Higher Order Contractive Auto-Encoder

Salah Rifai[1], Grégoire Mesnil[1,2], Pascal Vincent[1], Xavier Muller[1], Yoshua Bengio[1], Yann Dauphin[1], and Xavier Glorot[1]

[1] Dept. IRO, Université de Montréal. Montréal (QC), H2C 3J7, Canada
[2] LITIS EA 4108, Université de Rouen. 768000 Saint Etienne du Rouvray, France

Abstract. We propose a novel regularizer when training an auto-encoder for unsupervised feature extraction. We explicitly encourage the latent representation to contract the input space by regularizing the norm of the Jacobian (analytically) and the Hessian (stochastically) of the encoder's output with respect to its input, at the training points. While the penalty on the Jacobian's norm ensures robustness to tiny corruption of samples in the input space, constraining the norm of the Hessian extends this robustness when moving further away from the sample. From a manifold learning perspective, balancing this regularization with the auto-encoder's reconstruction objective yields a representation that varies most when moving along the data manifold in input space, and is most insensitive in directions orthogonal to the manifold. The second order regularization, using the Hessian, penalizes curvature, and thus favors smooth manifold. We show that our proposed technique, while remaining computationally efficient, yields representations that are significantly better suited for initializing deep architectures than previously proposed approaches, beating state-of-the-art performance on a number of datasets.

Keywords: unsupervised feature learning, deep learning, manifold

1 Introduction

Good techniques for learning a single layer of useful nonlinear feature extractors appear to be a fundamental ingredient behind most recent successes in training deeper architectures [1]. Many algorithms have already been investigated in this role, starting with the Restricted Boltzmann Machines (RBM) used to initialize ("pre-train") individual layers of Deep Belief Networks [10] and Deep Boltzmann Machines [18]. Alternatives that have been used successfully as learned nonlinear feature extractors include kernel PCA [6], semi-supervised embedding [25], sparse coding[1] [13], classical auto-encoders [2] and novel, better-suited variations of auto-encoders, such as sparse auto-encoders [14, 11, 8], and the Denoising Auto-Encoders (DAE) of [22, 23, 20].

[1] Note that in sparse coding, the forward feature mapping is not computed by a "simple" function, but is the result of an optimization procedure. It is nevertheless a deterministic mapping, albeit a computationally intensive one.

When used as deterministic feature extractors, both the Restricted Boltzmann Machines and the various flavors of auto-encoders, traditionally yield a mapping of the same basic form: extracted features are a linear projection of the input, passed through a sigmoid nonlinearity. Now all these approaches can easily be extended to other forms of nonlinear mappings, and the question of the relative merits of different types of nonlinear mappings is indeed an important one. But another equally important question that motivated the present study is what algorithm and associated *learning principle* will extract the "best" possible mapping of that traditional form. "Best" is to be understood here in the sense of producing a representation better suited to subsequent processing stages, i.e. extracting relevant, useful, features. This is typically measured objectively by the classification performance of a subsequently built classifier, starting from the representation obtained by unsupervised learning. It can also often be analyzed qualitatively by looking at the linear filters learned by the algorithm.

Modern successes of this kind of unsupervised feature learning approaches appear to depart from the past focus on dimensionality reduction. Quite the opposite, they embrace rich over-complete representations of higher dimensionality than the input. In the context of auto-encoders, no longer having a dimensionality bottleneck means one has to use some other form of regularization to preclude trivial useless solutions (where reconstruction error would be small not only for training examples but for any input configuration). Simple traditional weight decay regularization, which embodies a prior preference toward smaller magnitude weights[2] does not appear to offer an appropriate cure [23]. Some successful variants encourage sparse representations [15, 8], or in the case of the DAE [22, 23], stochastically corrupt the auto-encoder's input, thus changing the objective to that of denoising.

Recently a novel approach for regularizing auto-encoders was proposed, termed Contractive Auto-Encoders (CAE), [17, 16], showing significant improvements in state-of-the-art performance on a number of benchmark datasets. It shares a similar motivation to the DAE of [23]: aiming for robustness to small variations of the input. But the CAE achieves this in a rather different manner: instead of stochastically corrupting the input, it balances the reconstruction error with an analytical penalty term that penalizes the Frobenius norm of the encoder's Jacobian at training points. Another important difference is that the CAE aims directly at obtaining a robust *representation*, whereas the DAE's criterion is to have a robust *reconstruction* of the uncorrupted example. [17] further provide empirical evidence that the trade-off between reconstruction error and the CAE's regularization term yields a representation that captures the local directions of variation dictated by the data, which often correspond to a lower-dimensional non-linear manifold, while being more invariant to the vast majority of directions orthogonal to the manifold.

The present work extends the CAE approach by proposing a simple and computationally efficient technique to not only penalize the first order derivative (Jacobian) of the mapping but also the second order (Hessian) thus fur-

[2] Thus encourages staying closer to a linear mapping.

thering the stability of the learned representation around training points, which we find to improve the representation both in terms of filters learned (more of the visually interesting ones) and in terms of classification error. While the analytical computation of the Jacobian's norm is no more costly than computing the reconstruction error, an exact analytical computation of the Hessian would be prohibitive. We will retain computational efficiency by using a stochastic approximation of the Hessian's norm.

2 Considered framework

2.1 Setup and Notation

We are interested in learning a mapping function f that maps an input $x \in \mathbb{R}^{d_x}$ to a representation $h = f(x) \in \mathbb{R}^{d_h}$. We will be using the following notation conventions:

- Wx denotes the matrix vector product between a matrix W and vector x (vectors are considered column vectors by default).
- $\langle A, B \rangle$ denotes the inner product defined as the sum over all elements of the element-wise product. This corresponds to the ordinary dot product for vectors and to the Frobenius product for matrices.
- $\|A\| = \sqrt{\langle A, A \rangle}$, corresponds to the Euclidean norm for vectors and the Frobenius norm for matrices or tensors.
- $J_f(x) = \frac{\partial f}{\partial x}(x)$ denotes the $d_h \times d_x$ Jacobian matrix of f evaluated at x.
- $H_f(x) = \frac{\partial J}{\partial x}(x) = \frac{\partial^2 f}{\partial x^2}(x)$ denotes the $d_h \times d_x \times d_x$ Hessian tensor of f evaluated at x.
- $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ indicates ϵ is a random vector variable following an isotropic Gaussian distribution of variance σ^2.
- $\mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)}[g(\epsilon)]$ denotes the expectation of the enclosed expression with respect to the specified variable and distribution. When unambiguous, we may simply write $\mathbb{E}[g(\epsilon)]$.
- $D_n = \{x^{(1)}, \ldots, x^{(n)}\}$ is a training set of n points $x^{(i)} \in \mathbb{R}^{d_x}$ from which we want to learn mapping f.
- Function f is parameterized by a set of parameters θ. These will be learned by approximately optimizing an objective function $\mathcal{J}$, i.e. $\theta^* = \arg\min_\theta \mathcal{J}(\theta; D_n)$. This approximate optimization will be carried out with a stochastic gradient descent technique.

2.2 Basic Auto-Encoder

The basic Auto-Encoder (AE) framework considered here starts from an encoding function f that maps an input $x \in \mathbb{R}^{d_x}$ to hidden representation $h(x) \in \mathbb{R}^{d_h}$. It has the form

$$h = f(x) = s(Wx + b_h), \tag{1}$$

where s is the logistic sigmoid *activation function* $s(z) = \frac{1}{1+e^{-z}}$. The encoder is parametrized by a $d_h \times d_x$ weight matrix W, and a bias vector $b_h \in \mathbb{R}^{d_h}$.

A decoder function g then maps hidden representation h back to a reconstruction y:

$$y = g(h) = s(W'h + b_y), \tag{2}$$

The decoder's parameters are a bias vector $b_y \in \mathbb{R}^{d_x}$, and a matrix W'. In this paper we only explore the tied weights case, in which $W' = W^T$.

Basic auto-encoder training consists in finding parameters $\theta = \{W, b_h, b_y\}$ that minimize the reconstruction error on a training set of examples D_n, i.e. minimizing the following objective function:

$$\mathcal{J}_{\mathrm{AE}}(\theta) = \sum_{x \in D_n} L(x, g(f(x))), \tag{3}$$

where $L(t, r)$ is the reconstruction error between target t and reconstruction r (typically squared error or cross-entropy loss).

2.3 The First-Order Contractive Auto-Encoder

To encourage robustness of $f(x)$ to small variations of a training input x, [17] penalize its *sensitivity* to that input, measured as the Frobenius norm of the Jacobian $J_f(x)$ [16]. Thus a Contractive Auto-Encoder (CAE) is trained to optimize the following objective:

$$\mathcal{J}_{\mathrm{CAE}}(\theta) = \sum_{x \in D_n} L(x, g(f(x))) + \lambda \|J_f(x)\|^2, \tag{4}$$

where λ is a positive hyperparameter that controls the strength of the regularization.

Let $h = f(x)$. The linear+sigmoid mapping yields a simple expression for the penalty term: $\|J_f(x)\|^2 = \sum_{j=1}^{d_h} \|h_j(1 - h_j)W_j\|^2$. This has a similar computational cost as computing the reconstruction error (e.g. squared error is $\| - x + b_y + \sum_{j=1}^{d_h} h_j W_j\|^2$). Thus computing the objective and the gradient update in a CAE is only about twice as expensive as in an ordinary AE; both have the same overall computational complexity of $O(d_h d_x)$.

2.4 Proposed Higher Order Regularization

The penalty over the Jacobian yields a preference for mappings f that are invariant locally at the training points. We propose to extend the flat region further away from the training points by also penalizing higher order terms, in particular the curvature, characterized by the Hessian.

While computing the Jacobian regularization is essentially no more expensive than computing the reconstruction error, computational requirements for penalizing analytically computed higher orders grows exponentially with the order. Specifically, computing the norms of k^{th} order derivative of f has a computational complexity of $O(d_h d_x^k)$. Computing the gradient of such higher order

regularization terms with respect to model parameters thus becomes quickly prohibitive.

We propose instead to use a stochastic approximation of the Hessian Frobenius norm. Consider a noise random variable $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, we have

$$\|H_f(x)\|^2 = \lim_{\sigma \to 0} \frac{1}{\sigma^2} \mathbb{E}\left[||J_f(x) - J_f(x+\epsilon)||^2 \right] \tag{5}$$

This is obtained starting with a Taylor series expansion of J_f around x; the proof is given in the appendix. For non-infinitesimal noise, the right hand side would also contain contributions from higher order derivatives, but these vanish in the limit $\sigma \to 0$.

Our novel proposed algorithm, that we shall call Contractive Auto-Encoder with Hessian regularization (CAE+H) thus tries to optimize the following objective:

$$\mathcal{J}_{\text{CAE+H}}(\theta) = \sum_{x \in D_n} L(x, g(f(x))) + \lambda \left|\left|J_f(x)\right|\right|^2 + \gamma \mathbb{E}\left[||J_f(x) - J_f(x+\epsilon)||^2 \right], \tag{6}$$

where λ and γ are non-negative hyperparameters that control how strongly we penalize the Jacobian and the Hessian. Informally, we see that the last term limits the Hessian norm by encouraging the Jacobian norm at x and at neighboring points to be close to zero.

In practice, we will use a stochastic approximation of the expectation by generating a small mini-batch of a few corrupted samples $\tilde{x} = x + \epsilon$ (all from the same x, but different ϵ) thus incurring some variance for the computational benefit of not having to explicitly compute the analytical Hessian.

3 Geometric Interpretation

According to the manifold assumption [5], structured data in a high dimensional space, such as natural images, are thought to concentrate near a lower dimensional non-linear manifold. With this perspective in mind, [17] give the following geometric interpretation of the workings of the CAE. The penalty on the Jacobian norm encourages, at training points, an equally strong contractive mapping in all input space directions. This pressure is however exactly counteracted (at a local minimum) by the gradient of the reconstruction error term, that ensures that distinct training points can be accurately reconstructed from their representation[3]. In particular, neighboring training points on the manifold must receive a distinguishable mapping. This means that in the area surrounding the examples, the learnt mapping has to be far less contractive in directions parallel to the manifold, while it can be maximally contractive in the directions orthogonal

[3] Note that having tied weights means that encoder and decoder weights have the same magnitude. Consequently the CAE cannot merely play the game of having the encoder scale down the input and the decoder blow it up again.

to the manifold. This view means that the representation will change little when moving orthogonal to the manifold and most when moving along the manifold, so that the learnt representation constitutes a kind of coordinate system on the manifold.

[17] provide empirical evidence that this is indeed happening by examining, for the learnt mapping, the singular value spectrum of the Jacobian at training points. It characteristically showed but a few singular values much larger than the others, which confirms that the mapping is maximally contractive in the overwhelming majority of directions (presumed to be those orthogonal to the low dimensional manifold), while being significantly less contractive in only a few directions (those parallel to the low dimensional manifold). From this geometric interpretation, the leading singular vectors of the Jacobian matrix (those associated with large singular values) are the directions in input space to which the representation is most sensitive, and can be understood as *spanning the tangent space of the manifold*.

The addition of the Hessian penalty, introduced here, will encourage the Jacobian to change slowly (or not at all) as we move away from a training point. When the move is orthogonal to the the manifold, this should ensure that the directions to which the representation is most sensitive remain those parallel to the manifold. When the move is along the manifold, forcing the Jacobians at two nearby points to be close means, from the above geometrical interpretation, that the tangent spaces at these points must be close. It thus *prefers flatter manifolds by penalizing curvature*.

4 Related Previous Work

Traditional regularization [21] for learning a mapping f imposes a prior preference over the considered space of functions (or their parameters), through the addition of a penalty term $\lambda\Omega(f)$ to the regression objective. Thus the usual weight decay regularization penalizes the squared norm of the weights: $\Omega_{wd}(f) = \|W\|^2$. In the case of a linear mapping this corresponds precisely to the norm of its Jacobian matrix. But this is no longer the same with a nonlinear mapping. Penalization of second order derivatives (roughness penalty) is also a commonly employed in fitting statistical models. It is used extensively in non-parametric methods such as smoothing splines [24, 9]. These regularizers can all be expressed in the following general form, which penalizes the norm of the k^{th} order derivatives:

$$\Omega_k(f) = \int \left|\left| \frac{\partial^k f(x)}{\partial x^{(k)}} \right|\right|^2 dx,$$

where the integral is over the whole domain of f. This yields a simple tractable expression when applied to a linear mapping or to non-parametric spline models. But when learning a parameterized non-linear mapping such as linear+sigmoid, the integral becomes intractable.

Thus [4] investigated penalizing, *on training data points only*, the norm of the Hessian *diagonal only*, which he computes analytically. This is similar to our

approach, except we chose to penalize a stochastic approximation of the whole Hessian norm, rather than an analytically exact diagonal only. Our stochastic approximation scheme is a simple, practical, and computationally efficient alternative to penalizing the full Hessian norm. Also in [4] the goal was to regularize a supervised objective, which is quite different from our focus on unsupervised feature learning.

Note that applying a regularizer in this way is a departure from the classical view as a prior on function space. Since the regularization term is evaluated only on or around training points, the regularizer becomes data-dependent, and can no longer be considered a true "prior" in the Bayesian sense. Note that this is true also of the various attempts to yield sparse representations with auto-encoder variants [8, 14, 11] that were inspired by the influential work on sparse coding by [13]. Since the sparsity of representation is sought only at training points, it also constitutes a data-dependent regularizer. Whereas in previous work this restriction to the training points (instead of the whole input domain) might have been seen as an approximation, here this is a *very important feature*. **We only want the contraction and flatness effects where the data density is large.**

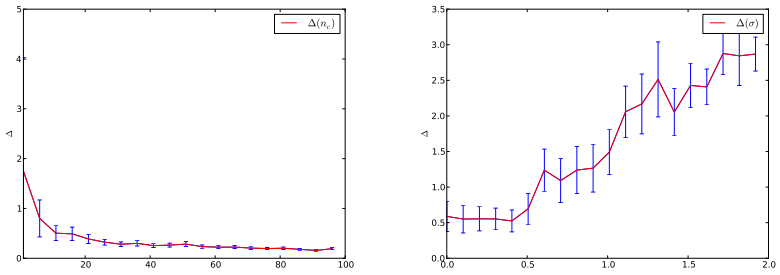
The geometrical interpretation of the Jacobian as representing the tangent space of a manifold is also related to the work on tangent propagation [19], semi-supervised embedding [25] and non-local manifold Parzen windows [3], but with marked differences. Tangent propagation [19] uses prior knowledge, based on known transformations, to define the contractive directions, and is used in a supervised setting. The CAE does not use any prior information to choose the directions of contraction. They are implicitly extracted from the dataset during the training. Semi-supervised embedding [25], in conjunction with a supervised criterion, uses pairs of neighboring points and tries to pull them together, thus explicitly contractive in directions *along* the manifold. This is to be contrasted with the CAE that contracts mostly in directions *orthogonal* to the manifold, without the computational requirement of a neighborhood graph. Non-local manifold Parzen windows [3] learns a function to explicitly output tangent directions at a point, whereas the CAE's tangent directions are to be found in the Jacobian matrix of the mapping function it learns. Contrary to the CAE+H that we propose here, none of these methods seem to address the curvature of the modeled manifold.

5 Experiments

5.1 Analysis of CAE+H

Efficiency of the approximation. Following Eq. (5), we approximate stochastically the Frobenius norm of the true Hessian of the hidden layer $h(x)$ with respect to the input. As we have seen earlier, the approximation depends on two hyperparameters, the number of corrupted inputs n_c, and the standard deviation of the Gaussian noise σ. To illustrate how the approximation is affected by different values of these hyperparameters, we estimate the absolute value of

the difference Δ between the true Hessian norm and the approximated norm as we vary both hyperparameters. As one can expect, the optimal number of corrupted inputs tends to infinity Figure 1(a), and the optimal variance tends to zero Figure 1(b). It should be noted that while evaluating the exact true Hessian norm is feasible in $O(d_x^2 d_h)$, in practice computing the gradient of the penalty w.r.t model parameters is prohibitive. With our approximation, estimating the norm costs only $O(n_c d_x d_h)$. In our experiments, due to numerical instability we avoid using tiny values for the standard deviation. This also has the beneficial effect of penalizing higher order derivatives.



(a) mini-batch size n_c (b) noise standard deviation σ

Fig. 1. Using the hidden representation $h(x)$ of an auto-encoder, we compute the meanand variance of the norm of the Hessian on 1000 samples with an optimal fixed set of hyperparameters. Evolution of Δ with respect to (a) n_c and (b) σ.

Penalization of higher order terms. In high dimensional input space, we will use a small number of corrupted samples, as it would be computationally expensive otherwise. Although the approximation is less accurate in this case, Figure 2 shows that using Eq. (5) in the objective cost is still penalizing the norm of the true Hessian in contrast with the simple CAE objective cost on the MNIST dataset. The variance of the noise is also chosen large enough so that higher order terms are also penalized. The values used in our experiments are in range $\sigma \in [0.1, 0.5]$ and $n_c \in \{4, 6, 8\}$.

5.2 Experimental results

Considered models. In our experiments, we compare the proposed Higher Order Contractive Auto-Encoder (CAE+H) against the following models for unsupervised feature extraction:

- RBM-binary : Restricted Boltzmann Machine trained with Contrastive Divergence,
- AE: Basic auto-encoder,

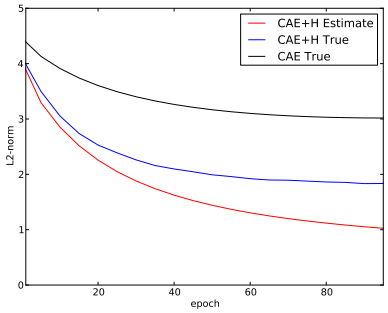


Fig. 2. On MNIST, during the optimization of CAE+H we measure the true Hessian norm and the approximated norm with $n_c = 4$ and $\sigma = 0.3$. We see that our computationnally efficient approximation is very effective at constraining the norm of the true Hessian.

– DAE: Denoising auto-encoder with Gaussian noise,
– CAE: Contractive auto-encoder.

MNIST. We tried our model on the well known digit classification problem (28×28 gray-scale pixel values scaled to [0,1]). We used the usual split of 50000 examples for training, 10000 for validation, and 10000 for test.

To empirically verify the advantage of the representation learnt using the CAE+H with respect to its discriminative power, we pretrained different auto-encoders using the regularization described above and used their hidden representation $h(x)$ as an input for a logistic regression. We also used these representations to initialize a one hidden layer MLP. The auto-encoders were also compared to an RBM.

As discussed in section 3, we illustrate how the CAE+H captures the directions of allowed variations within the data manifold. For any encoding function f, we can measure the *average contraction ratio* for pairs of points, one of which, x_0 is picked from the validation set, and the other x_1 randomly generated on a sphere of radius r centered on x_0 in input space. How this average ratio evolves as a function of r yields a *contraction curve*. We have computed these curves for the models for which we reported classification performance (the contraction curves are however computed with their initial parameters prior to fine tuning). Results are shown in Figure 6 for single-layer mappings.

5.3 MNIST variants

In addition to MNIST, we used some of its variants, namely MNIST-*rot*(digits with added random rotation) and MNIST-*bg-img*(digits with random image background) consisting of 10000 training, 2000 validation, 50000 test examples [12][4]. Finally, we considered an artificially generated dataset *rect* for shape

[4] Datasets available at `http://www.iro.umontreal.ca/~lisa/icml2007`.

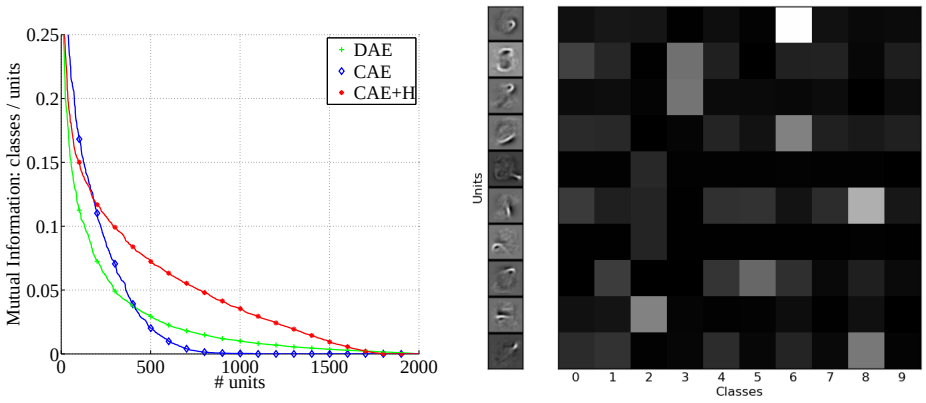


Fig. 3. On MNIST, *Left:* **Mutual information between class labels and individual hidden units** Hidden units were binarized with a threshold of 0.5. CAE+H extracts more discriminant features than others methods. *Right:* **Mutual Information between each class and random chosen hidden units.** E.g The first hidden unit is only responsive for the "6" digits

Model \ pretrain	AE	RBM	DAE	CAE	CAE+H
LogReg	2.17±0.29	2.04±0.28	2.05±0.28	1.82±0.26	**1.2**±0.21
MLP	1.78±0.26	1.3±0.22	1.18±0.21	1.14±0.21	**1.04**±0.20

Table 1. Comparison of the quality of extracted features from different models when using them as the fixed inputs to a logistic regression (top row) or to initialize a MLP that is fine-tuned (bottom row). Classification error rate is reported together with a 95% confidence interval. CAE+H is clearly less dependent on the fine-tuning step and outperforms all other considered models by the quality of its representation.

classification where the task is to discriminate between tall and wide rectangles (white on black).

5.4 CIFAR-10

In this section, we used the same preprocessing pipeline as [7] for feature extraction. First, we randomly extract 160000 patches from the first 10000 images of CIFAR-10. Each patch is locally contrast-normalized (subtract the mean and divide by its standard deviation) and ZCA whitened.

In order to qualitatively compare filters obtained with different approaches, we trained a CAE+H and other models in an unsupervised fashion with a high number n_h of hidden units on this set of preprocessed patches (see Figure 5). This figure will be discussed in the next section.

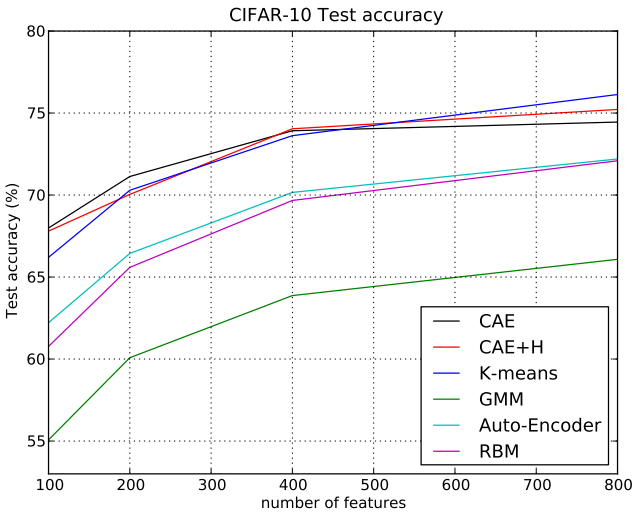


Fig. 4. Test accuracy of CAE and CAE+H versus different learning algorithms [7]

Once a non-linear feature-mapping has been learnt by unsupervised training, we can evaluate different feature extraction techniques on a set of labeled images using classification accuracy.

The same convolutional feature extraction as [7] has been used to compare CAE+H and CAE with different models namely Sparse Restricted Boltzmann Machine, Sparse Auto-encoder, Gaussian Mixtures Model and K-means clustering. On every image, each patch is preprocessed and passed through the encoder to obtain n_h feature maps. Then, to roughly reduce the dimension, features are sum-pooled together over quadrants of the feature maps. So the input dimension

Data Set	**SVM**$_{rbf}$	**SAE-3**	**RBM-3**	**DAE-b-3**	**CAE-2**	**CAE+H-1**	**CAE+H-2**
rot	11.11±0.28	10.30±0.27	10.30±0.27	**9.53**±0.26	**9.66**±0.26	10.9±0.27	**9.2**±0.25
bg-img	22.61±0.379	23.00±0.37	16.31±0.32	16.68±0.33	15.50±0.32	15.9±0.32	**14.8**±0.31
rect	2.15±0.13	2.41±0.13	2.60±0.14	1.99±0.12	1.21±0.10	**0.7**±0.07	**0.45**±0.06

Table 2. Comparison of stacked second order contractive auto-encoders with 1 and 2 layers (CAE+H-1 and CAE+H-2) with other 3-layer stacked models and baseline SVM. Test error rate on all considered classification problems is reported together with a 95% confidence interval. Best performer is in bold, as well as those for which confidence intervals overlap. Clearly CAE+Hs can be successfully used to build top-performing deep networks. 2 layers of CAE+H often outperformed 3 layers of other stacked models.

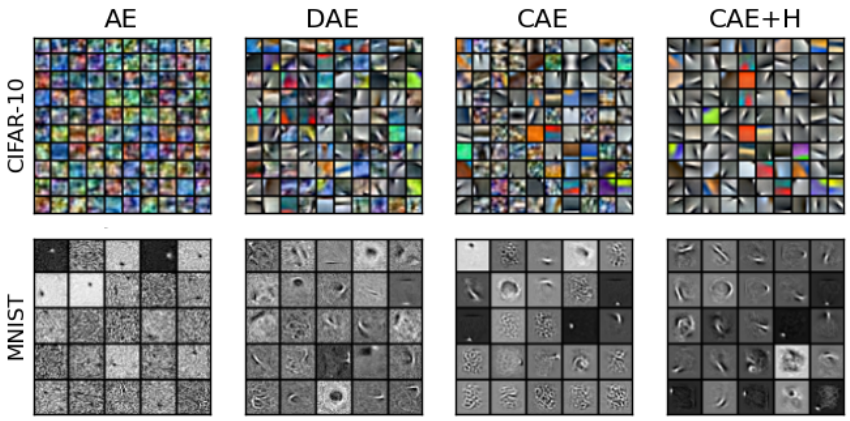


Fig. 5. Random filters from various model types with high dimensional hidden representation learnt on CIFAR10(4000*units*) and MNIST(2000*units*). CAE+H extracts smoother features and obtains a smaller proportion of noisy filters despite the exaggerated overcompleteness of the representation.

of the linear classifier is equal to $4n_h$. We trained a L2-regularized linear SVM on these features and reported the test classification accuracy in Figure 4.

As another experiment, we used the sum pooled-features learned above as the input to a shallow MLP to see if we could improve upon the SVM performance. We used a CAE to initialize the parameters of the MLP. With this method, we were able to achieve a classification accuracy of 78.5% on CIFAR-10.

6 Discussion

Upon qualitative examination of Figure 5, we can venture that the simple auto-encoder learns poor feature detectors while DAE, CAE and CAE+H appear to capture more relevant information. CAE-H and DAE present a higher proportion of structured, local and sharp filters than the CAE.

From this observation, we can hypothesize that the sharper-looking filters are likely to be more class-specific. In order to verify this objectively, we measured the mutual information between class labels and individual hidden units binarized with a threshold of 0.5). These results are reported in Figure 3.The CAE+H indeed has more specialized hidden units, that correlate better with specific class labels.

This can explain the much better classification performance obtained with a simple logistic regression stacked on the thus learnt representation, even without any supervised fine tuning of the filters, as we can see in Table 1.

We have proposed an original and computationally efficient way to regularize an auto-encoder by penalizing higher order derivatives of its learnt mapping,

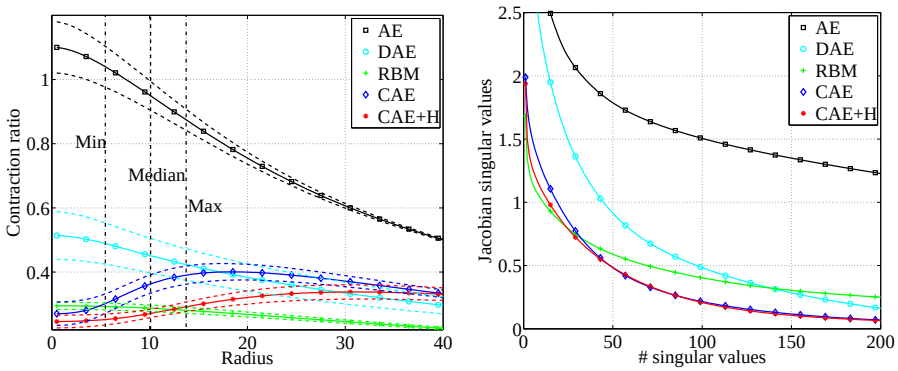


Fig. 6. On MNIST, *Left:* **Contraction ratio with respect to distance from test samples.** The contractive models have a non-monotonic contraction, CAE+H contracts further away from the samples. *Right:* **Averaged Jacobian spectrum across test samples.**

without having to explicitly compute them. This novel unsupervised method for learning feature detectors was shown to learn more appropriate features for supervised tasks than several recently proposed competing methods. In particular, it allowed us to beat the state-of-the-art on MNIST and variants, and to reach it on CIFAR-10.

Compared to the other considered approaches, the CAE+H doesn't seem to depend as much on a supervised fine-tuning step to yield good performance. This is especially interesting if one wants to apply the technique to build deep models by stacking. Indeed the fine-tuning step is subject to the vanishing gradient problem during back-propagation. So it is likely to be of great benefit to not depend so much on the fine-tuning step.

Appendix

Proof that $\lim_{\sigma \to 0} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 I)} \left[\frac{1}{\sigma^2} \left\| J(x+\epsilon) - J(x) \right\|^2 \right] = \left\| H(x) \right\|^2$

Let $\epsilon \in \mathbb{R}^{d_x}$. Taylor series expansion of the Jacobian around x yields

$$J(x+\epsilon) = J(x) + \left(\sum_{i=1}^{d_x} \epsilon_i \frac{\partial J}{\partial x_i}(x) \right) + R(x, \epsilon), \tag{7}$$

where the remainder $R(x, \epsilon)$ contains all higher order expansion terms:

$$R(x, \epsilon) = \sum_{K=2}^{\infty} \frac{1}{K!} \sum_{i_1, \ldots, i_K} \epsilon_{i_1} \ldots \epsilon_{i_K} \frac{\partial^K J}{\partial x_{i_1} \ldots \partial x_{i_K}}(x).$$

We can thus write for a given $\sigma \in \mathbb{R}$:

$$\begin{aligned}\frac{1}{\sigma^2}\left\|J(x+\epsilon)-J(x)\right\|^2 &= \frac{1}{\sigma^2}\left\|\left(\sum_{i=1}^{d_x}\epsilon_i\frac{\partial J}{\partial x_i}(x)\right)+R(x,\epsilon)\right\|^2 \\ &= \underbrace{\frac{1}{\sigma^2}\left\|\sum_{i=1}^{d_x}\epsilon_i\frac{\partial J}{\partial x_i}(x)\right\|^2}_{T_1} + \underbrace{\frac{2}{\sigma^2}\left\langle\sum_{i=1}^{d_x}\epsilon_i\frac{\partial J}{\partial x_i}(x)\,,\,R(x,\epsilon)\right\rangle}_{T_2} \\ &\quad + \underbrace{\frac{1}{\sigma^2}\left\langle R(x,\epsilon)\,,\,R(x,\epsilon)\right\rangle}_{T_3} \qquad (8)\end{aligned}$$

Let $\epsilon \sim \mathcal{N}(0,\sigma^2 I)$ and let us consider the expectation of subexpressions T_1, T_2, T_3 in the limit $\sigma \to 0$. Remainder R is a sum of terms of the form $c\,\epsilon_{i_1}\dots\epsilon_{i_K}$ where c is constant with respect to ϵ, and $K \geq 2$. Thus inner product T_2 and T_3 will be sums of products of the same form but where $K \geq 3$. Each such product is a polynomial in components of ϵ, with the smallest exponent being either 1 or ≥ 2. Such polynomials that have at least one odd exponent will have an expectation of 0 (odd moments of a 0-mean Gaussian are zero). Those that contain only even exponents will at least have one component with an exponent of 4 or more or two components with an exponent of at least 2 each, so their expectation will be a polynomial in σ whose smallest exponent is no less than 4. In all cases, for $K \geq 3$, we will have $\lim_{\sigma\to 0}\frac{1}{\sigma^2}\mathbb{E}\left[c\epsilon_{i_1}\dots\epsilon_{i_K}\right] = 0$. Since T_2 and T_3 are sums of terms of this form, we can write

$$\lim_{\sigma\to 0}\mathbb{E}\left[T_2\right] = \lim_{\sigma\to 0}\mathbb{E}\left[T_3\right] = 0. \qquad (9)$$

The expectation of the first term T_1 yields

$$\begin{aligned}\mathbb{E}\left[T_1\right] &= \mathbb{E}\left[\frac{1}{\sigma^2}\left\langle\sum_{i=1}^{d_x}\epsilon_i\frac{\partial J}{\partial x_i}(x)\,,\,\sum_{i=1}^{d_x}\epsilon_i\frac{\partial J}{\partial x_i}(x)\right\rangle\right] \\ &= \frac{1}{\sigma^2}\sum_{i=1}^{d_x}\sum_{j=1}^{d_x}\mathbb{E}\left[\epsilon_i\epsilon_j\right]\left\langle\frac{\partial J}{\partial x_i}(x)\,,\,\frac{\partial J}{\partial x_j}(x)\right\rangle.\end{aligned}$$

For $i \neq j$, $\mathbb{E}\left[\epsilon_i\epsilon_j\right] = \mathbb{E}\left[\epsilon_i\right]\mathbb{E}\left[\epsilon_j\right] = 0$ and all the corresponding terms in the above sum vanish. For $i = j$ however we have $\mathbb{E}\left[\epsilon_i\epsilon_j\right] = \mathbb{E}\left[\epsilon_i^2\right] = \sigma^2$. Consequently the above sum reduces to

$$\mathbb{E}\left[T_1\right] = \frac{1}{\sigma^2}\sum_{i=1}^{d_x}\sigma^2\left\langle\frac{\partial J}{\partial x_i}(x)\,,\,\frac{\partial J}{\partial x_i}(x)\right\rangle = \left\|H(x)\right\|^2. \qquad (10)$$

Putting together Equations 8, 9 and 10, we can conclude:

$$\begin{aligned}\lim_{\sigma\to 0}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I)}\left[\frac{1}{\sigma^2}\left\|J(x+\epsilon)-J(x)\right\|^2\right] &= \lim_{\sigma\to 0}\mathbb{E}\left[T_1\right] + \lim_{\sigma\to 0}\mathbb{E}\left[T_2\right] + \lim_{\sigma\to 0}\mathbb{E}\left[T_3\right] \\ &= \left\|H(x)\right\|^2.\end{aligned}$$

Bibliography

[1] Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009.

[2] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *NIPS 19*, pages 153–160. MIT Press, 2007.

[3] Yoshua Bengio, Hugo Larochelle, and Pascal Vincent. Non-local manifold parzen windows. In *NIPS 18*. MIT Press, 2006.

[4] Christopher M. Bishop. Curvature-driven smoothing: A learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 5(4):882–884, 1993.

[5] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[6] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In *NIPS'09*, pages 342–350. NIPS Foundation, 2010.

[7] A. Coates, H. Lee, and A. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011), JMLR W&CP*, 2011.

[8] Ian Goodfellow, Quoc Le, Andrew Saxe, and Andrew Ng. Measuring invariances in deep networks. In *NIPS'09*, pages 646–654. 2009.

[9] Trevor Hastie and Robert Tibshirani. *Generalized Additive Models*. Chapman and Hall, 1990.

[10] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[11] Koray Kavukcuoglu, Marc'Aurelio Ranzato, Rob Fergus, and Yann LeCun. Learning invariant features through topographic filter maps. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'09)*. IEEE, 2009.

[12] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In Zoubin Ghahramani, editor, *ICML 2007*, pages 473–480. ACM, 2007.

[13] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, 37:3311–3325, December 1997.

[14] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS'06*, 2007.

[15] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS'06*, pages 1137–1144. MIT Press, 2007.

[16] Salah Rifai, Xavier Muller, Gregoire Mesnil, Yoshua Bengio, and Pascal Vincent. Learning invariant features through local space contraction. Technical Report 1360, Département d'informatique et recherche opérationnelle, Université de Montréal, 2011.

[17] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contracting auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the Twenty-eight International Conference on Machine Learning (ICML'11)*, 2011.

[18] Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep Boltzmann machines. In *AISTATS'2009*, volume 5, pages 448–455, 2009.

[19] Patrice Simard, Bernard Victorri, Yann LeCun, and John Denker. Tangent prop - A formalism for specifying selected invariances in an adaptive network. In *NIPS'91*, pages 895–903, San Mateo, CA, 1992. Morgan Kaufmann.

[20] Kevin Swersky, Marc'Aurelio Ranzato, David Buchman, Benjamin Marlin, and Nando de Freitas. On score matching for energy based models: Generalizing autoencoders and simplifying deep learning. In *Proc. ICML'2011*. ACM, 2011.

[21] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. W. H. Winston, Washington D.C., 1977.

[22] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML'08*, pages 1096–1103. ACM, 2008.

[23] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11(3371–3408), 2010.

[24] G. Wahba. Spline models for observational data. In *CBMS-NSF Regional Conference Series in Applied Mathematics*, volume 59, Philadelphia, PA, 1990. Society for Industrial and Applied Mathematics (SIAM).

[25] Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *ICML 2008*, pages 1168–1175, New York, NY, USA, 2008. ACM.