Deep Generative Stochastic Networks Trainable by Backprop

Yoshua Bengio and Éric Thibodeau-Laufer
Département d'informatique et recherche opérationnelle
Université de Montréal

Abstract

Recent work showed that denoising auto-encoders can be interpreted as generative models. We generalize these results to arbitrary parametrizations that learn to reconstruct their input and where noise is injected, not just in input, but also in intermediate computations. We show that under reasonable assumptions (the parametrization is rich enough to provide a consistent estimator, and it prevents the learner from just copying its input in output and producing a dirac output distribution), such models are consistent estimators of the data generating distributions, and that they define the estimated distribution through a Markov chain that consists at each step in re-injecting sampled reconstructions as a sequence of inputs into the unfolded computational graph. As a consequence, one can define deep architectures similar to deep Boltzmann machines in that units are stochastic, that the model can learn to generate a distribution similar to its training distribution, that it can easily handle missing inputs, but without the troubling problem of intractable partition function and intractable inference as stumbling blocks for both training and using these models. In particular, we argue that if the underlying latent variables of a graphical model form a highly multimodal posterior (given the input), none of the currently known training methods can appropriately deal with this multimodality (when the number modes is much greater than the number of MCMC samples one is willing to perform, and when the structure of the posterior cannot be easily approximated by some tractable variational approximation). In contrast, the proposed models can simply be trained by back-propagating the reconstruction error (seen as log-likelihood of reconstruction) into the parameters, benefiting from the power and ease of training recently demonstrated for deep supervised networks with dropout noise.

1 Introduction

Research in deep learning (see Bengio (2009) and Bengio *et al.* (2013c) for reviews) has started with breakthroughs in unsupervised learning of representations, based mostly on the Restricted Boltzmann Machine (RBM) (Hinton *et al.*, 2006), auto-encoder variants (Bengio *et al.*, 2007; Vincent *et al.*, 2008), and sparse coding variants (Lee *et al.*, 2007; Ranzato *et al.*, 2007). However, the most impressive recent results have been obtained with purely supervised learning techniques for deep networks, in particular for speech recognition (Dahl *et al.*, 2010; Deng *et al.*, 2010; Seide *et al.*, 2011) and object recognition (Krizhevsky *et al.*, 2012). In all of these cases, the availability of large quantities of labeled data was important, and the latest breakthrough in object recognition (Krizhevsky *et al.*, 2012) was achieved with fairly deep convolutional networks with a form of noise injection in the input and hidden layers during training, called dropout (Hinton *et al.*, 2012).

On the other hand, progress with deep unsupervised architectures has been slower. Although single-layer unsupervised learners are fairly well developed, jointly training all the layers with respect to a single unsupervised criterion remains a challenge, with the best current options being the Deep

Belief Network (DBN) (Hinton *et al.*, 2006) and the Deep Boltzmann Machine (DBM) (Salakhutdinov and Hinton, 2009). Nonetheless, joint unsupervised training of all the layers remains a difficult, much more so than for its supervised counterparts. Since the amount of unlabeled data potentially available is very large, it would be very interesting to develop unsupervised learning algorithms for generative deep architectures that can take advantage of the progress in techniques for training deep supervised ones, i.e., based on back-propagated gradients. The approach presented here is one step in this direction, allowing to train jointly train all the levels of representation of a deep unsupervised probabilistic model solely by back-propagating gradients.

Another motivation for the approach presented here is a potential problem with probabilistic models with anonymous latent variables[1], discussed in more detail in Section 2. The gist of the issue is the following. Graphical models with latent variables often require dealing with either or both of the following fundamentally difficult problems in the inner loop of training, or to actually use the model for taking decisions: inference (estimating the posterior distribution over latent variables h given inputs x) and sampling (from the joint model of h and x). However, if the posterior $P(h|x)$ has a huge number of modes that matter, then all of the current approaches may be doomed for such tasks.

The main contribution of this paper is a theoretical extension of recent work (summarized in Section 3) on the generative view of denoising auto-encoders. It provides a statistically consistent way of estimating the underlying data distribution based on a denoising-like criterion where the noise can be injected not just in the input but anywhere in the computational graph that produces the predicted distribution for the denoised input.

We apply this idea to deep Generative Stochastic Networks (GSNs) whose computational graph resembles the one followed by Gibbs sampling in deep Boltzmann machines, but that can be trained efficiently with back-propagated gradients. The models can be trained and used in the presence of missing inputs, and they can be used to sample from the learned distribution (possibly conditioning on some of the inputs).

2 A Potential Problem with Anonymous Latent Variable Models

All of the graphical models studied for deep learning except the humble RBM require a non-trivial form of inference, i.e., guessing values of the latent variables h that are appropriate for the given visible input x. Several forms of inference have been investigated in the past: MAP inference is formulated like an optimization problem (looking for h that approximately maximizes $P(h \mid x)$); MCMC inference attempts to sample a sequence of h's from $P(h \mid x)$; variational inference looks for a simple (typically factorial) approximate posterior $q_x(h)$ that is close to $P(h \mid x)$, and usually involves an iterative optimization procedure. See a recent machine learning textbook for more details (Murphy, 2012).

In addition, a challenge related to inference is sampling (not just from $P(h \mid x)$ but also from $P(h, x)$ or $P(x)$), which like inference is often needed in the inner loop of learning algorithms for probabilistic models with latent variables such as energy-based models (LeCun *et al.*, 2006) or Markov Random Fields, where $P(x)$ or $P(h, x)$ is defined in terms of a parametrized energy function whose normalized exponential gives probabilities. Deep Boltzmann machines (Salakhutdinov and Hinton, 2009) combine the difficulty of inference (for the *"positive phase"* where one tries to push the energies associated with the observed x down) and also that of sampling (for the *"negative phase"* where one tries to push up the energies associated with x's sampled from $P(x)$). Sampling for the negative phase is usually done by MCMC, although some unsupervised learning algorithms (Collobert and Weston, 2008; Gutmann and Hyvarinen, 2010; Bordes *et al.*, 2013) involve "negative examples" that are sampled through simpler procedures (like perturbations of the observed input, in a spirit reminiscent of the approach presented here). In Salakhutdinov and Hinton (2009), inference for the positive phase is achieved with a mean-field variational approximation.[2]

[1] they are called anonymous because no a priori semantics is assigned to them, like in Boltzmann machines, and unlike in many knowledge-based graphical models. Whereas inference over non-anonymous latent variables is required to make sense of the model, anonymous variables are only a device to capture the structure of the distribution and need not have a clear human-readable meaning.

[2] In the mean-field approximation, computation proceeds like in Gibbs sampling, but with stochastic binary values replaced by their conditional expected value (probability of being 1), given the outputs of the other units.

2.1 Potentially Huge Number of Modes.

The challenge we propose to think about has to do with the potential existence of highly multimodal posteriors: all of the currently known approaches to inference and sampling are making very strong explicit or implicit assumptions on the form the distribution of interest ($P(h \mid x)$ or $P(h, x)$). As we argue below, these approaches make sense if this target distribution is either approximately unimodal (MAP), (conditionally) factorizes (variational approximations, i.e., the different factors h_i are approximately independent[3] of each other given x), or has only a few modes between which it is easy to mix (MCMC). However, approximate inference can be potentially hurtful, not just at test time but for training (Kulesza and Pereira, 2008), because it is often in the inner loop of the learning procedure. We want to consider here the case where neither a unimodal assumption (MAP), the assumption of a few major modes (MCMC) or of fitting a variational approximation (factorial or tree-structured distribution) are appropriate.

Imagine for example that h represents many explanatory variables of a rich audio-visual "scene" with a highly ambiguous raw input x, including the presence of several objects with ambiguous attributes or categories, such that one cannot really disambiguate one of the objects independently of the others (the so-called "structured output" scenario, but at the level of latent explanatory variables). Clearly, a factorized or unimodal representation would be inadequate (because these variables are not at all independent, given x) while the number of modes could grow exponentially with the number of ambiguous factors present in the scene. For example, consider x being the audio of speech pronounced in a foreign language that you do not master well, so that you cannot really segment and parse well each of the words. The number of plausible interpretations (given your poor knowledge of that foreign language) could be exponentially large (in the length of the utterance), and the individual factors (words) would certainly not be conditionally independent (actually having a very rich structure which corresponds to a language model). Even ignoring segmentation makes this a very difficult problem: say there are 10 word segments, each associated with 100 different plausible candidates (out of a million, counting proper nouns), but, due to the language model, only 1 out of 1000 of their combinations being plausible (i.e., the posterior does not factorize or fit a tree structure). So one really has to consider $\frac{1}{1000} \times 100^{10} = 10^{17}$ *plausible configurations* of the latent variables. If one has to take a decision y based on x, e.g., $P(y \mid x) = \sum_h P(y \mid h)P(h \mid x)$ involves summing over a huge number of non-negligible terms of the posterior $P(h \mid x)$, which we can consider as important modes. One way or another, *summing explicitly over that many modes seems implausible*, and assuming single mode (MAP) or a factorized distribution (mean-field) would yield very poor results. Under some assumptions on the underlying data-generating process, it might well be possible to do inference that is exact or a provably good approximation, and searching for graphical models with these properties is an interesting avenue to deal with this problem. Basically, these assumptions work because we assume a specific structure in the form of the underlying distribution. Also, if we are lucky, a few Monte-Carlo samples from $P(h \mid x)$ might suffice to obtain an acceptable approximation for our y, because somehow, as far as y is concerned, many probable values of h yield the same answer y and a Monte-Carlo sample will well represent these different "types" of values of h. That is one form of regularity that could be exploited (if it exists) to approximately solve that problem. What if these assumptions are not appropriate to solve challenging AI problems? Another, more general assumption (and thus one more likely to be appropriate for these problems) is similar to what we usually do with machine learning: function approximation, i.e., although the space of functions is combinatorially large, we are able to generalize by postulating a rather large and flexible family of functions (such as a deep neural net). Thus an interesting avenue is to assume that there exists a computationally tractable function that can compute $P(y \mid x)$ in spite of the apparent complexity of going through the intermediate steps involving h, and that we may learn $P(y \mid x)$ through (x, y) examples. In fact, this is exactly what we do when we train a deep supervised neural net or any black-box supervised machine learning algorithm.

The question we want to address here is that of unsupervised learning: could we take advantage of this idea of bypassing explicit latent variables in the realm of unsupervised probabilistic models of

This deterministic computation is iterated like in a recurrent network until convergence is approached, to obtain a marginal (factorized probability) approximation over all the units.

[3] this can be relaxed by considering tree-structured conditional dependencies (Saul and Jordan, 1996) and mixtures thereof

the data? The approach proposed here has this property. It avoids the strong assumptions on the latent variable structure but still has the potential of capturing very rich distributions, by having only "function approximation" and no approximate inference. Although it avoids explicit latent variables, it still retains the property of exploiting sampling in the computations associated with the model in order to answer questions about the variables of interest.

3 Denoising Auto-Encoders as Generative Models

Alain and Bengio (2013) showed that denoising auto-encoders with small Gaussian corruption and squared error loss estimated the score (derivative of the log-density with respect to the input) of continuous observed random variables. More recently, Bengio *et al.* (2013b) generalized this to arbitrary variables (discrete, continuous or both), arbitrary corruption (not necessarily asymptotically small), and arbitrary loss function (so long as can be seen as a log-likelihood). We first summarize these results below.

Let $\mathcal{P}(X)$ be the unknown data generating distribution. Let $\mathcal{C}(\tilde{X}|X)$ be a corruption process that stochastically transforms an X (such as sampled from $\mathcal{P}$) into a random variable $\tilde{X}$. Let $P_\theta(X|\tilde{X})$ be a denoising auto-encoder that assigns a probability to X, given $\tilde{X}$, when $\tilde{X} \sim \mathcal{C}(\tilde{X}|X)$. When n training examples are provided we obtain an estimator parametrized by θ_n. This estimator defines a Markov chain T_n obtained by sampling alternatively an $\tilde{X}$ from $\mathcal{C}(\tilde{X}|X)$ and an X from $P_\theta(X|\tilde{X})$. Let π_n be the asymptotic distribution of the chain defined by T_n, if it exists. The following theorem is proven by Bengio *et al.* (2013b).

Theorem 1. *If $P_{\theta_n}(X|\tilde{X})$ is a consistent estimator of the true conditional distribution $\mathcal{P}(X|\tilde{X})$ and T_n defines an irreducible and ergodic Markov chain, then as $n \rightarrow \infty$, the asymptotic distribution $\pi_n(X)$ of the generated samples converges to the data generating distribution $\mathcal{P}(X)$.*

It is accompanied with the following corollary, which defines some sufficient conditions for convergence of the chain, and hence applicability of Theorem 1.

Corollary 1. If *$P_\theta(X|\tilde{X})$ is a consistent estimator of the true conditional distribution $\mathcal{P}(X|\tilde{X})$,* **and** *both the data generating distribution and denoising model are contained in and non-zero in a finite-volume region V (i.e., $\forall\tilde{X},\ \forall X \notin V,\ \ \mathcal{P}(X) = 0, P_\theta(X|\tilde{X}) = 0$),* **and** *$\forall\tilde{X}$, $\forall X \in V,\ \ \mathcal{P}(X) > 0, P_\theta(X|\tilde{X}) > 0, \mathcal{C}(\tilde{X}|X) > 0$* **and** *these statements remain true in the limit of $n \rightarrow \infty$,* **then** *the asymptotic distribution $\pi_n(X)$ of the generated samples converges to the data generating distribution $\mathcal{P}(X)$.*

4 Reconstruction with Noise Injected in the Reconstruction Function: Consistent Estimation of the Underlying Data Generating Distribution

In the context where Theorem 1 was proven (Bengio *et al.*, 2013b), $\tilde{X}$ is a noisy or corrupted version of X, i.e., it lives in the same space as X, but nothing in the proof requires that. We exploit this observation to obtain a generalization in which we consider a source of noise Z independent from X, an arbitrary differentiable function $f_{\theta_1}(X, Z)$ from which X cannot be recovered exactly, and a reconstruction distribution $P_{\theta_2}(X|f_{\theta_1}(X, Z))$ which is trained to predict X given $f_{\theta_1}(X, Z)$.

Note that a special case that returns to the situation studied in Bengio *et al.* (2013b) is when $f(X, Z)$ is a fixed parameter-less corruption function that combines the noise Z with X to obtain a corrupted sample $\tilde{X}$, i.e. $\tilde{X} \sim \mathcal{C}(\tilde{X}|X)$. Equivalently, $\tilde{X}$ can be sampled by first sampling $Z \sim \mathcal{P}(Z)$ from a noise distribution and then applying the deterministic function $f(X, Z)$ to obtain $\tilde{X}$. Note that in practice, this is how random variates are generally sampled.

This view gives rise to the following corollary which is central to this paper.

Corollary 2. *Let training data $X \sim \mathcal{P}(X)$ and independent noise $Z \sim \mathcal{P}(Z)$. Consider a model $P_{\theta_2}(X|f_{\theta_1}(X, Z))$ trained (over both θ_1 and θ_2) by regularized conditional maximum likelihood with n examples of (X, Z) pairs. For a given θ_1, a random variable $\tilde{X} = f_{\theta_1}(X, Z)$ is defined. Assume that as n increases, P_{θ_2} is a consistent estimator of the true $\mathcal{P}(X|\tilde{X})$. Assume also that the Markov chain $X_t \sim P_{\theta_2}(X|f_{\theta_1}(X_{t-1}, Z_{t-1}))$ (where $Z_{t-1} \sim \mathcal{P}(Z)$) converges to a distribution π_n, even in the limit as $n \rightarrow \infty$. Then $\pi_n(X) \rightarrow \mathcal{P}(X)$ as $n \rightarrow \infty$.*

Proof. Consider that for a fixed n, θ_1 and θ_2 have been estimated such that, as assumed, as $n \to \infty$, θ_2 gives rise to a consistent estimator of $\mathcal{P}(X|\tilde{X})$. Note how the above Markov chain is equivalent to the Markov chain in Theorem 1 when we define $\tilde{X} = f_{\theta_1}(X, Z)$. We have the two conditions of Theorem 1 (consistent estimator of the conditional $\mathcal{P}(X|\tilde{X})$ and convergence of the Markov chain), so we can conclude that π_n converges to $\mathcal{P}(X)$. □

Since we are now considering the case where f is not a fixed function but a learned one, we have to be careful about defining it in such a way as to guarantee convergence of the Markov chain from which one would sample according to the estimated distribution. Corollary 1 provides some guidance for this purpose. In particular, in order to make sure that $\mathcal{C}(\tilde{X}|X) > 0$ for a given n and asymptotically, it would be necessary that f_θ be constructed such that the conditional entropy $H(f_\theta(X, Z)|X) > 0$, both for any fixed n and asymptotically. It means that the learning procedure should not have the freedom to choos parameters so as to simply eliminate the uncertainty injected with Z. Otherwise, the reconstruction distribution would simply converge to a dirac at the input X. This is the analogue of the constraint on auto-encoders that is needed to prevent them from learning the identity function. Here, we must design the family of reconstruction functions (which produces a distribution over X, given Z and X) such that when the noise Z is injected, there are always several possible values of X that could have been the correct original input.

Another extreme case to think about is when $f(X, Z)$ is overwhelmed by the noise and lost all information about X. In that case the theorems are still valid while giving uninteresting results: the learner must capture the full distribution of X in $P_{\theta_2}(X|\tilde{X})$ because the latter is now equivalent to $P_{\theta_2}(X)$, since $\tilde{X} = f(X, Z)$ does not contain any information about X. What this illustrates, though, is that when the noise is large, the reconstruction distribution (parametrized by θ_2) will need to have the expressive power to represent multiple modes. Otherwise, the reconstruction will tend to capture some kind of average output, which would visually look like a fuzzy combination of the actual modes. In the experiments performed here, we have only considered unimodal reconstruction distributions (with factorized outputs), but future work should investigate multimodal alternatives.

A related element to keep in mind is that one should pick the family of conditional distributions $P_{\theta_2}(X|\tilde{X})$ so that one can sample from them and one can easily train them when given $(X, \tilde{X})$ pairs, e.g., by maximum likelihood.

5 Dealing with Missing Inputs or Structured Output

In general, a simple way to deal with missing inputs is to clamp the observed inputs and then apply the Markov chain with the constraint that the observed inputs are fixed and not resampled at each time step, whereas the unobserved inputs are resampled each time. More precisely, the output (reconstruction) distribution of the model must allow us to sample a subset of variables in the vector X conditionally on the value of the rest. One readily prove that this procedure gives rise to sampling from the appropriate conditional distribution.

Proposition 1. *If a subset $x^{(s)}$ of the elements of X is kept fixed (not resampled) while the remainder $X^{(-s)}$ is updated stochastically during the Markov chain of corollary 2, but using $P(X_{t+1}|f(X_t, Z_t), X_{t+1}^{(s)} = x^{(s)})$, then the asymptotic distribution π_n produces samples of $X^{(-s)}$ from the conditional distribution $\pi_n(X^{(-s)}|X^{(s)} = x^{(s)})$.*

Proof. Without constraint, we know that at convergence of the chain, $P(X_t|f(X_{t-1}, Z_{t-1}))$ produces a sample of π_n. A subset of these samples satisfies the condition $X = x^{(s)}$, and these constrained samples could equally have been produced by sampling from $P(X_t|f(X_{t-1}, Z_{t-1}), X_{t+1}^{(s)} = X^{(s)})$, by definition of conditional distribution. Therefore, at convergence of the chain, we have that $P(X_t|f(X_{t-1}, Z_{t-1}), X_{t+1}^{(s)} = x^{(s)})$ produces a sample from π_n under the condition $X^{(s)} = x^{(s)}$. □

Practically, it means that we must choose a reconstruction distribution from which it is not only easy to sample from, but also from which it is easy to sample conditioned on any subset of the values being known. In the experiments below, we have used a factorial distribution for the reconstruction, meaning that it is trivial to sample conditionally a subset of the input variables.

This method of dealing with missing inputs can be immediately applied to dealing with structured outputs. If $X^{(s)}$ is viewed as an "input" and $X^{(-s)}$ as an "output", then sampling from the chain $X_{t+1}^{(-s)} \sim P(X^{(-s)}|f((X^{(s)}, X_t^{(-s)}), Z_t), X^{(s)})$ will converge to estimators of $\mathcal{P}(X^{(-s)}|X^{(s)})$. This still requires good choices of the parametrization (for f as well as for the conditional probability P), but the advantages of this approach are that there is no approximate inference of latent variables and the learner is trained with respect to simpler conditional probabilities: in the limit of small noise, we conjecture that these conditional probabilities can be well approximated by unimodal distributions. One piece of theoretical evidence comes from Alain and Bengio (2013): *when the amount of corruption noise converges to 0 and the input variables have a smooth continuous density, then a unimodal Gaussian reconstruction density suffices to fully capture the joint distribution.*

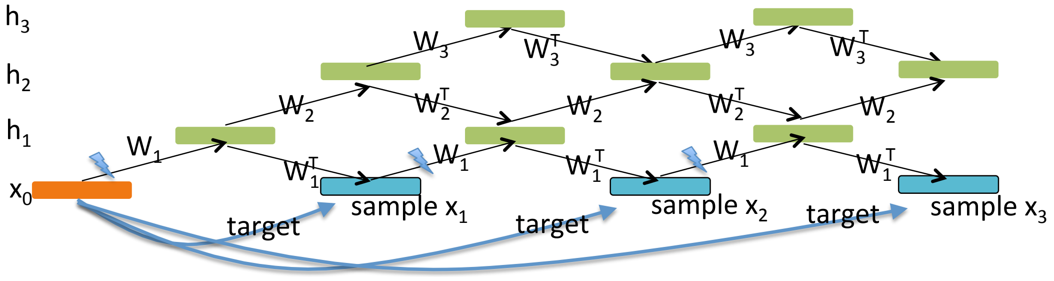

Figure 1: Unfolded computational graph inspired by the Deep Boltzmann Machine inference or sampling, but with backprop-able stochastic units at each layer. The training example $X = x_0$ starts the chain. Either odd or even layers are stochastically updated at each step. Original or sampled x_t's are corrupted by salt-and-pepper noise before entering the graph (lightning symbol). Each x_t for $t > 0$ is obtained by sampling from the reconstruction distribution for this step, and the log-likelihood of target X under that distribution is also computed and used as part of the training objective.

6 Deep Generative Stochastic Networks Trainable by Backprop

The theoretical results on Generative Stochastic Networks (GSNs) in this paper open a large class of possible parametrizations which will share the property that they can capture the underlying data distribution through the above Markov chain. What parametrizations will work well? Where and how to inject noise? We present here the results of preliminary experiments with specific choices for these, but the reader should keep in mind that the space of possibilities is vast.

As a conservative starting point, we propose to explore families of parametrizations which are similar to existing deep stochastic architectures such as the Deep Boltzmann Machine (DBM) (Salakhutdinov and Hinton, 2009) and the Deep Belief Network (DBN) (Hinton *et al.*, 2006). Basically, the idea is to construct a computational graph that is similar to the computational graph for Gibbs sampling or variational inference in Deep Boltzmann Machines. However, we have to diverge a bit from these architectures in order to accommodate the desirable property that it will be possible to back-propagate the gradient of reconstruction log-likelihood with respect to the parameters θ_1 and θ_2. Since the gradient of a binary stochastic unit is 0 almost everywhere, we have to consider related alternatives. An interesting source of inspiration regarding this question is a recent paper on estimating or propagating gradients through stochastic neurons (Bengio, 2013). Here we consider the following stochastic non-linearities: $h_i = \eta_{\text{out}} + \tanh(\eta_{\text{in}} + a_i)$ where a_i is the linear activation for unit i (an affine transformation applied to the input of the unit, coming from the layer below, the layer above, or both), η_{in} and η_{out} are zero-mean Gaussian noises.

To emulate a sampling procedure similar to Boltzmann machines in which the filled-in missing values can depend on the representations at the top level, the computational graph must allow information to propagate both upwards (from input to higher levels of representation) and backwards (vice-versa), giving rise to the computational graph structured illustrated in Figure 1, which is similar to that explored for *deterministic* recurrent auto-encoders (Seung, 1998; Behnke, 2001; Savard, 2011). Downward weight matrices have been fixed to the transpose of corresponding upward weight matrices.

The *walkback* algorithm was proposed in Bengio *et al.* (2013b) to make training of generalized denoising auto-encoders (a special case of the models studied here) more efficient. The basic idea is that the reconstruction is actually obtained after several steps of the sampling Markov chain. In the context presented here, it simply means that the computational graph from X to a reconstruction probability actually involves generating intermediate samples as if we were running the Markov chain starting at X. In the experiments, the graph was unfolded so that $2D$ sampled reconstructions would be produced, where D is the depth (number of hidden layers). The training loss is the sum of the reconstruction negative log-likelihoods (of target X) over all those reconstruction steps.

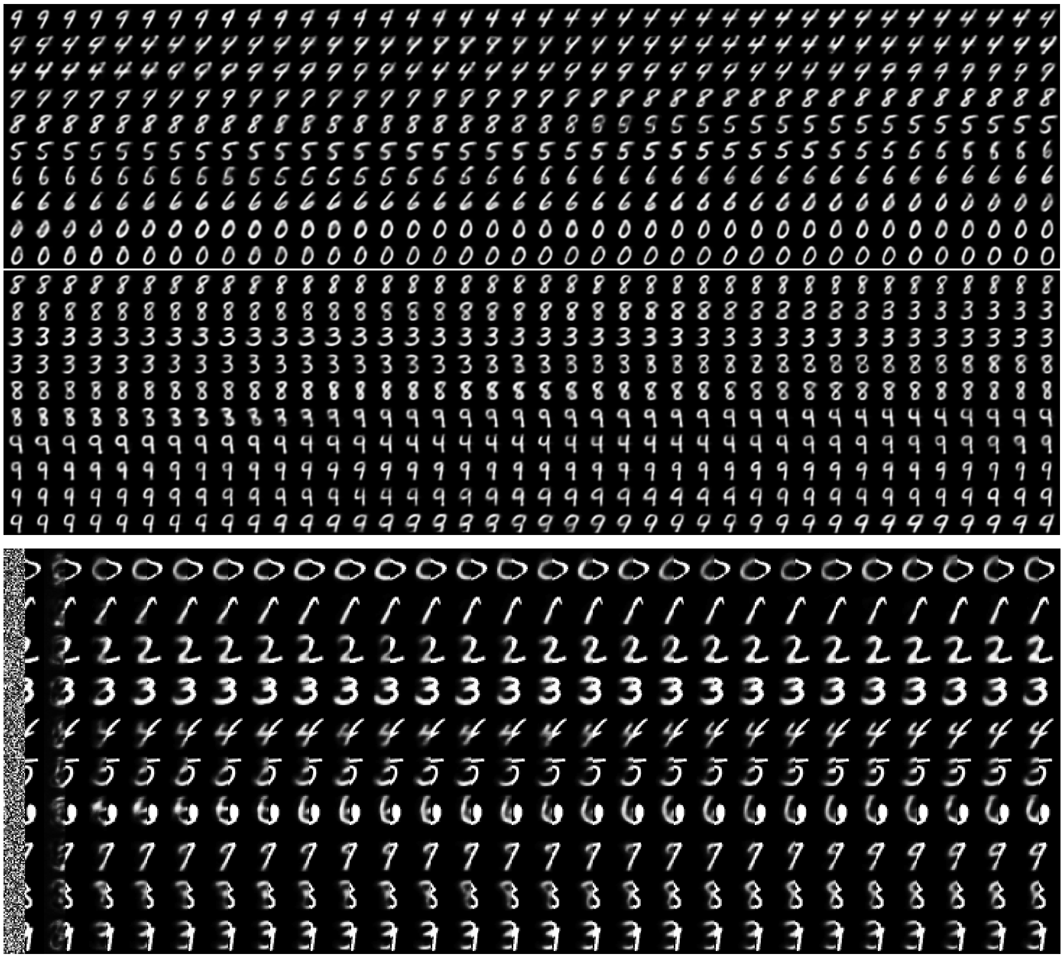

Figure 2: Top: two runs of consecutive samples (one row after the other) generated from a 2-layer GSN model, showing that it mixes well between classes and produces nice and sharp images. Bottom: conditional Markov chain, with the right half of the image clamped to one of the MNIST digit images and the left half successively resampled, illustrating the power of the trained generative model to stochastically fill-in missing inputs.

7 Experimental Validation of GSNs

Experiments evaluating the ability of the GSN models to generate good samples were performed on the MNIST and TFD datasets, following the setup in Bengio *et al.* (2013a). Networks with 2

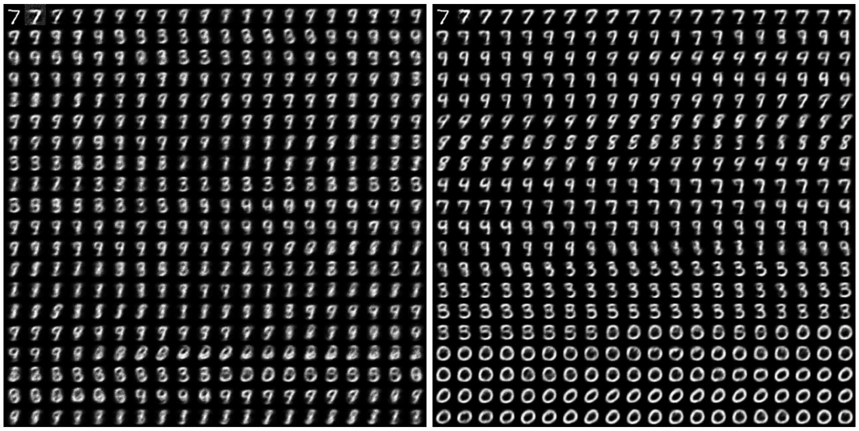

Figure 3: Left: consecutive GSN samples obtained after 10 training epochs. Right: GSN samples obtained after 25 training epochs. This shows quick convergence to a model that samples well. The samples in Figure 2 are obtained after 600 training epochs.

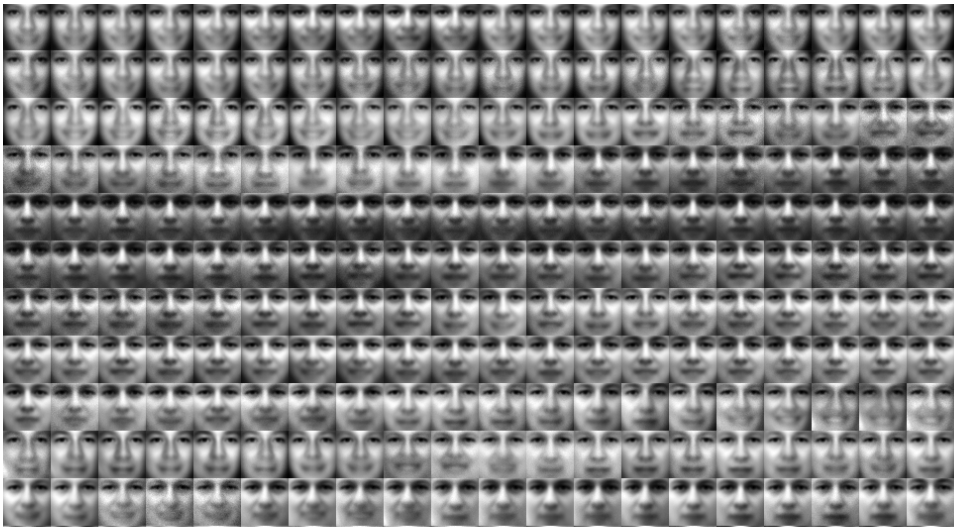

Figure 4: Consecutive GSN samples from a 3-layer model trained on the TFD dataset.

and 3 hidden layers were evaluated and compared to regular denoising auto-encoders (just 1 hidden layer, i.e., the computational graph separates into separate ones for each reconstruction step in the walkback algorithm). They all have tanh hidden units and pre- and post-activation Gaussian noise of standard deviation 2, applied to all hidden layers except the first. In addition, at each step in the chain, the input (or the resampled X_t) is corrupted with salt-and-pepper noise of 40% (i.e., 40% of the pixels are corrupted, and replaced with a 0 or a 1 with probability 0.5). Training is over 100 to 600 epochs at most, with good results obtained after around 100 epochs. Hidden layer sizes vary between 1000 and 1500 depending on the experiments, and a learning rate of 0.25 and momentum of 0.5 were selected to approximately minimize the reconstruction negative log-likelihood. The learning rate is reduced multiplicatively by 0.99 after each epoch. Following Breuleux *et al.* (2011), the quality of the samples was also estimated quantitatively by measuring the log-likelihood of the test set under a Parzen density estimator constructed from 10000 consecutively generated samples (using the real-valued mean-field reconstructions as the training data for the Parzen density estimator). Results are summarized in Table 1. The test set Parzen log-likelihood was not used to select among model architectures, but visual inspection of samples generated did guide the preliminary search reported here. Optimization hyper-parameters (learning rate, momentum, and learning rate reduction schedule) were selected based on the reconstruction log-likelihood training objective. The Parzen log-likelihood obtained with a two-layer model on MNIST is 214 (± standard error of 1.1), while the log-likelihood obtained by a single-layer model (regular denoising auto-encoder, DAE in the table) is substantially worse, at -152±2.2. In comparison, Bengio *et al.* (2013a) report a log-likelihood of -244±54 for RBMs and 138±2 for a 2-hidden layer DBN, using the same setup. We have also evaluated a 3-hidden layer DBM (Salakhutdinov and Hinton, 2009), using the weights provided by the author, and obtained a log-likelihood of 32±2. See `http://www.mit.edu/~rsalakhu/DBM.html` for details. Interestingly, the GSN and the DBN-2 actually perform slightly better than when using samples directly coming from the MNIST training set, maybe becaue they generate more "prototypical" samples (we are using mean-field outputs).

Table 1: Test set log-likelihood obtained by a Parzen density estimator trained on 10000 generated samples, for different generative models trained on MNIST. A DBN-2 has 2 hidden layers and an DBM-3 has 3 hidden layers. The DAE is basically a GSN-1, with no injection of noise inside the network. The last column uses 10000 MNIST training examples to train the Parzen density estimator.

	GSN-2	DAE	RBM	DBM-3	DBN-2	MNIST
LOG-LIKELIHOOD	214	-152	-244	32	138	24
STANDARD ERROR	1.1	2.2	54	1.9	2.0	0.23

Figure 2 shows a single run of consecutive samples from this trained model, illustrating that it mixes quite well (better than RBMs) and produces rather sharp digit images. The figure shows that it can also stochastically complete missing values: the left half of the image was initialized to random pixels and the right side was clamped to an MNIST image. The Markov chain explores plausible variations of the completion according to the trained conditional distribution.

A smaller set of experiments was also run on TFD, yielding a test set Parzen log-likelihood of 1890 ±29. The setup is exactly the same and was not tuned after the MNIST experiments. A DBN-2 yields a Parzen log-likelihood of 1908 ±66, which is undistinguishable statistically, while an RBM yields 604 ± 15. Consecutive samples from the GSN-3 model are shown in Figure 4. Figure 3 shows consecutive samples obtained early on during training, after only 5 and 25 epochs respectively, illustrating the fast convergence of the training procedure.

8 Conclusion

We have introduced a new approach to training generative models that avoid the potential pitfalls of intractable or approximate inference and sampling in models with many latent variables. We argue that if the true posterior distribution of a latent variable model is highly multimodal, then the current methods for training and using such models could yield very poor results. Motivated by this possibility and the recent success of training deep but supervised neural networks, a new framework for training generative models is introduced, called Generative Stochastic Networks (GSNs). The proposed theoretical results state that if noise is injected in the networks that prevents perfect reconstruction, training them to reconstruct their observations suffices to capture the data generating distribution through a simple Markov chain. One of the theoretical assumptions that may not be guaranteed here (unless the amount of noise is very small, which may hurt mixing) is the ability to capture multimodal reconstruction distributions, and this should be the subject of future investigations. This would show up as more fuzzy reconstructions corresponding to averaging several modes. Although it did not seem to clearly hurt in the case of the reported experiments, it might become important for more complex data sets. The theoretical and empirical contributions also include a demonstration of the ability of GSNs to handle and stochastically reconstruct missing inputs, which could be applied to construct structured output models. The experiments clearly validate the theoretical results: it is possible and simple to train a GSN and sample from it, conditionally or unconditionally, while obtaining samples of comparable quality to those obtained with currently established generative models such as RBMs, DBNs and DBMs. Future work on larger scale problems should investigate whether GSNs are less sensitive to the potentially huge number of significant modes in the space of causes and whether this would actually hurt generative models based on anonymous latent variables.

Acknowledgments

The authors would like to acknowledge the stimulating discussions and help from Vincent Dumoulin, Guillaume Alain, Pascal Vincent, Yao Li, Aaron Courville, and Ian Goodfellow, as well as funding from NSERC, CIFAR (YB is a CIFAR Fellow), and the Canada Research Chairs.

References

Alain, G. and Bengio, Y. (2013). What regularized auto-encoders learn from the data generating distribution. In *International Conference on Learning Representations (ICLR'2013)*.

Behnke, S. (2001). Learning iterative image reconstruction in the neural abstraction pyramid. *Int. J. Computational Intelligence and Applications*, **1**(4), 427–438.

Bengio, Y. (2009). *Learning deep architectures for AI*. Now Publishers.

Bengio, Y. (2013). Estimating or propagating gradients through stochastic neurons. Technical Report arXiv:1305.2982, Universite de Montreal.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *NIPS'2006*.

Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. (2013a). Better mixing via deep representations. In *ICML'2013*.

Bengio, Y., Li, Y., Alain, G., and Vincent, P. (2013b). Generalized denoising auto-encoders as generative models. Technical Report arXiv:1305.6663, Universite de Montreal.

Bengio, Y., Courville, A., and Vincent, P. (2013c). Unsupervised feature learning and deep learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*.

Bordes, A., Glorot, X., Weston, J., and Bengio, Y. (2013). A semantic matching energy function for learning with multi-relational data. *Machine Learning: Special Issue on Learning Semantics*.

Breuleux, O., Bengio, Y., and Vincent, P. (2011). Quickly generating representative samples from an RBM-derived process. *Neural Computation*, **23**(8), 2053–2073.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML'2008*.

Dahl, G. E., Ranzato, M., Mohamed, A., and Hinton, G. E. (2010). Phone recognition with the mean-covariance restricted Boltzmann machine. In *NIPS'2010*.

Deng, L., Seltzer, M., Yu, D., Acero, A., Mohamed, A., and Hinton, G. (2010). Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech 2010*, Makuhari, Chiba, Japan.

Gutmann, M. and Hyvarinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS'2010*.

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

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. Technical report, arXiv:1207.0580.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *NIPS'2012*.

Kulesza, A. and Pereira, F. (2008). Structured learning with approximate inference. In *NIPS'2007*.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M.-A., and Huang, F.-J. (2006). A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Scholkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*, pages 191–246. MIT Press.

Lee, H., Battle, A., Raina, R., and Ng, A. (2007). Efficient sparse coding algorithms. In *NIPS'06*, pages 801–808. MIT Press.

Murphy, K. P. (2012). *Machine Learning: a Probabilistic Perspective*. MIT Press, Cambridge, MA, USA.

Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. In *NIPS'2006*.

Salakhutdinov, R. and Hinton, G. E. (2009). Deep Boltzmann machines. In *AISTATS'2009*, pages 448–455.

Saul, L. K. and Jordan, M. I. (1996). Exploiting tractable substructures in intractable networks. In *NIPS'95*. MIT Press, Cambridge, MA.

Savard, F. (2011). *Réseaux de neurones à relaxation entraînés par critère d'autoencodeur débruitant*. Master's thesis, U. Montréal.

Seide, F., Li, G., and Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *Interspeech 2011*, pages 437–440.

Seung, S. H. (1998). Learning continuous attractors in recurrent networks. In *NIPS'97*, pages 654–660. MIT Press.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *ICML 2008*.