2012.

[15] Olshausen, B. A. and Field, D. J. Emergence of simple- cell receptive field properties by learning a sparse code for natural images. Nature, 381(6583):607609, 1996.

[16] Hyvarinen, A. and Oja, E. Independent component analysis: algorithms and applications. Neural net- works, 13(4-5), 2000.

[17] Hinton, G., Osindero, S., and Teh, Y. A fast learning algorithm for deep belief nets. Neural Computation, 18(7), 2006.

[18] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. Extracting and composing robust features with denoising autoencoders. In International Conference on Machine Learning, 2008.

[19] http://karpathy.ca/myblog/2011/04/27/lessons-learned-from-manually-classifying-cifar-10-with-code/

[20] Torralba, A. and Efros, A.A., Unbiased look at dataset bias, Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, Jun 2011, pages 1521 -1528.

[21] Hou, X. and Yuille, A. and Koch, H., A Meta-Theory of Boundary Detection Benchmarks, NIPS Workshop on Human Computation for Science and Computational Sustainability, 2012.

[22] R. Collobert, K. Kavukcuoglu and C. Farabet. Torch7: A Matlab-like Environment for Machine Learning. In BigLearn, NIPS Workshop, 2011.

[23] Learning Multiple Layers of Features from Tiny Images, http://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf, Alex Krizhevsky, 2009.

[24] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011.

[25] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers, in Proc. of the International Conference on Machine Learning (ICML'12), Edinburgh, Scotland, 2012.

[26] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011.

[27] B. A. Wandell, Foundations of Vision. Sunderland, MA: Sinauer, 1995.

[28] Z. Kalal, K. Mikolajczyk, and J. Matas, Tracking-Learning-Detection, Pattern Analysis and Machine Intelligence, 2011.

[29] D. Martin and C. Fowlkes and D. Tal and J. Malik, A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics, Proc. 8th Int'l Conf. Computer Vision, July 2001, volume 2, pages 416-423.

[30] `http://info.ee.surrey.ac.uk/Personal/Z.Kalal/TLD/TLD_dataset.pdf`

[31] Aysegul Dundar and Jonghoon Jin and Eugenio Culurciello, Visual Tracking with Similarity Matching Ratio, CoRR, abs/1209.2696, 2012, http://arxiv.org/abs/1209.2696.

[32] Richard Socher, Brody Huval, Bharath Bhat, Christopher D. Manning, Andrew Y. Ng, Convolutional-Recursive Deep Learning for 3D Object Classification, Neural Information Processing Systems, NIPS 2012.

[33] Krizhevsky, A., Sutskever, I. and Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks, Neural Information Processing Systems, NIPS 2012.

[34] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello and Y. LeCun, NeuFlow: A Runtime Reconfigurable Dataflow Processor for Vision, in Proc. of the Fifth IEEE Workshop on Embedded Computer Vision (ECV'11 @ CVPR'11), IEEE, Colorado Springs, 2011. Invited Paper.

[35] Terence D. Sanger, Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network, Neural Networks, Vol. 2, pp. 459-473, 1989.

[36] Fldik P., Forming sparse representations by local anti-Hebbian learning, Biol Cybern. 1990;64(2):165-70.

[37] Oja E. Neural networks, principal components and linear neural networks. Neural Networks. 1989;5:927935.

[38] Bell A. and T.J. Sejnoswki, The Independent Components of Natural Scenes are Edge Filters, Vision Res. 1997 December; 37(23): 33273338.