Learning connections between layers in deep networks

Eugenio Culurciello[*]
Purdue University
euge@purdue.edu

Aysegul Dundar
Purdue University
adundar@purdue.edu

Jonghoon Jin
Purdue University
jhjin@purdue.edu

Jordan Bates
Purdue University
jtbates@purdue.edu

Abstract

We present the clustering learning technique applied to learning the connection matrix between layers of multi-layer feedforward deep neural networks. We show that networks and connections trained with clustering learning can obtain similar levels of performance as supervised deep network trained to .... We obtain accuracy of ... on dataset ... This work can be applied to improve the efficiency of connections and combination of feateures between the layers of deep neural networks.

1 Introduction

Most scientists and engineers are fascinated by the design of an artificial vision system that can reproduce some of the human visual system capabilities in detecting, categorizing, tracking objects in view. The availability of a real-time synthetic vision system with such capabilities would find use in a large variety of applications, such as: autonomous cars, co-robots helpers, smart appliances, cellular-phones, to name a few.

The most promising approach in recent years, is the fusion of bio-inspired and neuromorphic vision models with machine learning [1–9]. This field, named Deep Learning, has provided state-of-the-art results in the categorization of multiple objects in static frames [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**].

Deep Learning networks are computer-vision and computational-neuroscience models of the mammalian visual system implemented in deep neural networks, where each network layer is composed of: linear two-dimensional filtering, non-linearity, pooling of data, output data normalization [7–9]. Deep Networks training is performed by means of the abundant image frames and videos one can find on the internet and large labeled datasets. 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 representations 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 [10–12], the recent focus is on learning these features purely from unlabeled input data [13–17]. 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 learning models. Traditional work in unsupervised deep learning has focused on learning layer feateures, and rarely on the connections between layers.

[*]More information on Eugenio Culurciello's laboratory and research can be found here: http://engineering.purdue.edu/elab/. Real time robotic vision systems: http://www.neuflow.org/

Recent techniques based on unsupervised clustering algorithms are especially promising because they use simple learning methods that quickly converge [17]. 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 unsupervised clustering algorithms applied to learning the connectivity matrix between layers and filters of deep neural networks for general-purpose vision systems.

The paper presents the following key innovations: (1) use of clustering learning for learning both filters and connections of deep networks with fully unsupervised techniques (Section **??**), (2) the ability of trained network to perform as well as supervised networks trained on still frames, (3) ability to run in real-time?.

2 Main idea and contribution

In this paper we created and tested a model of unsupervised clustering algorithms in deep neural networks. We extend the results obtained in previous work [http://arxiv.org/abs/1301.2820], aimed at learning filters in multi-layer networks, to also learn the connection matrix between layers.

The connections between layers in a deep neural network are a very important parameter. It has been shown that in many cases random filters perform only slightly worse than fully trained network layers [cite: On Random Weights and Unsupervised Feature Learning, Saxe et all] suggesting that filters (used in the convolutional filtering stage of each layer in a deep network) might not play a dominant role in determining the performance of a deep network. Rather, the connections between layers and the combination of features are mainly responsible for the recent success of supervised deep convolutional networks [cite: hinton krizevsky 2012 et al]. In these networks the connection between layers is learned from the data using global gradient-descent techniques at large scales.

In unsupervised deep networks, one cannot rely on back-propagation techniques to learn the connections between layers. This is because unsupervised networks do not used labeled data, and thus an error signal cannot be computed. In order to use unlabeled data to train large deep networks, we will need a technique that can derive how to group features from one layer into features of the next layers, and at the same time learn the filters needed by the next layer. This is the main contribution of this paper, and what makes it unique. A figure that qualitatively explains the main contribution of the paper is Fig. 1.

In the figure we show two layers of a deep network: layer N and layer $N+1$. The maps on the left of the figure, are the $N1$ output feature maps of the N-the layer in response to an input image. The maps on the right of the figure, are the $N2$ outputs of the first convolutional module of layer $N+1$. Our learning technique groups feature maps from layer N into "receptive fields" (RF). These receptive fields are the input of one or multiple maps in layer $N+1$. We form these RF by grouping a small number "fanin" of layer N features that co-occur in

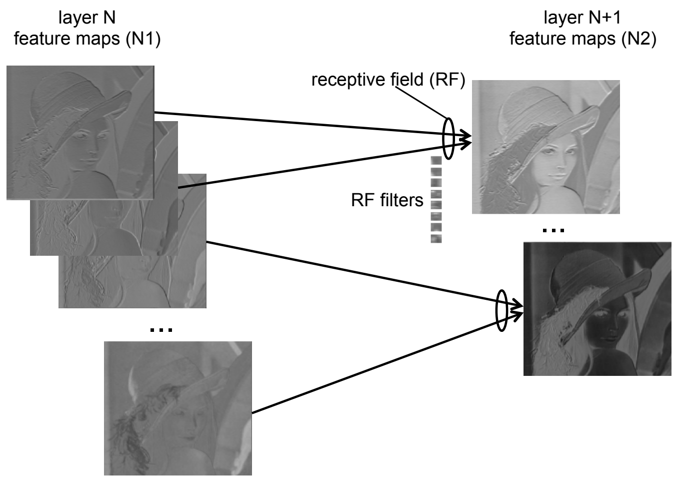


Figure 1: Learn layers

one or multiple sets of feature maps from layer N. In other words, we look for features that are highly activated in the same pixel, and we group these features into the RF. The idea behind this step is to group features that occur often together in the data. These RF are then representative of the connections between features of one layer and the next layer, and are completely obtained in a data-driven approach. In order to learn the filters for layer $N+1$ convolutional module, we perform clustering (Clustering Learning) only within the RF (and not all $N1$ maps). We then obtain both a set of connections from layer N to layer $N+1$ and the filters used by layer $N+1$ convolutional module.

The main driving force for this idea is a bio-inspired model of learning in the mammalian brain. It is widely believed that one of the possible physical mechanism of learning in the brain is the Hebbian method [**?**]. This learning method suggests that a group of pre-synaptic neurons is strongly connected to a post-synaptic neuron if the group (or receptive field, RF) can predict the response of the post-synaptic neuron. In other words neuron in the RF have strong connections to a neuron in layer $N+1$ if they can predict its response. Our idea above implements a real-valued approximation of this learning rule. The RF group features that are highly activated together in the same location. This group of highly activated features will then propagate a high value to the same location in the relative map of layer $N+1$. This implements the Hebbian rule.

We call this connection matric algorithm 'CL-connex'.

About other literature: There is not much other work in learning connections [cite: adam coates, yi-lan boureau]

Learning Feature Representations with K-means, Adam Coates and Andrew Y. Ng. In Neural Networks: Tricks of the Trade, Reloaded, Springer LNCS, 2012.

Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun: Ask the locals: multi-way local pooling for image recognition, Proc. International Conference on Computer Vision (ICCV'11), 2011

The paper [**?**] groups feateures on each layer by similarity, but does not explicitely use connection matrices between layers, rather they flatten the data at each layer and re-apply clustering algorithms. This approach is equivalent to forming a full-connection between layers, which is less efficient because each feateure is averaged with a large number of other feaetures, thus reducing the signal-to-noise ratio of important feateures. The paper [**?**] presents the interesting idea of learning pooling strategies that include feateure maps with similarity in local feateure space. This paper uses simple clustering procedures to cluster data in order to improve pooling in the feateure space. In this respect this paper present similar ideas to CL-connex, but is not apploed to deep neural networks.

3 Methods

We used the Torch7 software for all our experiments [18], since this software can reduce training and learning of deep networks by 5-10 times compared to similar Matlab and Python tools.

3.1 Input data

We tested and obtained results using the CIFAR10 [19] and the Street View House Numbers (SVHN) [20] 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 32x32 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.SpatialContrastiveNormalization" function.

ADD: INRIA Person dataset used in 3-layer networks!

Even if other groups showed slight improvements using the YUV color space [7], we did not use it because they were reporting $\sim 2-4\%$ loss in accuracy. Rather, we kept the images in their original RGB format. In our experiment we fed all RGB planes to the deep network. We also subsampled the RGB data as much as the deep network. We then concatenated teh subsampled RGB data with

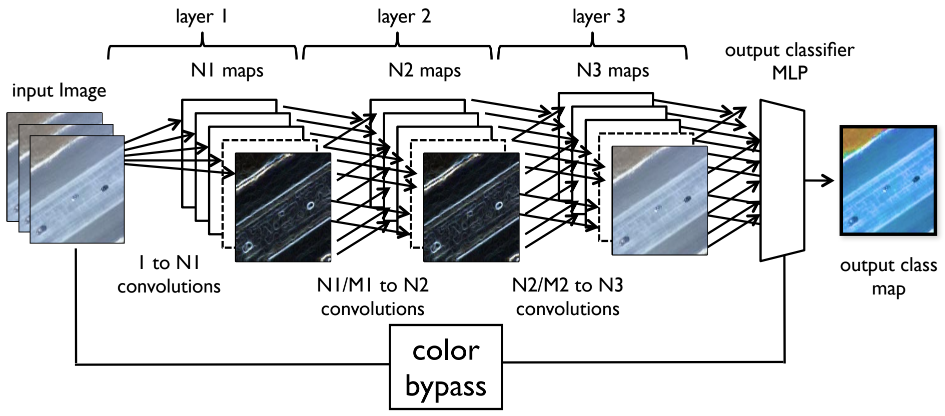


Figure 2: Architecture of the CL network used in this paper: a 2 or 3 layer convolutional network with color bypass.

the ouput of the deep network into a final vector. This final vector was then passed to a 2-layer MLP classifier.

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.

3.2 Network architecture

We experimented by training an unsupervised deep neural network with 2 layers, not counting pooling and normalization operations. The two layers were composed of a two-dimensional convolutional linear filtering stage, a spatial max pooling stage, and a subtractive and/or divisive normalization layer for removing the mean and resetting the std of all outputs to unity. The filters of the first two layers are generated with unsupervised clustering algorithms, as explained below. Using the naming convention in [8], for each layer l x_i is an input feature map, y_i is an output feature map. The input of each layer is a 3D array with n_l 2D feature maps of size $n_{l1} \cdot n_{l2}$. Each component (pixel, neuron) is denoted x_{ijk}. The output is also a 3D array, y_i composed of m_l feature maps of size $m_{l1} \cdot m_{l2}$.

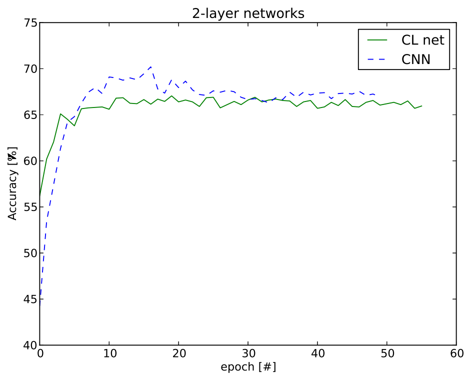
The layers in the clustering learning network used the following sequence of operations:

1. Spatial Convolution module: performing convolutions on images with the learned CL filters: $yc_i = b_j + \sum_i k_{ij} * x_i$, where $*$ is the 2D discrete convolution operator and b_j is a trainable bias parameter.
2. Spatial Max Pooling module: $yp_i = max_{n \times n}(yc_{ij})$ with $n = 2$ in this work.
3. Threshold nonlinearity: $ynl_i = 0$ if $yp_i < 0$, yp_i if $yp_i >= 0$

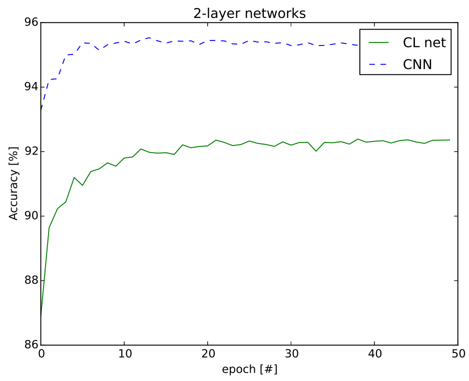
All networks used 32 filters on the first layer, 64 filters on the second layer. Clustering learning networks used the connection matrix technique CL-connex, with receptive fields of size 32 and groups of features of size 2 and 4. The CNN networks used a fully connected input to 1st, and 1st to 2nd layer.

For the INRIA person dataset, we used a 3-layer network of size 32, 64, 128 filters in each convolutional layer.

We experimented with SLAC connections [21]. In this case we used 4x more features than the numbers given above as beginning features, and then we decreased them to the same size of 32,64 in each respective layer. After the Convolution modules we added a SpatialMaxMap module to



(a) Results of CL network applied to CIFAR10.



(b) Results of CL network applied to SVHN.

Figure 3: Comparions of results between CL network and CNN.

compute the groups and reduce the feature numbers. Unfortunately SLAC did not give us any advantege because of the small number of feateures used, so we removed it from our workflow.

The final classifier was fixed to 128 hidden units and 10 output classes for CIFAR and SVHN, 2 classes for the INRIA datasets.

3.3 Learning

We use k-means clustering algorithm (we refer to this step as *Clustering Learning algorithm*) to learn a set of 32 filters in the first layer, and 64 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.

4 Results

In Fig. 3 we present the results of our test on the CIFAR10 and SVHN datasets. We compared accuracy in the test-set between our CL network and a standard convolutional neural network (CNN). We report learning rates per epochs to visualize converge speed and overfitting issues.

As one can see Fig. 3, our CL network performs very closely to the accuracy of the CNN. The CNN is always slightly better, since it uses global optimization techniques based on stochastic gradient descent (SGD) to adapt to both the signal and the noise.

Training the CNN takes $\sim$ 2 hours on our test computer with a quad core Intel i7. Training of the CL network takes $\sim$ 4 minutes. The two convolutional layers CL network are trained unsupervised using Cl on the train set for CIFAR10 and SVHN respectively. IThese layers can also be also trained with another set of natural images (a video of driving taken from YouTube), with only a slight loss of performance of $\sim 2\%$.

Notice also that the test accuracy curves per epoch of the CL network do not overfit. Rather, they saturate to a constant average value. CL network cannot overfit by design, since they perform averages on the train-set. Because of the averaging intrinsic to the algorithms of CL, the network cannot learn specific examples of the dataset and thus overfit in the test data.

Table **??** reports the execution time of some of the tested networks reported in table **??**.

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 from videos in minutes. We show results on ...

Acknowledgments

We are especially grateful to the the Torch7 developing team, in particular Ronan Collobert, who first developed this great, easy and efficient tool, Clement Farabet, Koray Kavukcuoglu, Leon Bottou. We could not have done any of this work without standing on these giants shoulders.

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] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press, 2006.

[3] K. Gregor, A. Szlam, and Y. LeCun. Structured sparse coding via lateral inhibition. In *Advances in Neural Information Processing Systems (NIPS 2011)*, volume 24, 2011.

[4] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2:10191025, 1999.

[5] T. Serre, A. Oliva, and T. Poggio. A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences*, 104(15):64246429, 2007.

[6] T. Serre and T. Poggio. A neuromorphic approach to computer vision. *Communications of the ACM*, 53(10):5461, 2010.

[7] 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.

[8] Y. LeCun, K. Kavukvuoglu, and C. Farabet. Convolutional networks and applications in vision. In *Proc. International Symposium on Circuits and Systems (ISCAS'10)*. IEEE, 2010.

[9] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in vision algorithms. In *Proc. International Conference on Machine learning (ICML'10)*, 2010.

[10] A. Karpathy. Lessons learned from manually classifying CIFAR-10. `http://karpathy.ca/myblog/2011/04/27/lessons-learned-from-manually-classifying-cifar-10-with-code/`, April 2011.

[11] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, page 15211528, 2011.

[12] X. Hou, A. Yuille, and H. Koch. A meta-theory of boundary detection benchmarks. In *NIPS Workshop on Human Computation for Science and Computational Sustainability*, 2012.

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

[14] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411430, 2000.

[15] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):15271554, 2006.

[16] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, page 10961103, 2008.

[17] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. *AISTATS*, 14, 2011.

[18] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *NIPS Workshop BigLearn*, 2011.

[19] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[20] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, 2011.

[21] Adam Coates, Andrej Karpathy, and Andrew Ng. Emergence of object-selective features in unsupervised feature learning. In *Advances in Neural Information Processing Systems 25*, pages 2690–2698, 2012.