# Contractive Auto-Encoders:
# Explicit Invariance During Feature Extraction

**Salah Rifai**[(1)]                                   RIFAISAL@IRO.UMONTREAL.CA
**Pascal Vincent**[(1)]                              VINCENTP@IRO.UMONTREAL.CA
**Xavier Muller**[(1)]                                 MULLERX@IRO.UMONTREAL.CA
**Xavier Glorot**[(1)]                                GLOROTXA@IRO.UMONTREAL.CA
**Yoshua Bengio**[(1)]                              BENGIOY@IRO.UMONTREAL.CA

[(1)] Dept. IRO, Université de Montréal. Montréal (QC), H3C 3J7, Canada

## Abstract

We present in this paper a novel approach for training deterministic auto-encoders. We show that by adding a well chosen penalty term to the classical reconstruction cost function, we can achieve results that equal or surpass those attained by other regularized auto-encoders as well as denoising auto-encoders on a range of datasets. This penalty term corresponds to the Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input. We show that this penalty term results in a localized space contraction which in turn yields robust features on the activation layer. Furthermore, we show how this penalty term is related to both regularized auto-encoders and denoising auto-encoders and how it can be seen as a link between deterministic and non-deterministic auto-encoders. We find empirically that this penalty helps to carve a representation that better captures the local directions of variation dictated by the data, corresponding to a lower-dimensional non-linear manifold, while being more invariant to the vast majority of directions orthogonal to the manifold. Finally, we show that by using the learned features to initialize a MLP, we achieve state of the art classification error on a range of datasets, surpassing other methods of pre-training.

## 1. Introduction

A recent topic of interest[1] in the machine learning community is the development of algorithms for unsupervised learning of a useful representation. This automatic discovery and extraction of features is often used in building a deep hierarchy of features, within the contexts of supervised, semi-supervised, or unsupervised modeling. See Bengio (2009) for a recent review of Deep Learning algorithms. Most of these methods exploit as basic building block algorithms for learning one level of feature extraction: the representation learned at one level is used as input for learning the next level, etc. The objective is that these representations become *better* as depth is increased, but *what defines a good representation?* It is fairly well understood what PCA or ICA do, but much remains to be done to understand the characteristics and theoretical advantages of the representations learned by a Restricted Boltzmann Machine (Hinton *et al.*, 2006), an auto-encoder (Bengio *et al.*, 2007), sparse coding (Olshausen and Field, 1997; Kavukcuoglu *et al.*, 2009), or semi-supervised embedding (Weston *et al.*, 2008). All of these produce a non-linear representation which, unlike that of PCA or ICA, can be stacked (composed) to yield deeper levels of representation. It has also been observed empirically (Lee *et al.*, 2009) that the deeper levels often capture more abstract features. A simple approach (Hinton *et al.*, 2006; Bengio *et al.*, 2007), also used here, to empirically verify that the learned representations are useful, is to use them to initialize a classifier (such as a multi-layer neural network), and measure classification error. Many experiments show that deeper models can thus yield lower classification error (Bengio *et al.*, 2007; Jarrett *et al.*, 2009; Vincent

---

[1]see NIPS'2010 Workshop on Deep Learning and Unsupervised Feature Learning

*et al.*, 2010).

**Contribution.** What principles should guide the learning of such intermediate representations? They should capture as much as possible of the information in each given example, when that example is likely under the underlying generating distribution. That is what auto-encoders (Vincent *et al.*, 2010) and sparse coding aim to achieve when minimizing reconstruction error. We would also like these representations to be useful in characterizing the input distribution, and that is what is achieved by directly optimizing a generative model's likelihood (such as RBMs). In this paper, we introduce a penalty term that could be added in either of the above contexts, which encourages the intermediate representation to be robust to small changes of the input *around the training examples*. We show through comparative experiments on many benchmark datasets that this characteristic is useful to learn representations that help training better classifiers. We hypothesize that whereas the proposed penalty term encourages the learned features to be locally invariant without any preference for particular directions, when it is combined with a reconstruction error or likelihood criterion we obtain invariance in the directions that make sense in the context of the given training data, i.e., the variations that are present in the data should also be captured in the learned representation, but the other directions may be *contracted* in the learned representation.

## 2. How to extract robust features

To encourage robustness of the representation $f(x)$ obtained for a training input $x$ we propose to penalize its *sensitivity* to that input, measured as the Frobenius norm of the Jacobian $J_f(x)$ of the non-linear mapping. Formally, if input $x \in \mathbb{R}^{d_x}$ is mapped by encoding function $f$ to hidden representation $h \in \mathbb{R}^{d_h}$, this sensitivity penalization term is the sum of squares of all partial derivatives of the extracted features with respect to input dimensions:

$$\|J_f(x)\|_F^2 = \sum_{ij} \left( \frac{\partial h_j(x)}{\partial x_i} \right)^2 . \qquad (1)$$

Penalizing $\|J_f\|_F^2$ encourages the mapping to the feature space to be *contractive* in the neighborhood of the training data. This geometric perspective, which gives its name to our algorithm, will be further elaborated on, in section 5.3, based on experimental evidence. The *flatness* induced by having low valued first derivatives will imply an *invariance* or *robustness* of the representation for small variations of the input. Thus in this study, terms like invariance, (in-)sensitivity, ro-

bustness, flatness and contraction all point to the same notion.

While such a Jacobian term alone would encourage mapping to a useless constant representation, it is counterbalanced in auto-encoder training[2] by the need for the learnt representation to allow a good reconstruction of the input[3].

## 3. Auto-encoders variants

In its simplest form, an auto-encoder (AE) is composed of two parts, an encoder and a decoder. It was introduced in the late 80's (Rumelhart *et al.*, 1986; Baldi and Hornik, 1989) as a technique for dimensionality reduction, where the output of the encoder represents the reduced representation and where the decoder is tuned to reconstruct the initial input from the encoder's representation through the minimization of a cost function. More specifically when the encoding activation functions are linear and the number of hidden units is inferior to the input dimension (hence forming a bottleneck), it has been shown that the learnt parameters of the encoder are a subspace of the principal components of the input space (Baldi and Hornik, 1989). With the use of non-linear activation functions an AE can however be expected to learn more useful feature-detectors than what can be obtained with a simple PCA (Japkowicz *et al.*, 2000). Moreover, contrary to their classical use as dimensionality-reduction techniques, in their modern instantiation auto-encoders are often employed in a so-called *over-complete* setting to extract a number of features larger than the input dimension, yielding a rich higher-dimensional representation. In this setup, using some form of regularization becomes essential to avoid uninteresting solutions where the auto-encoder could perfectly reconstruct the input without needing to extract any useful feature. This section formally defines the auto-encoder variants considered in this study.

**Basic auto-encoder (AE).** The encoder is a 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

$$h = f(x) = s_f(Wx + b_h), \qquad (2)$$

where $s_f$ is a nonlinear *activation function*, typically a logistic sigmoid$(z) = \frac{1}{1+e^{-z}}$. The encoder is parametrized by a $d_h \times d_x$ weight matrix $W$, and a

---

[2]Using also the now common additional constraint of encoder and decoder sharing the same (transposed) weights, which precludes a mere global contracting scaling in the encoder and expansion in the decoder.

[3]A likelihood-related criterion would also similarly prevent a collapse of the representation.

bias vector $b_h \in \mathbb{R}^{d_h}$.

The decoder function $g$ maps hidden representation $h$ back to a reconstruction $y$:

$$y = g(h) = s_g(W'h + b_y), \qquad (3)$$

where $s_g$ is the decoder's activation function, typically either the identity (yielding linear reconstruction) or a sigmoid. The decoder's parameters are a bias vector $b_y \in \mathbb{R}^{d_x}$, and matrix $W'$. In this paper we only explore the tied weights case, in which $W' = W^T$.

Auto-encoder training consists in finding parameters $\theta = \{W, b_h, b_y\}$ that minimize the reconstruction error on a training set of examples $D_n$, which corresponds to minimizing the following objective function:

$$\mathcal{J}_{\text{AE}}(\theta) = \sum_{x \in D_n} L(x, g(f(x))), \qquad (4)$$

where $L$ is the reconstruction error. Typical choices include the squared error $L(x, y) = \|x - y\|^2$ used in cases of linear reconstruction and the cross-entropy loss when $s_g$ is the sigmoid (and inputs are in $[0, 1]$): $L(x, y) = -\sum_{i=1}^{d_x} x_i \log(y_i) + (1 - x_i)\log(1 - y_i)$.

**Regularized auto-encoders (AE+wd).** The simplest form of regularization is weight-decay which favors small weights by optimizing instead the following regularized objective:

$$\mathcal{J}_{\text{AE+wd}}(\theta) = \left( \sum_{x \in D_n} L(x, g(f(x))) \right) + \lambda \sum_{ij} W_{ij}^2, \quad (5)$$

where the $\lambda$ hyper-parameter controls the strength of the regularization.

Note that rather than having a prior on what the weights should be, it is possible to have a prior on what the hidden unit activations should be. From this viewpoint, several techniques have been developed to encourage sparsity of the representation (Kavukcuoglu *et al.*, 2009; Lee *et al.*, 2008).

**Denoising Auto-encoders (DAE).** A successful alternative form of regularization is obtained through the technique of denoising auto-encoders (DAE) put forward by Vincent *et al.* (2010), where one simply corrupts input $x$ before sending it through the auto-encoder, that is trained to reconstruct the clean version (i.e. to denoise). This yields the following objective function:

$$\mathcal{J}_{\text{DAE}}(\theta) = \sum_{x \in D_n} \mathbb{E}_{\tilde{x} \sim q(\tilde{x}|x)}[L(x, g(f(\tilde{x})))], \qquad (6)$$

where the expectation is over corrupted versions $\tilde{x}$ of examples $x$ obtained from a corruption process $q(\tilde{x}|x)$. This objective is optimized by stochastic gradient descent (sampling corrupted examples).

Typically, we consider corruptions such as additive isotropic Gaussian noise: $\tilde{x} = x + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2 I)$ and a binary masking noise, where a fraction $\nu$ of input components (randomly chosen) have their value set to 0. The degree of the corruption ($\sigma$ or $\nu$) controls the degree of regularization.

## 4. Contractive auto-encoders (CAE)

From the motivation of robustness to small perturbations around the training points, as discussed in section 2, we propose an alternative regularization that favors mappings that are more strongly contracting at the training samples (see section 5.3 for a longer discussion). The Contractive auto-encoder (CAE) is obtained with the regularization term of eq. 1 yielding objective function

$$\mathcal{J}_{\text{CAE}}(\theta) = \sum_{x \in D_n} \left( L(x, g(f(x))) + \lambda \|J_f(x)\|_F^2 \right) \quad (7)$$

**Relationship with weight decay.** It is easy to see that the squared Frobenius norm of the Jacobian corresponds to a L2 weight decay in the case of a *linear encoder* (i.e. when $s_f$ is the identity function). In this special case $\mathcal{J}_{\text{CAE}}$ and $\mathcal{J}_{\text{AE+wd}}$ are identical. Note that in the linear case, keeping weights small is the only way to have a contraction. But with a sigmoid non-linearity, contraction and robustness can also be achieved by driving the hidden units to their saturated regime.

**Relationship with sparse auto-encoders.** Auto-encoder variants that encourage sparse representations aim at having, for each example, a majority of the components of the representation close to zero. For these features to be close to zero, they must have been computed in the left saturated part of the sigmoid non-linearity, which is almost flat, with a tiny first derivative. This yields a corresponding small entry in the Jacobian $J_f(x)$. Thus, sparse auto-encoders that output many close-to-zero features, are likely to correspond to a highly contractive mapping, even though contraction or robustness are not *explicitly* encouraged through their learning criterion.

**Relationship with denoising auto-encoders.** Robustness to input perturbations was also one of the motivation of the denoising auto-encoder, as stated in

Vincent *et al.* (2010). The CAE and the DAE differ however in the following ways:

CAEs explicitly encourage robustness of representation $f(x)$, whereas DAEs encourages robustness of reconstruction $(g \circ f)(x)$ (which may only partially and indirectly encourage robustness of the representation, as the invariance requirement is shared between the two parts of the auto-encoder). We believe that this property make CAEs a better choice than DAEs to learn useful feature extractors. Since we will use only the encoder part for classification, robustness of the extracted features appears more important than robustness of the reconstruction.

DAEs' robustness is obtained *stochastically* (eq. 6) by having several explicitly corrupted versions of a training point aim for an identical reconstruction. By contrast, CAEs' robustness to tiny perturbations is obtained *analytically* by penalizing the magnitude of first derivatives $\|J_f(x)\|_F^2$ at training points only (eq. 7).

Note that an *analytic approximation* for DAE's stochastic robustness criterion can be obtained in the limit of very small additive Gaussian noise, by following Bishop (1995). This yields, not surprisingly, a term in $\|J_{g \circ f}(x)\|_F^2$ (Jacobian of reconstruction) rather than the $\|J_f(x)\|_F^2$ (Jacobian of representation) of CAEs.

### Computational considerations

In the case of a sigmoid nonlinearity, the penalty on the Jacobian norm has the following simple expression:

$$\|J_f(x)\|_F^2 = \sum_{i=1}^{d_h} (h_i (1 - h_i))^2 \sum_{j=1}^{d_x} W_{ij}^2.$$

Computing this penalty (or its gradient), is similar to and has about the same cost as computing the reconstruction error (or, respectively, its gradient). The overall computational complexity is $O(d_x \times d_h)$.

## 5. Experiments and results

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

- RBM-binary : Restricted Boltzmann Machine trained by Contrastive Divergence,
- AE: Basic auto-encoder,
- AE+wd: Auto-encoder with weight-decay regularization,
- DAE-g: Denoising auto-encoder with Gaussian noise,
- DAE-b: Denoising auto-encoder with binary

masking noise,

All auto-encoder variants used tied weights, a sigmoid activation function for both encoder and decoder, and a cross-entropy reconstruction error (see Section 3). They were trained by optimizing their (regularized) objective function on the training set by stochastic gradient descent. As for RBMs, they were trained by Contrastive Divergence.

These algorithms were applied on the training set without using the labels (unsupervised) to extract a first layer of features. Optionally the procedure was repeated to stack additional feature-extraction layers on top of the first one. Once thus trained, the learnt parameter values of the resulting feature-extractors (weight and bias of the encoder) were used as initialisation of a multilayer perceptron (MLP) with an extra random-initialised output layer. The whole network was then *fine-tuned* by a gradient descent on a supervised objective appropriate for classification [4], using the labels in the training set.

**Datasets used.** We have tested our approach on a benchmark of image classification problems, namely: *CIFAR-bw*: a gray-scale version of the CIFAR-10 image-classification task(Krizhevsky and Hinton, 2009) and *MNIST*: the well-known digit classification problem. We also used problems from the same benchmark as Vincent *et al.* (2010) which includes five harder digit recognition problems derived by adding extra factors of variation to MNIST digits, each with 10000 examples for training, 2000 for validation, 50000 for test as well as two artificial shape classification problems[5].

### 5.1. Classification performance

Our first series of experiments focuses on the MNIST and CIFAR-bw datasets. We compare the classification performance obtained by a neural network with one hidden layer of 1000 units, initialized with each of the unsupervised algorithms under consideration. For each case, we selected the value of hyperparameters (such as the strength of regularization) that yielded, after supervised fine-tuning, the best classification per-

---

[4]We used sigmoid+cross-entropy for binary classification, and log of softmax for multi-class problems

[5]These datasets are available at http://www.iro.umontreal.ca/~lisa/icml2007: *basic*: smaller subset of MNIST; *rot*: digits with added random rotation; *bg-rand*: digits with random noise background; *bg-img*: digits with random image background; *bg-img-rot*: digits with rotation and image background; *rect*: discriminate between tall and wide rectangles (white on black); *rect-img*: discriminate between tall and wide rectangular image on a different background image.

*Table 1.* Performance comparison of the considered models on MNIST (top half) and CIFAR-bw (bottom half). Results are sorted in ascending order of classification error on the test set. Best performer and models whose difference with the best performer was not statistically significant are in bold. Notice how the average Jacobian norm (before fine-tuning) appears correlated with the final test error. SAT is the average fraction of saturated units per example. Not surprisingly, the CAE yields a higher proportion of saturated units.

| | Model | Test error | Average $\|J_f(x)\|_F$ | SAT |
|---|---|---|---|---|
| **MNIST** | CAE | **1.14** | $0.73\ _{10^{-4}}$ | 86.36% |
| | DAE-g | **1.18** | $0.86\ _{10^{-4}}$ | 17.77% |
| | RBM-binary | 1.30 | $2.50\ _{10^{-4}}$ | 78.59% |
| | DAE-b | 1.57 | $7.87\ _{10^{-4}}$ | 68.19% |
| | AE+wd | 1.68 | $5.00\ _{10^{-4}}$ | 12.97% |
| | AE | 1.78 | $17.5\ _{10^{-4}}$ | 49.90% |
| **CIFAR-bw** | CAE | **47.86** | $2.40\ _{10^{-5}}$ | 85,65% |
| | DAE-b | **49.03** | $4.85\ _{10^{-5}}$ | 80,66% |
| | DAE-g | 54.81 | $4.94\ _{10^{-5}}$ | 19,90% |
| | AE+wd | 55.03 | $34.9\ _{10^{-5}}$ | 23,04% |
| | AE | 55.47 | $44.9\ _{10^{-5}}$ | 22,57% |

formance on the validation set. Final classification error rate was then computed on the test set. With the parameters obtained after unsupervised pre-training (before fine-tuning), we also computed in each case the average value of the encoder's contraction $\|J_f(x)\|_F$ on the validation set, as well as a measure of the average fraction of saturated units per example[6]. These results are reported in Table 1. We see that the local contraction measure (the average $\|J_f\|_F$) on the pre-trained model **strongly correlates** with the final classification error. The CAE, which explicitly tries to minimize this measure while maintaining a good reconstruction, is the best-performing model. datasets.

Results given in Table 2 compares the performance of stacked CAEs on the benchmark problems of Larochelle *et al.* (2007) to the three-layer models reported in Vincent *et al.* (2010). Stacking a second layer CAE on top of a first layer appears to significantly improves performance, thus demonstrating their usefulness for building deep networks. Moreover on the majority of datasets, 2-layer CAE beat the state-of-the-art 3-layer model.

---

[6]We consider a unit saturated if its activation is below 0.05 or above 0.95. Note that in general the set of saturated units is expected to vary with each example.

## 5.2. Closer examination of the contraction

To better understand the feature extractor produced by each algorithm, in terms of their contractive properties, we used the following analytical tools:

**What happens locally: looking at the singular values of the Jacobian.** A high dimensional Jacobian contains directional information: the amount of contraction is generally not the same in all directions. This can be examined by performing a singular value decomposition of $J_f$. We computed the average singular value spectrum of the Jacobian over the validation set for the above models. Results are shown in Figure 2 and will be discussed in section 5.3.

**What happens further away: contraction curves.** The Frobenius norm of the Jacobian at some point $x$ measures the contraction of the mapping *locally* at that point. Intuitively the contraction induced by the proposed penalty term can be measured beyond the immediate training examples, by the ratio of the distances between two points in their original (input) space and their distance once mapped in the feature space. We call this isotropic measure *contraction ratio*. In the limit where the variation in the input space is infinitesimal, this corresponds to the derivative (i.e. Jacobian) of the representation map.

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 1 for single-layer mappings and in Figure 3 for 2 and 3 layer mappings. They will be discussed in detail in the next section.

## 5.3. Discussion: Local Space Contraction

From a geometrical point of view, the robustness of the features can be seen as a contraction of the input space when projected in the feature space, in particular *in the neighborhood of the examples from the data-generating distribution*: otherwise (if the contraction was the same at all distances) it would not be useful, because it would just be a global scaling. Such a contraction is happening with the proposed penalty, but much less without it, as illustrated on the contraction curves of Figure 1. For all algorithms tested except the proposed CAE and the Gaussian corruption DAE

*Table 2.* Comparison of stacked contractive auto-encoders with 1 and 2 layers (CAE-1 and CAE-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 CAEs can be successfully used to build top-performing deep networks. 2 layers of CAE often outperformed 3 layers of other stacked models.

| Data Set | $SVM_{rbf}$ | SAE-3 | RBM-3 | DAE-b-3 | CAE-1 | CAE-2 |
|---|---|---|---|---|---|---|
| *basic* | $3.03_{\pm 0.15}$ | $3.46_{\pm 0.16}$ | $3.11_{\pm 0.15}$ | $2.84_{\pm 0.15}$ | $2.83_{\pm 0.15}$ | $\mathbf{2.48}_{\pm 0.14}$ |
| *rot* | $11.11_{\pm 0.28}$ | $10.30_{\pm 0.27}$ | $10.30_{\pm 0.27}$ | $\mathbf{9.53}_{\pm 0.26}$ | $11.59_{\pm 0.28}$ | $\mathbf{9.66}_{\pm 0.26}$ |
| *bg-rand* | $14.58_{\pm 0.31}$ | $11.28_{\pm 0.28}$ | $\mathbf{6.73}_{\pm 0.22}$ | $10.30_{\pm 0.27}$ | $13.57_{\pm 0.30}$ | $10.90_{\pm 0.27}$ |
| *bg-img* | $22.61_{\pm 0.379}$ | $23.00_{\pm 0.37}$ | $16.31_{\pm 0.32}$ | $16.68_{\pm 0.33}$ | $16.70_{\pm 0.33}$ | $\mathbf{15.50}_{\pm 0.32}$ |
| *bg-img-rot* | $55.18_{\pm 0.44}$ | $51.93_{\pm 0.44}$ | $47.39_{\pm 0.44}$ | $\mathbf{43.76}_{\pm 0.43}$ | $48.10_{\pm 0.44}$ | $45.23_{\pm 0.44}$ |
| *rect* | $2.15_{\pm 0.13}$ | $2.41_{\pm 0.13}$ | $2.60_{\pm 0.14}$ | $1.99_{\pm 0.12}$ | $1.48_{\pm 0.10}$ | $\mathbf{1.21}_{\pm 0.10}$ |
| *rect-img* | $24.04_{\pm 0.37}$ | $24.05_{\pm 0.37}$ | $22.50_{\pm 0.37}$ | $\mathbf{21.59}_{\pm 0.36}$ | $\mathbf{21.86}_{\pm 0.36}$ | $\mathbf{21.54}_{\pm 0.36}$ |

(DAE-g), the contraction ratio *decreases* (i.e., towards more contraction) as we move away from the training examples (this is due to more saturation, and was expected), whereas for the CAE the contraction ratio *initially increases*, up to the point where the effect of saturation takes over (the bump occurs at about the maximum distance between two training examples).

Think about the case where the training examples congregate near a low-dimensional manifold. The variations present in the data (e.g. translation and rotations of objects in images) correspond to local dimensions along the manifold, while the variations that are small or rare in the data correspond to the directions orthogonal to the manifold (at a particular point near the manifold, corresponding to a particular example). The proposed criterion is trying to make the features invariant in all directions around the training examples, but the reconstruction error (or likelihood) is making sure that that the representation is faithful, i.e., can be used to reconstruct the input example. Hence the directions that *resist* to this contracting pressure (strong invariance to input changes) are the directions present in the training set. Indeed, *if the variations along these directions present in the training set were not preserved, neighboring training examples could not be distinguished and properly reconstructed.* Hence the directions where the contraction is strong (small ratio, small singular values of the Jacobian matrix) are also the directions where the model believes that the input density drops quickly, whereas the directions where the contraction is weak (closer to 1, larger contraction ratio, larger singular values of the Jacobian matrix) correspond to the directions where the model believes that the input density is flat (and large, since we are near a training example).

We believe that this contraction penalty thus helps the learner carve a kind of mountain supported by the training examples, and generalizing to a ridge between them. What we would like is for these ridges to correspond to some directions of variation present in the data, associated with underlying factors of variation. How far do these ridges extend around each training example and how flat are they? This can be visualized comparatively with the analysis of Figure 1, with the contraction ratio for different distances from the training examples.

Note that different features (elements of the representation vector) would be expected to have ridges (i.e. directions of invariance) in different directions, and that the "dimensionality" of these ridges (we are in a fairly high-dimensional space) gives a hint as to the local dimensionality of the manifold near which the data examples congregate. The singular value spectrum of the Jacobian informs us about that geometry. The number of large singular values should reveal the dimensionality of these ridges, i.e., of that manifold near which examples concentrate. This is illustrated in Figure 2, showing the singular values spectrum of the encoder's Jacobian. The CAE by far does the best job at representing the data variations near a lower-dimensional manifold, and the DAE is second best, while ordinary auto-encoders (regularized or not) do not succeed at all in this respect.

What happens when we stack a CAE on top of another one, to build a deeper encoder? This is illustrated in Figure 3, which shows the average contraction ratio for different distances around each training point, for depth 1 vs depth 2 encoders. Composing two CAEs yields even more contraction and even more non-linearity, i.e. a sharper profile, with a flatter level of contraction at short and medium distances, and a delayed effect of saturation (the bump only comes up at farther distances). We would thus expect higher-level features to be more invariant in their feature-
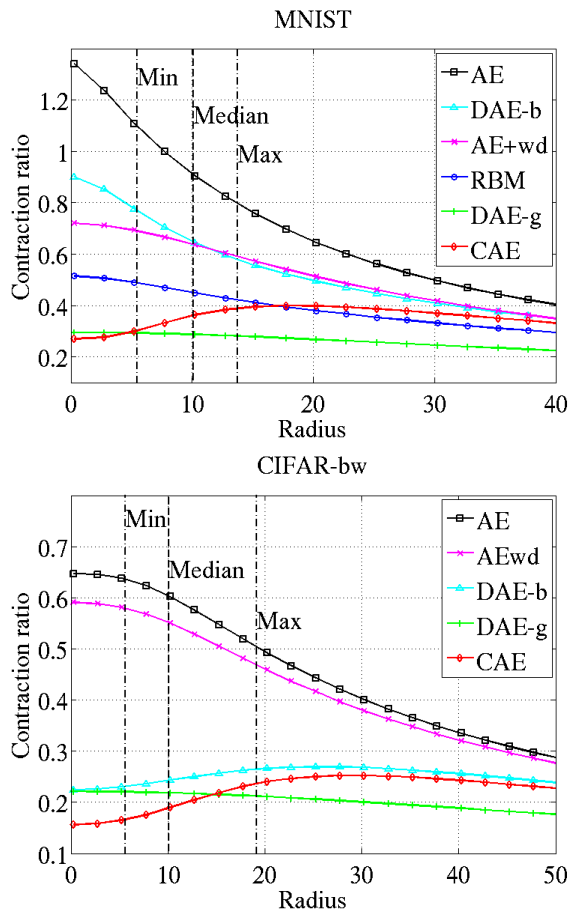
*Figure 1.* Contraction curves obtained with the considered models on MNIST (top) and CIFAR-bw (bottom). See the main text for a detailed interpretation.

specific directions of invariance, which is exactly the kind of property that motivates the use of deeper architectures.

A related approach is tangent propagation (Simard *et al.*, 1992), which uses *prior known transformations* to define the directions to which the final *classification output* should be invariant. By contrast, we encourage invariance of the *hidden representation* in directions *learnt* from the dataset in a fully unsupervised manner. Another related approach is semi-supervised embedding (Weston *et al.*, 2008) in which a contraction is explicitly applied in directions *along* the manifold, obtained from a neighborhood graph. Also it is used in conjunction with a supervised classification cost. By contrast, our unsupervised criterion contracts mostly in directions *orthogonal* to the manifold (due to the reconstruction cost canceling the contraction in the directions parallel to the manifold), and does not explicitly use neighborhood relationships.
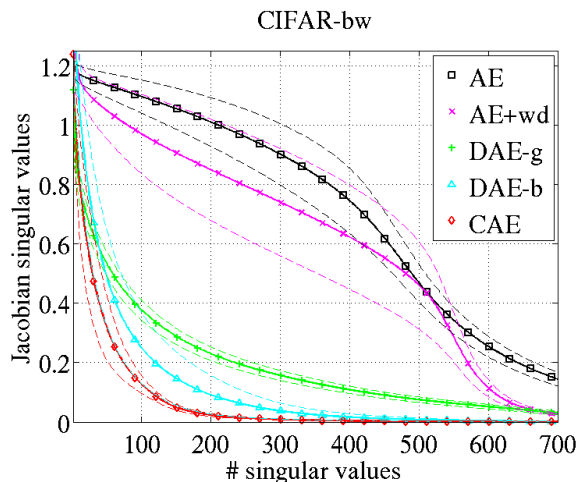


*Figure 2.* Average spectrum of the encoder's Jacobian, for the CIFAR-bw dataset. Large singular values correspond to the local directions of "allowed" variation learnt from the dataset. The CAE having fewer large singular values and a sharper decreasing spectrum, it suggests that it does a better job of characterizing a *low-dimensional manifold* near the training examples.

## 6. Conclusion

In this paper, we attempt to answer the following question: *what makes a good representation?* Besides being useful for a particular task, which we can measure, or towards which we can train a representation, this paper highlights the advantages for representations to be *locally invariant in many directions* of change of the raw input. This idea is implemented by a penalty on the Frobenius norm of the Jacobian matrix of the encoder mapping, which computes the representation. The paper also introduces empirical measures of robustness and invariance, based on the contraction ratio of the learned mapping, at different distances and in different directions around the training examples. We hypothesize that this reveals the manifold structure learned by the model, and we find (by looking at the singular value spectrum of the mapping) that the Contractive Auto-Encoder discovers lower-dimensional manifolds. In addition, experiments on many datasets suggest that this penalty always helps an auto-encoder to perform better, and competes or improves upon the representations learned by Denoising Auto-Encoders or RBMs, in terms of classification error.

## References

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from exam-
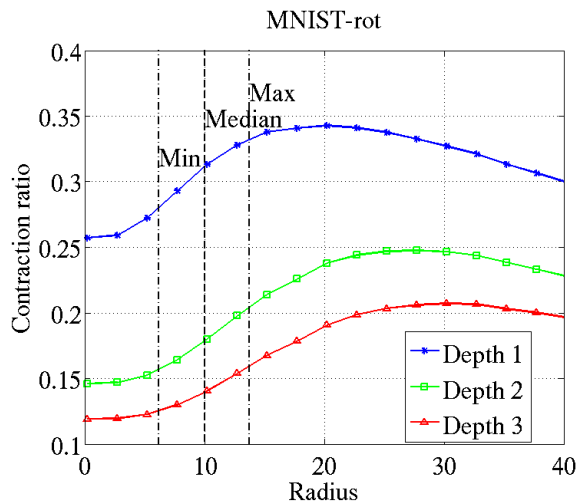
*Figure 3.* Contraction effect as a function of depth. Deeper encoders produce features that are more invariant, over a farther distance, corresponding to flatter ridge of the density in the directions of variation captured.

ples without local minima. *Neural Networks*, **2**, 53–58.

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

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 153–160. MIT Press.

Bishop, C. M. (1995). Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, **7**(1), 108–116.

Hinton, G. E., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, **18**, 1527–1554.

Japkowicz, N., Hanson, S. J., and Gluck, M. A. (2000). Nonlinear autoassociation is not equivalent to PCA. *Neural Computation*, **12**(3), 531–545.

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

Kavukcuoglu, K., Ranzato, M., Fergus, R., and LeCun, Y. (2009). Learning invariant features through topographic filter maps. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'09)*. IEEE.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In Z. Ghahramani, editor, *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML'07)*, pages 473–480. ACM.

Lee, H., Ekanadham, C., and Ng, A. (2008). Sparse deep belief net model for visual area V2. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS'07)*, pages 873–880. MIT Press, Cambridge, MA.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In L. Bottou and M. Littman, editors, *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML'09)*. ACM, Montreal (Qc), Canada.

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

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, **323**, 533–536.

Simard, P., Victorri, B., LeCun, Y., and Denker, J. (1992). Tangent prop - A formalism for specifying selected invariances in an adaptive network. In J. M. S. Hanson and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4 (NIPS'91)*, pages 895–903, San Mateo, CA. Morgan Kaufmann.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, **11**(3371–3408).

Weston, J., Ratle, F., and Collobert, R. (2008). Deep learning via semi-supervised embedding. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, pages 1168–1175, New York, NY, USA. ACM.