Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Lugano, Switzerland
{jonathan,ueli,dan,juergen}@idsia.ch

Abstract. We present a novel convolutional auto-encoder (CAE) for unsupervised feature learning. A stack of CAEs forms a convolutional neural network (CNN). Each CAE is trained using conventional on-line gradient descent without additional regularization terms. A max-pooling layer is essential to learn biologically plausible features consistent with those found by previous approaches. Initializing a CNN with filters of a trained CAE stack yields superior performance on a digit (MNIST) and an object recognition (CIFAR10) benchmark.

Keywords: convolutional neural network, auto-encoder, unsupervised learning, classification.

1 Introduction

The main purpose of unsupervised learning methods is to extract generally useful features from unlabelled data, to detect and remove input redundancies, and to preserve only essential aspects of the data in robust and discriminative representations. Unsupervised methods have been routinely used in many scientific and industrial applications. In the context of neural network architectures, unsupervised layers can be stacked on top of each other to build deep hierarchies [7]. Input layer activations are fed to the first layer which feeds the next, and so on, for all layers in the hierarchy. Deep architectures can be trained in an unsupervised layer-wise fashion, and later fine-tuned by back-propagation to become classifiers [9]. Unsupervised initializations tend to avoid local minima and increase the network's performance stability [6].

Most methods are based on the *encoder-decoder* paradigm, e.g., [20]. The input is first transformed into a typically lower-dimensional space *(encoder)*, and then expanded to reproduce the initial data *(decoder)*. Once a layer is trained, its code is fed to the next, to better model highly non-linear dependencies in the input. Methods using this paradigm include stacks of: Low-Complexity Coding and Decoding machines (LOCOCODE) [10], Predictability Minimization layers [23,24], Restricted Boltzmann Machines (RBMs) [8], auto-encoders [20] and energy based models [15].

In visual object recognition, CNNs [1,3,4,14,26] often excel. Unlike patch-based methods [19] they preserve the input's neighborhood relations and

T. Honkela et al. (Eds.): ICANN 2011, Part I, LNCS 6791, pp. 52–59, 2011.

spatial locality in their latent higher-level feature representations. While the common fully connected deep architectures do not scale well to realistic-sized high-dimensional images in terms of computational complexity, CNNs do, since the number of free parameters describing their shared weights does not depend on the input dimensionality [16,18,28].

This paper introduces the *Convolutional Auto-Encoder*, a hierarchical unsupervised feature extractor that scales well to high-dimensional inputs. It learns non-trivial features using plain stochastic gradient descent, and discovers good CNNs initializations that avoid the numerous distinct local minima of highly non-convex objective functions arising in virtually all deep learning problems.

2 Preliminaries

2.1 Auto-Encoder

We recall the basic principles of auto-encoder models, e.g., [2]. An auto-encoder takes an input $\mathbf{x} \in \mathcal{R}^d$ and first maps it to the latent representation $\mathbf{h} \in \mathcal{R}^{d'}$ using a deterministic function of the type $\mathbf{h} = f_\theta = \sigma(Wx + b)$ with parameters $\theta = \{W, b\}$. This "code" is then used to reconstruct the input by a reverse mapping of f: $\mathbf{y} = f_{\theta'}(h) = \sigma(W'h + b')$ with $\theta' = \{W', b'\}$. The two parameter sets are usually constrained to be of the form $W' = W^T$, using the same weights for encoding the input and decoding the latent representation. Each training pattern x_i is then mapped onto its code h_i and its reconstruction y_i. The parameters are optimized, minimizing an appropriate cost function over the training set $\mathcal{D}_n = \{(x_0, t_0), ..., (x_n, t_n)\}$.

2.2 Denoising Auto-Encoder

Without any additional constraints, conventional auto-encoders learn the identity mapping. This problem can be circumvented by using a probabilistic RBM approach, or sparse coding, or *denoising auto-encoders* (DAs) trying to reconstruct noisy inputs [27]. The latter performs as well as or even better than RBMs [2]. Training involves the reconstruction of a clean input from a partially destroyed one. Input x becomes corrupted input $\bar{x}$ by adding a variable amount v of noise distributed according to the characteristics of the input image. Common choices include binomial noise (switching pixels on or off) for black and white images, or uncorrelated Gaussian noise for color images. The parameter v represents the percentage of permissible corruption. The auto-encoder is trained to *denoise* the inputs by first finding the latent representation $\mathbf{h} = f_\theta(\bar{x}) = \sigma(W\bar{x} + b)$ from which to reconstruct the original input $\mathbf{y} = f_{\theta'}(h) = \sigma(W'h + b')$.

2.3 Convolutional Neural Networks

CNNs are hierarchical models whose convolutional layers alternate with subsampling layers, reminiscent of simple and complex cells in the primary visual cortex [11]. The network architecture consists of three basic building blocks to be stacked and composed as needed. We have the convolutional layer, the max-pooling layer and the classification layer [14]. CNNs are among the most successful models for supervised image classification and set the state-of-the-art in many benchmarks [13,14].

3 Convolutional Auto-Encoder (CAE)

Fully connected AEs and DAEs both ignore the 2D image structure. This is not only a problem when dealing with realistically sized inputs, but also introduces redundancy in the parameters, forcing each feature to be global (i.e., to span the entire visual field). However, the trend in vision and object recognition adopted by the most successful models [17,25] is to discover localized features that repeat themselves all over the input. CAEs differs from conventional AEs as their weights are shared among all locations in the input, preserving spatial locality. The reconstruction is hence due to a linear combination of basic image patches based on the latent code.

The CAE architecture is intuitively similar to the one described in Sec. 2.2, except that the weights are shared. For a mono-channel input x the latent representation of the k-*th* feature map is given by

$$h^k = \sigma(\mathrm{x} * \mathrm{W^k} + b^k) \tag{1}$$

where the bias is broadcasted to the whole map, σ is an activation function (we used the scaled hyperbolic tangent in all our experiments), and $*$ denotes the 2D convolution. A single bias per latent map is used, as we want each filter to specialize on features of the whole input (one bias per pixel would introduce too many degrees of freedom). The reconstruction is obtained using

$$y = \sigma(\sum_{k \in H} \mathrm{h^k} * \tilde{\mathrm{W}}^{\mathrm{k}} + c) \tag{2}$$

where again there is one bias c per input channel. H identifies the group of latent feature maps; $\tilde{W}$ identifies the flip operation over both dimensions of the weights. The 2D convolution in equation (1) and (2) is determined by context. The convolution of an $m \times m$ matrix with an $n \times n$ matrix may in fact result in an $(m+n-1) \times (m+n-1)$ matrix (full convolution) or in an $(m-n+1) \times (m-n+1)$ (valid convolution). The cost function to minimize is the mean squared error (MSE):

$$E(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (x_i - y_i)^2. \tag{3}$$

Just as for standard networks the backpropagation algorithm is applied to compute the gradient of the error function with respect to the parameters. This can be easily obtained by convolution operations using the following formula:

$$\frac{\partial E(\theta)}{\partial W^k} = x * \delta h^k + \tilde{h}^k * \delta y. \tag{4}$$

δh and δy are the deltas of the hidden states and the reconstruction, respectively. The weights are then updated using stochastic gradient descent.

3.1 Max-Pooling

For hierarchical networks in general and CNNs in particular, a max-pooling layer [22] is often introduced to obtain translation-invariant representations. Max-pooling down-samples the latent representation by a constant factor, usually taking the maximum value over non overlapping sub-regions. This helps improving filter selectivity, as the activation of each neuron in the latent representation is determined by the "match" between the feature and the input field over the region of interest. Max-pooling was originally intended for fully-supervised feed-forward architectures only.

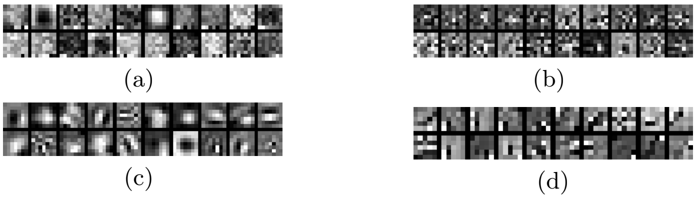
Here we introduce a max-pooling layer that introduces sparsity over the hidden representation by erasing all non-maximal values in non overlapping sub-regions. This forces the feature detectors to become more broadly applicable, avoiding trivial solutions such as having only one weight "on" (identity function). During the reconstruction phase, such a sparse latent code decreases the average number of filters contributing to the decoding of each pixel, forcing filters to be more general. Consequently, with a max-pooling layer there is no obvious need for L1 and/or L2 regularization over hidden units and/or weights.

3.2 Stacked Convolutional Auto-Encoders (CAES)

Several AEs can be stacked to form a deep hierarchy, e.g. [27]. Each layer receives its input from the latent representation of the layer below. As for deep belief networks, unsupervised pre-training can be done in greedy, layer-wise fashion. Afterwards the weights can be fine-tuned using back-propagation, or the top level activations can be used as feature vectors for SVMs or other classifiers. Analogously, a CAE stack (CAES) can be used to initialize a CNN with identical topology prior to a supervised training stage.

4 Experiments

We begin by visually inspecting the filters of various CAEs, trained in various setups on a digit dataset (MNIST [14]) and on natural images (CIFAR10 [13]). In Figure 1 we compare 20 7 × 7 filters (learned on MNIST) of four CAEs of the same topology, but trained differently. The first is trained on original digits (a), the second on noisy inputs with 50% binomial noise added (b), the third has an additional max-pooling layer of size 2 × 2 (c), and the fourth is trained on noisy inputs (30% binomial noise) and has a max-pooling layer of size 2 × 2 (d). We add 30% noise in conjunction with max-pooling layers, to avoid loss of too much relevant information. The CAE without any additional constraints (a) learns trivial solutions. Interesting and biologically plausible filters only emerge once the CAE is trained with a max-pooling layer. With additional noise the filters become more localized. For this particular example, max-pooling yields the visually nicest filters; those of the other approaches do not have a well-defined shape. A max-pooling layer is an elegant way of enforcing a sparse code required to deal with the overcomplete representations of convolutional architectures.



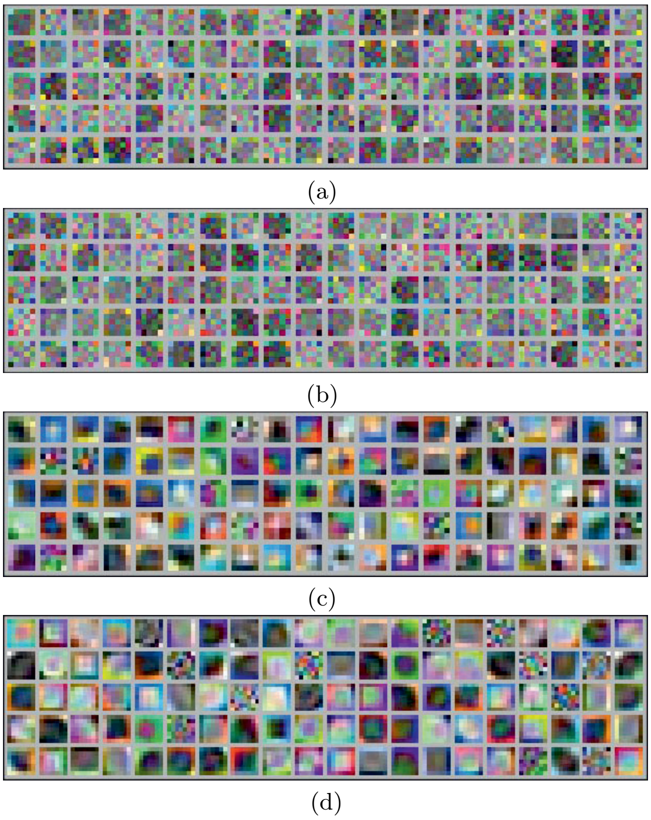
(a)

(b)

(c)

(d)

Fig. 1. A randomly selected subset of the first layer's filters learned on MNIST to compare noise and pooling. (a) No max-pooling, 0% noise, (b) No max-pooling, 50% noise, (c) Max-pooling of 2x2, (d) Max-pooling of 2x2, 30% noise.



(a)

(b)

(c)

(d)

Fig. 2. A randomly selected subset of the first layer's filters learned on CIFAR10 to compare noise and pooling (best viewed in colours). (a) No pooling and 0% noise, (b) No pooling and 50% noise, (c) Pooling of 2x2 and 0% noise, (d) Pooling of 2x2 and 50% noise.

When dealing with natural color images, Gaussian noise instead of binomial noise is added to the input of a denoising CAE. We repeat the above experiment on CIFAR10. The corresponding filters are shown in Figure 2. The impact of a max-pooling layer is striking (c), whereas adding noise (b) has almost no visual effect except on the weight magnitudes (d). As for MNIST, only a max-pooling layer guarantees convincing solutions, indicating that max-pooling is essential. It seems to at least partially solve the problems that usually arise when training auto-encoders by gradient descent. Another welcome aspect of our approach is that except for the max-pooling kernel size, no additional parameters have to be set by trial and error or time consuming cross-validation.

4.1 Initializing a CNN with Trained CAES Weights

The filters found in the previous section are not only interesting in themselves but also biologically plausible. We now train a CAES and use it to initialize a CNN with the same topology, to be fine-tuned for classification tasks. This has already shown to alleviate common problems with training deep standard MLPs, [6]. We investigate the benefits of unsupervised pre-training through comparisons with randomly initialized CNNs.

We begin with the well established MNIST benchmark [14] to show the effect of pre-training for subsets of various sizes. Classification results in Table 1 are based on the complete test set and the specified numbers of training samples. The network has 6 hidden layers: 1) convolutional layer with 100 5x5 filters per input channel; 2) max-pooling layer of 2x2; 3) convolutional layer with 150 5x5 filters per map; 4) max-pooling layer of 2x2; 5) convolutional layer of 200 maps of size 3x3; 6) a fully-connected layer of 300 hidden neurons. The output layer has a softmax activation function with one neuron per class. The learning rate is annealed during training. No deformations are applied to MNIST to increase the "virtual" number of training samples, which would reduce the impact of unsupervised pre-training for this problem that is already considered as good as solved. We also test our model on CIFAR10. This dataset is challenging because little information is conveyed by its 32 by 32 pixel input patterns. Many methods were tested on it. The most successful ones use normalization techniques to remove second order information among pixels [5,12], or deep CNNs [3]. Our method provides good recognition rates even when trained on "raw"

Table 1. Classification results on MNIST using various subsets of the full data

	1k	10k	50k
CAE [%]	**7.23**	**1.88**	**0.71**
CNN [%]	7.63	2.21	0.79
K-means (4k feat) [5][a]	-	-	0.88

[a] We performed this experiment using the code provide by the authors.

Table 2. Classification results on CIFAR10 using various subsets of the full data; comparison with other unsupervised methods

	1k	10k	50k
CAE [%]	**52.30**	**34.35**	**21.80**
CNN [%]	55.52	35.23	22.50
Mean-cov. RBM [21]	-	-	29.00
Conv. RBM [12]	-	-	21.10
K-means (4k feat) [5]	-	-	***20.40***

pixel information only. We add 5% translations only for supervised fine-tuning, and re-use the MNIST CNN architecture, except that the input layer has three maps, one for each color channel. Results are shown in Table 2. On CIFAR10 we obtain, to our knowledge, the best result so far for any unsupervised architecture trained on non-whitened data. Using raw data makes the system fully on-line and, additionally, there is no need to gather statistics over the whole training set. The performance improvement with respect to the randomly initialized CNN is bigger than for MNIST because the problem is much harder and the network profits more from unsupervised pre-training.

5 Conclusion

We introduced the Convolutional Auto-Encoder, an unsupervised method for hierarchical feature extraction. It learns biologically plausible filters. A CNN can be initialized by a CAE stack. While the CAE's overcomplete hidden representation makes learning even harder than for standard auto-encoders, good filters emerge if we use a max-pooling layer, an elegant way of enforcing sparse codes without any regularization parameters to be set by trial and error. Pre-trained CNNs tend to outperform randomly initialized nets slightly, but consistently. Our CIFAR10 result is the best for any unsupervised method trained on the raw data, and close to the best published result on this benchmark.

References

1. Behnke, S.: Hierarchical Neural Networks for Image Interpretation. LNCS, vol. 2766, pp. 1–13. Springer, Heidelberg (2003)
2. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Neural Information Processing Systems, NIPS (2007)
3. Cireşan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: High-Performance Neural Networks for Visual Object Classification. ArXiv e-prints, arXiv:1102.0183v1 (cs.AI) (Febuary 2011)
4. Ciresan, D.C., Meier, U., Masci, J., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: International Joint Conference on Artificial Intelligence, IJCAI (to appear 201I)
5. Coates, A., Lee, H., Ng, A.: An analysis of single-layer networks in unsupervised feature learning. Advances in Neural Information Processing Systems (2010)
6. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P.: Why Does Unsupervised Pre-training Help Deep Learning? Journal of Machine Learning Research 11, 625–660 (2010)
7. Fukushima, K.: Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36(4), 193–202 (1980)
8. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Comp. 14(8), 1771–1800 (2002)
9. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation (2006)

10. Hochreiter, S., Schmidhuber, J.: Feature extraction through LOCOCODE. Neural Computation 11(3), 679–714 (1999)
11. Hubel, D.H., Wiesel, T.N.: Receptive fields and functional architecture of monkey striate cortex. The Journal of Physiology 195(1), 215–243 (1968), http://jp.physoc.org/cgi/content/abstract/195/1/215
12. Krishevsky, A.: Convolutional deep belief networks on CIFAR-2010 (2010)
13. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, Computer Science Department, University of Toronto (2009)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
15. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.: A tutorial on energy-based learning. In: Bakir, G., Hofman, T., Schölkopf, B., Smola, A., Taskar, B. (eds.) Predicting Structured Data. MIT Press, Cambridge (2006)
16. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th International Conference on Machine Learning, pp. 609–616 (2009)
17. Lowe, D.: Object recognition from local scale-invariant features. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157 (1999)
18. Norouzi, M., Ranjbar, M., Mori, G.: Stacks of convolutional Restricted Boltzmann Machines for shift-invariant feature learning. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2735–2742 (June 2009), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5206577
19. Ranzato, M., Boureau, Y., LeCun, Y.: Sparse feature learning for deep belief networks. In: Advances in Neural Information Processing Systems, NIPS 2007 (2007)
20. Ranzato, M., Fu Jie Huang, Y.L.B., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: Proc. of Computer Vision and Pattern Recognition Conference (2007)
21. Ranzato, M., Hinton, G.E.: Modeling pixel means and covariances using factorized third-order boltzmann machines. In: Proc. of Computer Vision and Pattern Recognition Conference, CVPR 2010 (2010)
22. Scherer, D., Müller, A., Behnke, S.: Evaluation of pooling operations in convolutional architectures for object recognition. In: International Conference on Artificial Neural Networks (2010)
23. Schmidhuber, J.: Learning factorial codes by predictability minimization. Neural Computation 4(6), 863–879 (1992)
24. Schmidhuber, J., Eldracher, M., Foltin, B.: Semilinear predictability minimization produces well-known feature detectors. Neural Computation 8(4), 773–786 (1996)
25. Serre, T., Wolf, L., Poggio, T.: Object recognition with features inspired by visual cortex. In: Proc. of Computer Vision and Pattern Recognition Conference (2007)
26. Simard, P., Steinkraus, D., Platt, J.: Best practices for convolutional neural networks applied to visual document analysis. In: Seventh International Conference on Document Analysis and Recognition, pp. 958–963 (2003)
27. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and Composing Robust Features with Denoising Autoencoders. In: Neural Information Processing Systems, NIPS (2008)
28. Zeiler, M.D., Krishnan, D., Taylor, G.W., Fergus, R.: Deconvolutional Networks. In: Proc. Computer Vision and Pattern Recognition Conference, CVPR 2010 (2010)