Extracting and Composing Robust Features with Denoising Autoencoders

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol
Dept. IRO, Université de Montréal
C.P. 6128, Montreal, Qc, H3C 3J7, Canada
http://www.iro.umontreal.ca/∼lisa

Technical Report 1316, February 2008

Abstract

Previous work has shown that the difficulties in learning deep generative or discriminative models can be overcome by an initial unsupervised learning step that maps inputs to useful intermediate representations. We introduce and motivate a new training principle for unsupervised learning of a representation based on the idea of making the learned representations robust to partial corruption of the input pattern. This approach can be used to train autoencoders, and these denoising autoencoders can be stacked to initialize deep architectures. The algorithm can be motivated from a manifold learning and information theoretic perspective or from a generative model perspective. Comparative experiments clearly show the surprising advantage of corrupting the input of autoencoders on a pattern classification benchmark suite.

1 Introduction

Recent theoretical studies indicate that deep architectures (Bengio & Le Cun, 2007; Bengio, 2007) may be needed to *efficiently* model complex distributions and achieve better generalization performance on challenging recognition tasks. The belief that additional levels of functional composition will yield increased representational and modeling power is not new (McClelland et al., 1986; Hinton, 1989; Utgoff & Stracuzzi, 2002). However, in practice, learning in deep architectures has proven to be difficult. One needs only to ponder the difficult problem of inference in deep directed graphical models, due to "explaining away". Also looking back at the history of multi-layer neural networks, their difficult optimization (Bengio et al., 2007; Bengio, 2007) has long prevented reaping the expected benefits of going beyond one or two hidden layers. However this situation has recently changed with the successful approach of (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio et al., 2007; Ranzato et al., 2007; Lee et al., 2008) for training Deep Belief Networks and stacked autoencoders.

One key ingredient to this success appears to be the use of an unsupervised training criterion to perform a layer-by-layer initialization: each layer is at first trained to produce a higher level (hidden) representation of the observed patterns, based on the representation it receives as input from the layer below, by optimizing a local unsupervised criterion. Each level produces a representation of the input pattern that is more abstract than the previous level's, because it is obtained by composing more operations. This initialization yields a starting point, from which a global fine-tuning of the model's parameters is then performed using another training criterion appropriate for the task at hand. This technique has been shown empirically to avoid getting stuck in the kind of poor solutions one typically reaches with random initializations. While unsupervised learning of a mapping that produces "good" intermediate representations of the input pattern seems to be key, little is understood regarding what constitutes "good" representations for initializing deep architectures, or what explicit criteria may guide learning such representations. We know of only a few algorithms that seem to work well for this purpose: Restricted Boltzmann Machines (RBMs) trained with contrastive divergence on one hand, and various types of autoencoders on the other.

The present research begins with the question of what explicit criteria a good intermediate representation should satisfy. Obviously, it should at a minimum retain a certain amount of "information" about its input, while at the same time being constrained to a given form (e.g. a real-valued vector of a given size in the case of an autoencoder). A supplemental criterion that has been proposed for such models is sparsity of the representation (Ranzato et al., 2008; Lee et al., 2008). Here we hypothesize and investigate an additional specific criterion: **robustness to partial destruction of the input**, i.e., partially destructed inputs should yield almost the same representation. It is motivated by the following informal reasoning: a good representation is expected to capture stable structures in the form of dependencies and regularities characteristic of the (unknown) distribution of its observed input. For high dimensional redundant input (such as images) at least, such structures are likely to depend on evidence gathered from a combination of many input dimensions. They should thus be recoverable from partial observation only. A hallmark of this is our human ability to recognize partially occluded or corrupted images. Further evidence is our ability to form a high level concept associated to multiple modalities (such as image and sound) and recall it even when some of the modalities are missing.

To validate our hypothesis and assess its usefulness as one of the guiding principles in learning deep architectures, we propose a modification to the autoencoder framework to explicitly integrate robustness to partially destroyed inputs. Section 2 describes the algorithm in details. Section 3 discusses links with other approaches in the literature. Section 4 is devoted to a closer inspection of the model from different theoretical standpoints. In section 5 we verify empirically if the algorithm leads to a difference in performance. Section 6 concludes the study.

2 Description of the Algorithm

2.1 Notation and Setup

Let X and Y be two random variables with joint probability density $p(X, Y)$, with marginal distributions $p(X)$ and $p(Y)$. Throughout the text, we will use the following notation: Expectation: $\mathbb{E}_{p(X)}[f(X)] = \int p(\mathbf{x})f(\mathbf{x})d\mathbf{x}$. Entropy: $\mathbb{H}(X) = \mathbb{H}(p) = \mathbb{E}_{p(X)}[-\log p(X)]$. Conditional entropy: $\mathbb{H}(X|Y) = \mathbb{E}_{p(X,Y)}[-\log p(X|Y)]$. Kullback-Leibler divergence: $\mathbb{D}_{\mathsf{KL}}(p\|q) = \mathbb{E}_{p(X)}[\log \frac{p(X)}{q(X)}]$. Cross-entropy: $\mathbb{H}(p\|q) = \mathbb{E}_{p(X)}[-\log q(X)] = \mathbb{H}(p) + \mathbb{D}_{\mathsf{KL}}(p\|q)$. Mutual information: $\mathbf{I}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y)$. Sigmoid: $s(x) = \frac{1}{1+e^{-x}}$ and $s(\mathbf{x}) = (s(\mathbf{x}_1), \ldots, s(\mathbf{x}_d))^T$. Bernoulli distribution with mean μ: $\mathcal{B}_\mu(x)$. and by extension $\mathcal{B}_\mu(\mathbf{x}) = (\mathcal{B}_{\mu_1}(\mathbf{x}_1), \ldots, \mathcal{B}_{\mu_d}(\mathbf{x}_d))$.

The setup we consider is the typical supervised learning setup with a training set of n (input, target) pairs $D_n = \{(\mathbf{x}^{(1)}, t^{(1)}) \ldots, (\mathbf{x}^{(n)}, t^{(n)})\}$, that we suppose to be an i.i.d. sample from an unknown distribution $q(X, T)$ with corresponding marginals $q(X)$ and $q(T)$.

2.2 The Basic Autoencoder

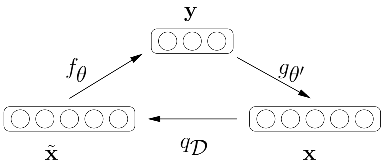
We begin by recalling the traditional autoencoder model such as the one used in (Bengio et al., 2007) to build deep networks. An autoencoder takes an input vector $\mathbf{x} \in [0,1]^d$, and first maps it to a hidden representation $\mathbf{y} \in [0,1]^{d'}$ through a deterministic mapping $\mathbf{y} = f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b})$, parameterized by $\theta = \{\mathbf{W}, \mathbf{b}\}$. $\mathbf{W}$ is a $d' \times d$ weight matrix and $\mathbf{b}$ is a bias vector. The resulting latent representation $\mathbf{y}$ is then mapped back to a "reconstructed" vector $\mathbf{z} \in [0,1]^d$ in input space $\mathbf{z} = g_{\theta'}(\mathbf{y}) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$ with $\theta' = \{\mathbf{W}', \mathbf{b}'\}$. The weight matrix $\mathbf{W}'$ of the reverse mapping may optionally be constrained by $\mathbf{W}' = \mathbf{W}^T$, in which case the autoencoder is said to have *tied weights*. Each training $\mathbf{x}^{(i)}$ is thus mapped to a corresponding $\mathbf{y}^{(i)}$ and a reconstruction $\mathbf{z}^{(i)}$. The parameters of this model are optimized to minimize the *average reconstruction error*:

$$\begin{aligned} \theta^\star, \theta'^\star = & \arg\min_{\theta,\theta'} \frac{1}{n}\sum_{i=1}^{n} L\left(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}\right) \\ = & \arg\min_{\theta,\theta'} \frac{1}{n}\sum_{i=1}^{n} L\left(\mathbf{x}^{(i)}, g_{\theta'}(f_\theta(\mathbf{x}^{(i)}))\right) \end{aligned} \quad (1)$$

where L is a loss function such as the traditional *squared error* $L(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|^2$. An alternative loss, suggested by the interpretation of $\mathbf{x}$ and $\mathbf{z}$ as either bit vectors or vectors of bit probabilities (Bernoullis) is the *reconstruction cross-entropy*:

$$\begin{aligned} L_{\mathbb{H}}(\mathbf{x}, \mathbf{z}) = & \mathbb{H}(\mathcal{B}_\mathbf{x}\|\mathcal{B}_\mathbf{z}) \\ = & -\sum_{k=1}^{d}[\mathbf{x}_k \log \mathbf{z}_k + (1-\mathbf{x}_k)\log(1-\mathbf{z}_k)] \end{aligned} \quad (2)$$

Figure 1: An example $\mathbf{x}$ is corrupted to $\tilde{\mathbf{x}}$. The autoencoder then maps it to $\mathbf{y}$ and attempts to reconstruct $\mathbf{x}$.



Note that if $\mathbf{x}$ is a binary vector, $L_\mathbb{H}(\mathbf{x}, \mathbf{z})$ is a negative log-likelihood for the example $\mathbf{x}$, given the Bernoulli parameters $\mathbf{z}$. Equation 1 with $L = L_\mathbb{H}$ can be written

$$\theta^\star, \theta'^\star = \arg\min_{\theta,\theta'} \mathbb{E}_{q^0(X)} \left[L_\mathbb{H} \left(X, g_{\theta'}(f_\theta(X)) \right) \right] \tag{3}$$

where $q^0(X)$ denotes the empirical distribution associated to our n training inputs. This optimization will typically be carried out by stochastic gradient descent.

2.3 The Denoising Autoencoder

To test our hypothesis and enforce robustness to partially destroyed inputs we modify the basic autoencoder we just described. We will now train it to reconstruct a clean "repaired" input from a *corrupted*, partially destroyed one. This is done by first corrupting the initial input $\mathbf{x}$ to get a partially destroyed version $\tilde{\mathbf{x}}$ by means of a stochastic mapping $\tilde{\mathbf{x}} \sim q_\mathcal{D}(\tilde{\mathbf{x}}|\mathbf{x})$. In our experiments, we considered the following corrupting process, parameterized by the desired proportion ν of "destruction": for each input $\mathbf{x}$, a fixed number νd of components are chosen at random, and their value is forced to 0, while the others are left untouched. The procedure can be viewed as replacing a component considered missing by a default value, which is a common technique. A motivation for zeroing the destroyed components is that it simulates the removal of these components from the input. For images on a white (0) background, this corresponds to "salt noise". Note that alternative corrupting noises could be considered[1]. The corrupted input $\tilde{\mathbf{x}}$ is then mapped, as with the basic autoencoder, to a hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}}) = s(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b})$ from which we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y}) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$ (see figure 1 for a schematic representation of the process). As before the parameters are trained to minimize the average reconstruction error $L_\mathbb{H}(\mathbf{x}, \mathbf{z}) = \mathbb{H}(\mathcal{B}_\mathbf{x} \| \mathcal{B}_\mathbf{z})$ over a training set, i.e. to have $\mathbf{z}$ as close as possible to the uncorrupted input $\mathbf{x}$. But the key difference is that $\mathbf{z}$ is now a deterministic function of $\tilde{\mathbf{x}}$ rather than $\mathbf{x}$ and thus the result of a stochastic mapping of $\mathbf{x}$.

Let us define the joint distribution

$$q^0(X, \widetilde{X}, Y) = q^0(X) q_\mathcal{D}(\widetilde{X}|X) \delta_{f_\theta(\widetilde{X})}(Y) \tag{4}$$

[1] the approach we describe and our analysis is not specific to a particular kind of corrupting noise.

where $\delta_u(v)$ puts mass 0 when $u \neq v$. Thus Y is a deterministic function of $\widetilde{X}$. $q^0(X, \widetilde{X}, Y)$ is parameterized by θ. The objective function minimized by stochastic gradient descent becomes:

$$\arg\min_{\theta,\theta'} \mathbb{E}_{q^0(X,\widetilde{X})} \left[L_{\mathbb{H}} \left(X, g_{\theta'}(f_\theta(\widetilde{X})) \right) \right] . \tag{5}$$

So from the point of view of the stochastic gradient descent algorithm, in addition to picking an input sample from the training set, we will also produce a random corrupted version of it, and take a gradient step towards reconstructing the uncorrupted version from the corrupted version. Note that in this way, the autoencoder cannot learn the identity, unlike the basic autoencoder, thus removing the constraint that $d' < d$ or the need to regularize specifically to avoid such a trivial solution.

2.4 Layer-wise Initialization and Fine Tuning

The basic autoencoder has been used as a building block to train deep networks (Bengio et al., 2007), with the representation of the k-th layer used as input for the $(k+1)$-th, and the $(k+1)$-th layer trained after the k-th has been trained. After a few layers have been trained, the parameters are used as initialization for a network optimized with respect to a supervised training criterion. This greedy layer-wise procedure has been shown to yield significantly better local minima than random initialization of deep networks (Bengio et al., 2007), achieving better generalization on a number of tasks (Larochelle et al., 2007).

The procedure to train a deep network using the denoising autoencoder is similar. The only difference is how each layer is trained, i.e., to minimize the criterion in eq. 5 instead of eq. 3. Note that the corruption process $q_{\mathcal{D}}$ is only used during training, but not for propagating representations from the raw input to higher-level representations. Note also that when layer k is trained, it receives as input the uncorrupted output of the previous layers.

3 Relationship to Other Approaches

Our training procedure for the denoising autoencoder involves learning to recover a clean input from a corrupted version, a task known as *denoising*. The problem of image denoising, in particular, has been extensively studied in the image processing community and many recent developments rely on machine learning approaches (see e.g. Roth and Black (2005); Elad and Aharon (2006); Hammond and Simoncelli (2007)). A particular form of gated autoencoders has also been used for denoising in Memisevic (2007). Denoising using autoencoders was actually introduced much earlier (LeCun, 1987; Gallinari et al., 1987), as an alternative to Hopfield models (Hopfield, 1982). Our objective however is fundamentally different from that of developing a competitive image denoising algorithm. We investigate explicit robustness to corrupting noise only as a criterion to guide the learning of suitable intermediate representations, with the goal to build a better general purpose learning algorithm. Thus our corruption+denoising procedure is applied not only on the input, but also recursively to intermediate representations. It is not specific to images and does not use prior knowledge of image topology.

Whereas the proposed approach does not rely on prior knowledge, it bears resemblance to the well known technique of augmenting the training data with stochastically "transformed" patterns. But again we do not rely on prior knowledge. Moreover we only use the corrupted patterns to optimize an *unsupervised* criterion, as an initialization step.

There are also similarities with the work of (Doi et al., 2006) on robust coding over noisy channels. In their framework, a linear encoder is to encode a clean input for optimal transmission over a noisy channel to a decoder that reconstructs the input. This work was later extended to robustness to noise in the input, in a proposal for a model of retinal coding (Doi & Lewicki, 2007). Though some of the inspiration behind our work comes from neural coding and computation, our goal is not to account for experimental data of neuronal activity as in (Doi & Lewicki, 2007). Also, the non-linearity of our denoising autoencoder is crucial for its use in initializing a deep neural network.

It may be objected that, if our goal is to handle *missing values* correctly, we could have more naturally defined a proper latent variable generative model, and infer the posterior over the latent (hidden) representation in the presence of missing inputs. But this usually requires a costly marginalization[2] which has to be carried out for each new example. By contrast, our approach tries to learn a fast and robust deterministic mapping f_θ from examples of *already corrupted* inputs. The burden is on learning such a constrained mapping during training, rather than on unconstrained inference at use time. We expect this may force the model to capture implicit invariances in the data, and result in interesting features. Also note that in section 4.4 we will see how our learning algorithm for the denoising autoencoder can be viewed as a form of variational inference in a particular generative model.

4 Analysis of the Denoising Autoencoder

The above intuitive motivation for the denoising autoencoder was given with the perspective of discovering robust representations. In the following, which can be skipped without hurting the remainder of the paper, we try to gain insight by considering several alternative perspectives on the algorithm.

4.1 Manifold Learning Perspective

The process of mapping a corrupted example to an uncorrupted one can be visualized in Figure 2, with a low-dimensional manifold near which the data concentrate. We learn a stochastic operator $p(X|\widetilde{X})$ that maps an $\widetilde{X}$ to an X, $p(X|\widetilde{X}) = \mathcal{B}_{g_{\theta'}(f_\theta(\widetilde{X}))}(X)$. The corrupted examples will be much more likely to

[2] as in the case of RBMs, where it is exponential in the number of missing values

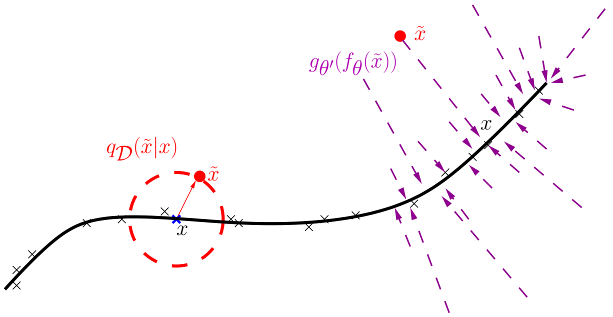

$\tilde{x}$
$g_{\theta'}(f_\theta(\tilde{x}))$
x
$q_{\mathcal{D}}(\tilde{x}|x)$
$\tilde{x}$
x

Figure 2: Illustration of what the denoising autoencoder is trying to learn. Suppose training data (crosses) concentrate near a low-dimensional manifold. A corrupted example (circle) is obtained by applying a corruption process $q_{\mathcal{D}}(\widetilde{X}|X)$ (left side). Corrupted examples (circles) are typically outside and farther from the manifold, hence the model learns with $p(X|\widetilde{X})$ to map points to more likely points (right side). Mapping from more corrupted examples requires bigger jumps (longer dashed arrows).

be outside and farther from the manifold than the uncorrupted ones. Hence the stochastic operator $p(X|\widetilde{X})$ learns a map that tends to go from lower probability points $\widetilde{X}$ to high probability points X, generally on or near the manifold. Note that when $\widetilde{X}$ is farther from the manifold, $p(X|\widetilde{X})$ should learn to make bigger steps, to reach the manifold. At the limit we see that the operator should map even far away points to a small volume near the manifold.

The denoising autoencoder can thus be seen as a way to define and learn a manifold. The intermediate representation $Y = f(X)$ can be interpreted as a coordinate system for points on the manifold (this is most clear if we force the dimension of Y to be smaller than the dimension of X). More generally, one can think of $Y = f(X)$ as a representation of X which is well suited to capture the main variations in the data, i.e., on the manifold. When additional criteria (such as sparsity) are introduced in the learning model, one can no longer directly view $Y = f(X)$ as an explicit low-dimensional coordinate system for points on the manifold, but it retains the property of capturing the main factors of variation in the data.

4.2 The Stochastic Operator Perspective

The denoising autoencoder can be seen as corresponding to a semi-parametric model from which we can sample. Let us augment the set of modeled random variables to include the corrupted example $\widetilde{X}$ in addition to the corresponding uncorrupted example X, and let us perform maximum likelihood training on a model of their joint. We consider here the simpler case where X is discrete, but the approach can be generalized. We define a joint distribution $p(X, \widetilde{X}) = p(\widetilde{X})p(X|\widetilde{X})$ from the stochastic operator $p(X|\widetilde{X})$, with marginal $p(\widetilde{X}) = q^0(\widetilde{X})$ set by construction.

We now have an empirical distribution q^0 and a model p on $(X, \widetilde{X})$ pairs.

Performing maximum likelihood on them or minimizing $\mathbb{D}_{\mathsf{KL}}(q^0(X,\widetilde{X})\|p(X,\widetilde{X}))$ is a reasonable training objective, again yielding the denoising criterion in eq. 5.

As an additional motivation for minimizing $\mathbb{D}_{\mathsf{KL}}(q^0(X,\widetilde{X})\|p(X,\widetilde{X}))$, note that as we minimize it (i.e., $\mathbb{D}_{\mathsf{KL}}(q^0(X,\widetilde{X})\|p(X,\widetilde{X})) \to 0$), the marginals of p approach those of q^0, hence in particular

$$p(X) \to q^0(X),$$

i.e., training in this way corresponds to a semi-parametric model $p(X)$ which approaches the empirical distribution $q^0(X)$. By applying the marginalization definition for $p(X)$, we see what the corresponding model is

$$p(X) = \frac{1}{n}\sum_{i=1}^{n}\sum_{\tilde{\mathbf{x}}} p(X|\widetilde{X}=\tilde{\mathbf{x}})q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x}_i) \tag{6}$$

where $\mathbf{x}_i$ is one of the n training examples. Note that only the parameters of $p(X|\widetilde{X})$ are optimized in this model. Note also that sampling from the model is easy. We have thus see that the denoising autoencoder learns a semi-parametric model which can be sampled from, based on the stochastic operator $p(X|\widetilde{X})$.

What would happen if we were to apply this operator repeatedly? That would define a chain $p_k(X)$ where $p_0(X=\mathbf{x}) = q^0(\widetilde{X}=\mathbf{x})$, $p_1(X) = p(X)$ and $p_k(X) = \sum_{\mathbf{x}} p(X|\widetilde{X}=\mathbf{x})p_{k-1}(\mathbf{x})$. If the operator was ergodic (which is plausible, following the above argumentation), its fixed point would define yet another distribution $\pi(X) = \lim_{k\to\infty} p_k(X)$, of which the semi-parametric $p(X)$ would be a first-order approximation. The advantage of this formulation is that $\pi(X)$ is purely parametric: it does not explicitly depend on the empirical distribution q^0.

4.3 Bottom-up Filtering, Information Theoretic Perspective

In this section we adopt a bottom-up filtering viewpoint, and an information-theoretic perspective. Let $X, \widetilde{X}, Y$ be random variables representing respectively an input sample, a corrupted version of it, and the corresponding hidden representation, i.e. $X \sim q(X)$ (the true generating process for X), $\widetilde{X} \sim q_{\mathcal{D}}(\widetilde{X}|X)$, $Y = f_\theta(\widetilde{X})$, with the associated joint $q(X,\widetilde{X},Y)$. Notice that it is defined with the same dependency structure as $q^0(X,\widetilde{X},Y)$, and the same conditionals $\widetilde{X}|X$ and $Y|\widetilde{X}$.

The role of the greedy layer initialization is to learn a non-linear filter that yields a good representation of its input for the next layer. A good representation should retain a sufficient amount of "information" about its input, while we might at the same time encourage its marginal distribution to display certain desirable properties. From this high level perspective, the filter we learn, which has a fixed parameterized form with parameters θ, should be optimized to realize a tradeoff between yielding a somewhat easier to model marginal distribution and retaining as much information as possible about the input. This can be formalized as optimizing an objective function of the form $\arg\max_\theta \{\mathbf{I}(X;Y) + \lambda \mathcal{J}(Y)\}$, where $\mathcal{J}$ is a functional that induces some preference over the marginal $q(Y)$, and hyper-parameter λ controls the tradeoff. $\mathcal{J}$ could for example encourage $\mathbb{D}_{\mathsf{KL}}$ closeness to some reference prior distribution, or be some measure of sparsity or independence. In the present study, however, we shall suppose that Y is only constrained by its dimensionality. This corresponds to the case where $\mathcal{J}(Y)$ is constant for all distributions (i.e. no preference).

Let us thus examine only the first term. We have $\mathbf{I}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y)$. If we suppose the unknown input distribution $q(X)$ fixed, $\mathbb{H}(X)$ is a constant. Maximizing mutual information then amounts to maximizing $-\mathbb{H}(X|Y)$, i.e.

$$\begin{aligned}
\arg\max_\theta \mathbf{I}(X;Y) &= \arg\max_\theta -\mathbb{H}(X|Y) \\
\max_\theta -\mathbb{H}(X|Y) &= \max_\theta \mathbb{E}_{q(X,Y)}[\log q(X|Y)] \\
&= \max_{\theta,p^\star} \mathbb{E}_{q(X,Y)}[\log p^\star(X|Y)]
\end{aligned}$$

where we optimize over all possible distributions $p^\star$. It is easy to show that the maximum is obtained for $p^\star(X|Y) = q(X|Y)$. If instead we constrain $p^\star(X|Y)$ to a given parametric form $p(X|Y)$ parameterized by θ', we get a lower bound:

$$\max_\theta -\mathbb{H}(X|Y) \geq \max_{\theta,\theta'} \mathbb{E}_{q(X,Y)}[\log p(X|Y)]$$

where we have an equality if $\exists\theta' \; q(X|Y) = p(X|Y)$.

In the case of an autoencoder with binary vector variable X, we can view the top-down reconstruction as representing a $p(X|Y) = \mathcal{B}_{g_{\theta'}(Y)}(X)$. Optimizing for the lower bound leads to:

$$\begin{aligned}
&\max_{\theta,\theta'} \mathbb{E}_{q(X,Y)}[\log \mathcal{B}_{g_{\theta'}(Y)}(X)] \\
&= \max_{\theta,\theta'} \mathbb{E}_{q(X,\widetilde{X})}[\log \mathcal{B}_{g_{\theta'}(f_\theta(\widetilde{X}))}(X)] \\
&= \min_{\theta,\theta'} \mathbb{E}_{q(X,\widetilde{X})}[L_{\mathbb{H}}(X, g_{\theta'}(f_\theta(\widetilde{X})))]
\end{aligned}$$

where in the second line we use the fact that $Y = f_\theta(\widetilde{X})$ deterministically. This is the criterion we use for training the autoencoder (eq. 5), but replacing the true generating process q by the empirical q^0.

This shows that minimizing the expected reconstruction error amounts to maximizing a lower bound on $\mathbb{H}(X|Y)$ and at the same time on the mutual information between input X and the hidden representation Y. Note that this reasoning holds equally for the basic autoencoder, but with the denoising autoencoder, we maximize the mutual information between X and Y *even as Y is a function of corrupted input.*

4.4 Top-down, Generative Model Perspective

In this section we try to recover the training criterion for our denoising autoencoder (eq. 5) from a generative model perspective. Specifically we show that training the denoising autoencoder as described in section 2.3 is equivalent to maximizing a variational bound on a particular generative model.

Consider the generative model $p(X, \widetilde{X}, Y) = p(Y)p(X|Y)p(\widetilde{X}|X)$ where $p(X|Y) = \mathcal{B}_{s(\mathbf{W}'Y+\mathbf{b}')}$ and $p(\widetilde{X}|X) = q_{\mathcal{D}}(\widetilde{X}|X)$. $p(Y)$ is a uniform prior over Y. This defines a generative model with parameter set $\theta' = \{\mathbf{W}', \mathbf{b}'\}$. We will use the previously defined $q^0(X, \widetilde{X}, Y) = q^0(X)q_{\mathcal{D}}(\widetilde{X}|X)\delta_{f_\theta(\widetilde{X})}(Y)$ (equation 4) as an auxiliary model in the context of a variational approximation of the log-likelihood of $p(\widetilde{X})$. Note that we abuse notation to make it lighter, and use the same letters X, $\widetilde{X}$ and Y for different sets of random variables representing the same quantity under different distributions: p or q^0. Keep in mind that whereas we had the dependency structure $X \rightarrow \widetilde{X} \rightarrow Y$ for q or q^0, we have $Y \rightarrow X \rightarrow \widetilde{X}$ for p.

Since p contains a corruption operation at the last generative stage, we propose to fit $p(\widetilde{X})$ to corrupted training samples. Performing maximum likelihood fitting for samples drawn from $q^0(\widetilde{X})$ corresponds to minimizing the cross-entropy, or maximizing

$$\begin{aligned}\mathcal{H} &= \max_{\theta'}\{-\mathbb{H}(q^0(\widetilde{X})\|p(\widetilde{X}))\} \\ &= \max_{\theta'}\{\mathbb{E}_{q^0(\widetilde{X})}[\log p(\widetilde{X})]\}. \qquad (7)\end{aligned}$$

Let $q^\star(X, Y|\widetilde{X})$ be a conditional density, the quantity $\mathcal{L}(q^\star, \widetilde{X}) = \mathbb{E}_{q^\star(X,Y|\widetilde{X})}\left[\log \frac{p(X,\widetilde{X},Y)}{q^\star(X,Y|\widetilde{X})}\right]$ is a lower bound on $\log p(\widetilde{X})$ since the following can be shown to be true for any $q^\star$:

$$\log p(\widetilde{X}) = \mathcal{L}(q^\star, \widetilde{X}) + \mathbb{D}_{\mathsf{KL}}(q^\star(X, Y|\widetilde{X})\|p(X, Y|\widetilde{X}))$$

Also it is easy to verify that the bound is tight when $q^\star(X, Y|\widetilde{X}) = p(X, Y|\widetilde{X})$, where the $\mathbb{D}_{\mathsf{KL}}$ becomes 0. We can thus write $\log p(\widetilde{X}) = \max_{q^\star} \mathcal{L}(q^\star, \widetilde{X})$, and consequently rewrite equation 7 as

$$\begin{aligned}\mathcal{H} &= \max_{\theta'}\{\mathbb{E}_{q^0(\widetilde{X})}[\max_{q^\star} \mathcal{L}(q^\star, \widetilde{X})]\} \\ &= \max_{\theta', q^\star}\{\mathbb{E}_{q^0(\widetilde{X})}[\mathcal{L}(q^\star, \widetilde{X})]\} \qquad (8)\end{aligned}$$

where we moved the maximization outside of the expectation because an unconstrained $q^\star(X, Y|\widetilde{X})$ can in principle perfectly model the conditional distribution needed to maximize $\mathcal{L}(q^\star, \widetilde{X})$ for any $\widetilde{X}$. Now if we replace the maximization over an unconstrained $q^\star$ by the maximization over the parameters θ of our q^0 (appearing in f_θ that maps an $\mathbf{x}$ to a $\mathbf{y}$), we get a lower bound on $\mathcal{H}$

$$\mathcal{H} \geq \max_{\theta', \theta}\{\mathbb{E}_{q^0(\widetilde{X})}[\mathcal{L}(q^0, \widetilde{X})]\} \qquad (9)$$

Maximizing this lower bound, we find

$$\begin{aligned}
&\arg\max_{\theta,\theta'}\{\mathbb{E}_{q^0(\widetilde{X})}[\mathcal{L}(q^0,\widetilde{X})]\}\\
&=\arg\max_{\theta,\theta'}\mathbb{E}_{q^0(X,\widetilde{X},Y)}\left[\log\frac{p(X,\widetilde{X},Y)}{q^0(X,Y|\widetilde{X})}\right]\\
&=\arg\max_{\theta,\theta'}\mathbb{E}_{q^0(X,\widetilde{X},Y)}\Big[\log p(X,\widetilde{X},Y)\Big]\\
&\qquad+\mathbb{E}_{q^0(\widetilde{X})}\Big[\mathbb{H}[q^0(X,Y|\widetilde{X})]\Big]\\
&=\arg\max_{\theta,\theta'}\mathbb{E}_{q^0(X,\widetilde{X},Y)}\Big[\log p(X,\widetilde{X},Y)\Big].
\end{aligned}$$

Note that θ only occurs in $Y=f_\theta(\widetilde{X})$, and θ' only occurs in $p(X|Y)$. The last line is therefore obtained because $q^0(X|\widetilde{X})\propto q_{\mathcal{D}}(\widetilde{X}|X)q^0(X)$ (none of which depends on (θ,θ')), and $q^0(Y|\widetilde{X})$ is deterministic, i.e., its entropy is constant, irrespective of (θ,θ'). Hence the entropy of $q^0(X,Y|\widetilde{X})=q^0(Y|\widetilde{X})q^0(X|\widetilde{X})$, does not vary with (θ,θ'). Finally, following from above, we obtain our training criterion (eq. 5):

$$\begin{aligned}
&\arg\max_{\theta,\theta'}\mathbb{E}_{q^0(\widetilde{X})}[\mathcal{L}(q^0,\widetilde{X})]\\
&=\arg\max_{\theta,\theta'}\mathbb{E}_{q^0(X,\widetilde{X},Y)}[\log[p(Y)p(X|Y)p(\widetilde{X}|X)]]\\
&=\arg\max_{\theta,\theta'}\mathbb{E}_{q^0(X,\widetilde{X},Y)}[\log p(X|Y)]\\
&=\arg\max_{\theta,\theta'}\mathbb{E}_{q^0(X,\widetilde{X})}[\log p(X|Y=f_\theta(\widetilde{X}))]\\
&=\arg\min_{\theta,\theta'}\mathbb{E}_{q^0(X,\widetilde{X})}\Big[L_{\mathbb{H}}\Big(X,g_{\theta'}(f_\theta(\widetilde{X}))\Big)\Big]
\end{aligned}$$

where the third line is obtained because (θ,θ') have no influence on $\mathbb{E}_{q^0(X,\widetilde{X},Y)}[\log p(Y)]$ because we chose $p(Y)$ uniform, i.e. constant, nor on $\mathbb{E}_{q^0(X,\widetilde{X})}[\log p(\widetilde{X}|X)]$, and the last line is obtained by inspection of the definition of $L_{\mathbb{H}}$ in eq. 2, when $p(X|Y=f_\theta(\widetilde{X}))$ is a $\mathcal{B}_{g_{\theta'}(f_\theta(\widetilde{X}))}$.

5 Experiments

We performed experiments with the proposed algorithm on the same benchmark of classification problems used in (Larochelle et al., 2007)[3]. It contains different variations of the MNIST digit classification problem, with added factors of variation such as rotation (*rot*), addition of a background composed of random pixels (*bg-rand*) or made from patches extracted from a set of images (*bg-img*), or combinations of these factors (*rot-bg-img*). These variations render the problems

[3] All the datasets for these problems were taken from http://www.iro.umontreal.ca/∼lisa/icml2007.

Table 1: **Comparison of stacked denoising autoencoders (SdA-3) with other models.**
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. SdA-3 appears to achieve performance superior or equivalent to the best other model on all problems except *bg-rand*. For SdA-3, we also indicate the fraction ν of destroyed input components, as chosen by proper model selection. Note that SAA-3 is equivalent to SdA-3 with $\nu = 0\%$.

Dataset	**SVM**$_{rbf}$	**SVM**$_{poly}$	**DBN-1**	**SAA-3**	**DBN-3**	**SdA-3** (ν)
basic	**3.03±0.15**	3.69±0.17	3.94±0.17	3.46±0.16	3.11±0.15	**2.80±0.14** (10%)
rot	11.11±0.28	15.42±0.32	14.69±0.31	**10.30±0.27**	**10.30±0.27**	**10.29±0.27** (10%)
bg-rand	14.58±0.31	16.62±0.33	9.80±0.26	11.28±0.28	**6.73±0.22**	10.38±0.27 (40%)
bg-img	22.61±0.37	24.01±0.37	**16.15±0.32**	23.00±0.37	**16.31±0.32**	**16.68±0.33** (25%)
rot-bg-img	55.18±0.44	56.41±0.43	52.21±0.44	51.93±0.44	47.39±0.44	**44.49±0.44** (25%)
rect	**2.15±0.13**	**2.15±0.13**	4.71±0.19	2.41±0.13	2.60±0.14	**1.99±0.12** (10%)
rect-img	24.04±0.37	24.05±0.37	23.69±0.37	24.05±0.37	22.50±0.37	**21.59±0.36** (25%)
convex	19.13±0.34	19.82±0.35	19.92±0.35	**18.41±0.34**	**18.63±0.34**	**19.06±0.34** (10%)

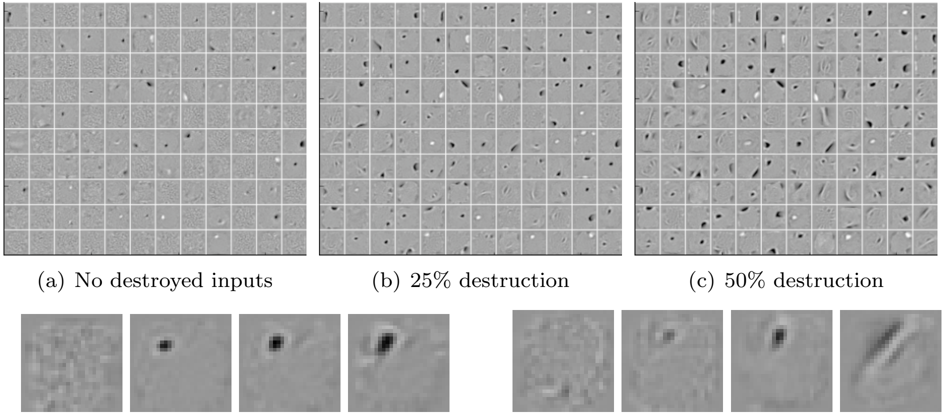
particularly challenging for current generic learning algorithms. Each problem is divided into a training, validation, and test set (10000, 2000, 50000 examples respectively). A subset of the original MNIST problem is also included with the same example set sizes (problem *basic*). The benchmark also contains additional binary classification problems: discriminating between convex and non-convex shapes (*convex*), and between wide and long rectangles (*rect*, *rect-img*).

Neural networks with 3 hidden layers initialized by stacking *denoising autoencoders* (SdA-3), and fine tuned on the classification tasks, were evaluated on all the problems in this benchmark. Model selection was conducted following a similar procedure as Larochelle et al. (2007). Several values of hyper parameters (destruction fraction ν, layer sizes, number of unsupervised training epochs) were tried, combined with early stopping in the fine tuning phase. For each task, the best model was selected based on its classification performance on the validation set.

Table 1 reports the resulting classification error on the test set for the new model (SdA-3), together with the performance reported in Larochelle et al. (2007)[4] for SVMs with Gaussian and polynomial kernels, 1 and 3 hidden layers deep belief network (DBN-1 and DBN-3) and a 3 hidden layer deep network initialized by stacking basic autoencoders (SAA-3). Note that SAA-3 is equivalent to a SdA-3 with $\nu = 0\%$ destruction. As can be seen in the table, the corruption+denoising training works remarkably well as an initialization step, and in most cases yields significantly better classification performance than basic autoencoder stacking with no noise. On all but one task the SdA-3 algorithm performs on par or better than the best other algorithms, including deep belief

[4] Except that *rot* and *rot-bg-img*, as reported on the website from which they are available, have been regenerated since Larochelle et al. (2007), to fix a problem in the initial data generation process. We used the updated data and corresponding benchmark results given on this website.

Figure 3: **Filters obtained after training the first denoising autoencoder.**
(a-c) show some of the filters obtained after training a denoising autoencoder on MNIST samples, with increasing destruction levels ν. The filters at the same position in the three images are related only by the fact that the autoencoders were started from the same random initialization point.
(d) and **(e)** zoom in on the filters obtained for two of the neurons, again for increasing destruction levels.
As can be seen, with no noise, many filters remain similarly uninteresting (undistinctive almost uniform grey patches). As we increase the noise level, denoising training forces the filters to differentiate more, and capture more distinctive features. Higher noise levels tend to induce less local filters, as expected. One can distinguish different kinds of filters, from local blob detectors, to stroke detectors, and some full character detectors at the higher noise levels.



(a) No destroyed inputs

(b) 25% destruction

(c) 50% destruction

(d) Neuron A (0%, 10%, 20%, 50% destruction)

(e) Neuron B (0%, 10%, 20%, 50% destruction)

nets. This suggests that our proposed procedure was indeed able to produce more useful feature detectors.

Next, we wanted to understand qualitatively the effect of the corruption+denoising training. To this end we display the filters obtained after initial training of the first denoising autoencoder on MNIST digits. Figure 3 shows a few of these filters as little image patches, for different noise levels. Each patch corresponds to a row of the learnt weight matrix $\mathbf{W}$, i.e. the incoming weights of one of the hidden layer neurons. The beneficial effect of the denoising training can clearly be seen. Without the denoising procedure, many filters appear to have learnt no interesting feature. They look like the filters obtained after random initialization. But when increasing the level of destructive corruption, an increasing number of filters resemble sensible feature detectors. As we move to higher noise levels, we observe a phenomenon that we expected: filters become less local, they appear sensitive to larger structures spread out across more input dimensions.

6 Conclusion and Future Work

We have introduced a very simple training principle for autoencoders, based on the objective of undoing a corruption process. This is motivated by the goal of learning representations of the input that are robust to small irrelevant changes in input. Several perspectives also help to motivate it from a manifold learning perspective and from the perspective of a generative model.

This principle can be used to train and stack autoencoders to initialize a deep neural network. A series of image classification experiments were performed to evaluate this new training principle. The empirical results support the following conclusions: unsupervised initialization of layers with an explicit denoising criterion helps to capture interesting structure in the input distribution. This in turn leads to intermediate representations much better suited for subsequent learning tasks such as supervised classification. The experimental results with Deep Belief Networks (whose layers are initialized as RBMs) suggest that RBMs may also encapsulate a form of robustness in the representations they learn, possibly because of their stochastic nature, which introduces noise in the representation during training. Future work inspired by this observation should investigate other types of corruption process, not only of the input but of the representation itself as well.

References

Bengio, Y. (2007). *Learning deep architectures for AI* (Technical Report 1312). Université de Montréal, dept. IRO.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems 19* (pp. 153–160). MIT Press.

Bengio, Y., & Le Cun, Y. (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste and J. Weston (Eds.), *Large scale kernel machines*. MIT Press.

Doi, E., Balcan, D. C., & Lewicki, M. S. (2006). A theoretical analysis of robust coding over noisy overcomplete channels. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*, 307–314. Cambridge, MA: MIT Press.

Doi, E., & Lewicki, M. S. (2007). A theory of retinal population coding. *NIPS* (pp. 353–360). MIT Press.

Elad, M., & Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, *15*, 3736–3745.

Gallinari, P., LeCun, Y., Thiria, S., & Fogelman-Soulie, F. (1987). Memoires associatives distribuees. *Proceedings of COGNITIVA 87*. Paris, La Villette.

Hammond, D., & Simoncelli, E. (2007). A machine learning framework for adaptive combination of signal denoising methods. *2007 International Conference on Image Processing* (pp. VI: 29–32).

Hinton, G. (1989). Connectionist learning procedures. *Artificial Intelligence*, *40*, 185–234.

Hinton, G., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504–507.

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

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, *79*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. *Twenty-fourth International Conference on Machine Learning (ICML'2007).*

LeCun, Y. (1987). *Modèles connexionistes de l'apprentissage.* Doctoral dissertation, Université de Paris VI.

Lee, H., Ekanadham, C., & Ng, A. (2008). Sparse deep belief net model for visual area V2. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20.* Cambridge, MA: MIT Press.

McClelland, J., Rumelhart, D., & the PDP Research Group (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*, vol. 2. Cambridge: MIT Press.

Memisevic, R. (2007). *Non-linear latent factor models for revealing structure in high-dimensional data.* Doctoral dissertation, Departement of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Ranzato, M., Boureau, Y.-L., & LeCun, Y. (2008). Sparse feature learning for deep belief networks. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20.* Cambridge, MA: MIT Press.

Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. *Advances in Neural Information Processing Systems (NIPS 2006).* MIT Press.

Roth, S., & Black, M. (2005). Fields of experts: a framework for learning image priors. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 860–867).

Utgoff, P., & Stracuzzi, D. (2002). Many-layered learning. *Neural Computation*, *14*, 2497–2539.