arXiv:1306.1091v3 [cs.LG] 25 Oct 2013

Deep Generative Stochastic Networks Trainable by Backprop

Yoshua Bengio*, Éric Thibodeau-Laufer* and Jason Yosinski†
* U. Montreal and † Cornell University

October 4th, 2013

Abstract

We introduce a novel training principle for probabilistic models that is an alternative to maximum likelihood. The proposed Generative Stochastic Networks (GSN) framework is based on learning the transition operator of a Markov chain whose stationary distribution estimates the data distribution. Because the transition distribution is a conditional distribution generally involving a small move, it has fewer dominant modes, being unimodal in the limit of small moves. Thus, it is easier to learn, more like learning to perform supervised function approximation, with gradients that can be obtained by backprop. The theorems provided here generalize recent work on the probabilistic interpretation of denoising autoencoders and provide an interesting justification for dependency networks and generalized pseudolikelihood (along with defining an appropriate joint distribution and sampling mechanism, even when the conditionals are not consistent). GSNs can be used with missing inputs and can be used to sample subsets of variables given the rest. Successful experiments are conducted, validating these theoretical results, on two image datasets and with a particular architecture that mimics the Deep Boltzmann Machine Gibbs sampler but allows training to proceed with backprop, without the need for layerwise pretraining.

1 Introduction

Research in deep learning (see Bengio (2009) and Bengio et al. (2013a) for reviews) grew from 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). 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). In all of these cases, the availability of large quantities of labeled data was critical.

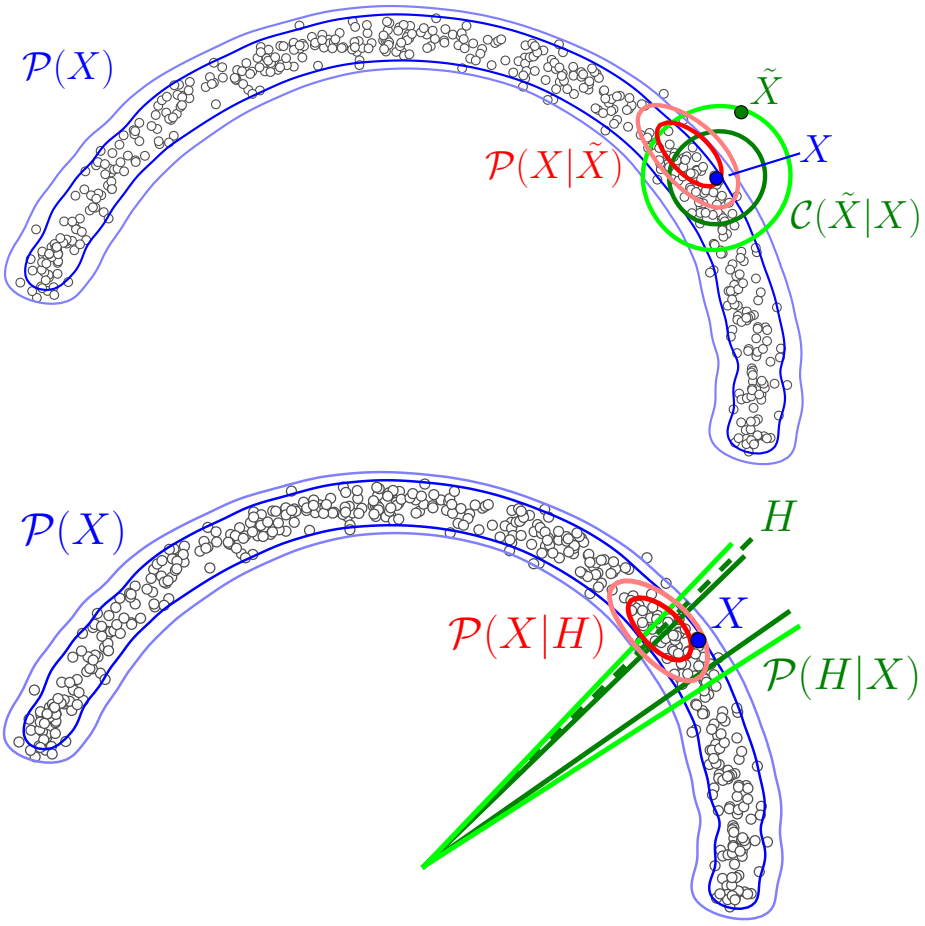

Figure 1: *Top:* A denoising auto-encoder defines an estimated Markov chain where the transition operator first samples a corrupted $\tilde{X}$ from $\mathcal{C}(\tilde{X}|X)$ and then samples a reconstruction from $P_\theta(X|\tilde{X})$, which is trained to estimate an estimated $P_\theta(X|\tilde{X})$ or the ground truth $\mathcal{P}(X|\tilde{X})$. Note how $\mathcal{P}(X|\tilde{X})$ is a much simpler (roughly unimodal) distribution than the ground truth $\mathcal{P}(X)$ and is thus easier to learn. *Bottom:* A GSN proceeds similarly, but allows the use of arbitrary latent variables H to represent X, with the Markov chain state (and mixing) involving both X and H. Here H is the angle about the origin. The GSN inherits the benefit of a simpler conditional and adds latent variables, which allow far more powerful deep representations where mixing is easier (Bengio et al., 2013b).

On the other hand, progress with deep unsupervised architectures has been slower, with the best current options being the Deep Belief Network (DBN) (Hinton et al., 2006) and the Deep Boltzmann Machine (DBM) (Salakhutdinov & Hinton, 2009). Although single-layer unsupervised learners are fairly well developed and used to pretrain these, jointly training all the layers with respect to a single unsupervised criterion remains a challenge, with a few techniques arising to reduce that dependency (Montavon & Muller, 2012; Goodfellow et al., 2013). In contrast to recent progress toward joint supervised training of models with many layers, joint unsupervised training of deep models remains a difficult task. Another exciting frontier in machine learning is the problem of modeling so-called structured outputs, i.e., modeling a conditional distribution where the output variable is high-dimensional and has a complex multi-modal joint distribution (given the input variable). Here we hypothesize that a good part of the difficulties involved in both cases arises from the traditional probabilistic inference and maximum likelihood training criterion reliance on marginalizing across an intractable number of configurations of random variables (observed, latent, or both). In particular, the normalization constant involved in complex multi-modal probabilistic models is often intractable and this is dealt with using various approximations (discussed below) whose limitations may be an important part of the difficulty for training and using deep unsupervised, semi-supervised or structured output models.

Though the goal of training large unsupervised networks has turned out to be more elusive than its supervised counterpart, the vastly larger available volume of unlabeled data still beckons for efficient methods to model it. Recent progress in training supervised models raises the question: can we take advantage of this progress to improve our ability to train deep, generative, unsupervised, semi-supervised or structured output models? This paper lays theoretical foundations for a move in this direction.

This paper takes advantage of recent work on the generative view of denoising autoencoders Bengio et al. (2013c). We provide 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 predicts the distribution of the denoised input.

Our approach creates a Markov chain whose transition operator consists of alternately sampling from a noisy representation distribution $P(h_{t+1}|X, h_t)$ and a denoising reconstruction distribution $P(X|h_{t+1})$, where h_t are stochastic latent variables. We show that the denoising component needs only to model a conditional distribution $P(X|h)$ for some noisy representation h, which is often a far easier task than modeling the full data distribution $P(X)$ (see Figure 1). We also prove that if the denoising component is a consistent (e.g. maximum likelihood) estimator, the stationary distribution of the resulting chain is a consistent estimator of the data density, $P(X)$.

This approach has several appealing properties:

• The model is general and extends to a wide range of architectures, including any sampling procedure whose computation can be unrolled as a Markov Chain, i.e., architectures that add noise during intermediate computation in order to produce random samples of a desired distribution.

• The Markov Chain resulting from the model can be defined over a state (X, h) that includes latent variables, in contrast to the generative view of denoising autoencoders, whose chain includes only X. Including a latent state affords the dual advantage of

more powerful models for a given number of parameters and better mixing in the chain as we add noise to variables representing higher-level information (as first suggested by the results obtained by Bengio et al. (2013b) and Luo et al. (2013)).
• Most critically, we transform the unsupervised density estimation problem into one which is more similar to supervised function approximation. This enables training by (possibly regularized) maximum likelihood and gradient descent computed via simple back-propagation, avoiding the need to compute intractable partition functions. Depending on the model, this may allow us to draw from any number of recently demonstrated supervised training tricks. For example, one could use a convolutional architecture with max-pooling for parametric parsimony and computational efficiency, Adagrad Duchi et al. (2011) for quick gradient descent, and dropout (Hinton et al., 2012) to prevent co-adaptation of hidden representations.
• The model can easily handle missing inputs, sampling conditioned on some of the inputs, and structured outputs.

In Section 2 we discuss in more detail what we view as basic motivation for studying alternate ways of training unsupervised probabilistic models, and then in Section 3 we present the main proofs and properties of the Generative Stochastic Network (GSN) model.

Finally, in Section 4 we apply this idea to create a deep GSN whose computational graph resembles the one followed by Gibbs sampling in deep Boltzmann machines, but that can be trained efficiently with back-propagated gradients and no need for layerwise pre-training.

2 Summing over too many major modes

The approach presented in this paper is motivated by a difficulty often encountered with probabilistic models, especially those containing anonymous latent variables. They are called anonymous because no a priori semantics are 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.

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 making 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.

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. A similarly difficult sampling task may be required in structured output problems where one wants to sample from $P(y, h|x)$ and both y and h are high-dimensional and have a complex highly multi-modal joint distribution (given x). Deep Boltzmann machines (Salakhutdinov & 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 & Weston, 2008; Gutmann & 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 & Hinton (2009), inference for the positive phase is achieved with a mean-field variational approximation.[1]

2.1 The problem of multi-modality

To evade the problem of highly multimodal joint or posterior distributions, all of the currently known approaches to inference and sampling make very strong explicit or implicit assumptions on the form of 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[2] 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 & Pereira, 2008), because it is often in the inner loop of the learning procedure. And what about the cases 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?

2.2 An illustration of the multi-modal problem

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,

[1] In the mean-field approximation, computation proceeds as 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. This deterministic computation is iterated like in a recurrent network until convergence is approached, to obtain a marginal (factorized) approximation over all units.

[2] This can be relaxed by considering tree-structured conditional dependencies (Saul & Jordan, 1996) and mixtures thereof.

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 know well, so that you cannot easily segment and parse 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). Say there are 10^6 possible segmentations into around 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 10^6 of their combinations are plausible (i.e., the posterior does not factorize or fit a tree structure). So one really has to consider $\frac{1}{10^6} \times 100^{10} \times 10^6 = 10^{20}$ *plausible configurations* of the latent variables (high-probability modes). Making 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. But what if these assumptions are not appropriate to solve challenging AI problems? We may find that another more general assumption will suffice: the effectiveness of function approximation. As is typical in machine learning, we postulate a rather large and flexible family of functions (such as deep neural nets) and then use all manner of tricks to pick a member from that combinatorially large family (i.e. to train the neural net) that both fits observed data and generalizes to unseen data well.

The approach proposed here has this property. It avoids the strong assumptions on the latent variable structure (or the structured outputs) but still has the potential of capturing very rich distributions, by having only "function approximation" and no approximate inference. Although it avoids marginalizing over latent variables during training, 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. Besides the approach discussed here, there may well be other very different ways of evading this problem of marginalization, including approaches such as sum-product networks (Poon & Domingos, 2011), which are based on learning a probability function that has a tractable form by construction and yet is from a flexible enough family of distributions.

3 Generative Stochastic Networks

Assume the problem we face is to construct a model for some unknown data-generating distribution $\mathcal{P}(X)$ given only examples of X drawn from that distribution. In many cases, the unknown distribution $\mathcal{P}(X)$ is complicated, and modeling it directly can be difficult.

A recently proposed approach using denoising autoencoders transforms the difficult task of modeling $\mathcal{P}(X)$ into a supervised learning problem that may be much easier to solve. The basic approach is as follows: given a clean example data point X from $\mathcal{P}(X)$, we obtain a corrupted version $\tilde{X}$ by sampling from some corruption distribution $\mathcal{C}(\tilde{X}|X)$. For example, we might take a clean image, X, and add random white noise to produce $\tilde{X}$. We then use supervised learning methods to train a function to reconstruct, as accurately as possible, any X from the data set given only a noisy version $\tilde{X}$. As shown in Figure 1, the reconstruction distribution $\mathcal{P}(X|\tilde{X})$ may often be much easier to learn that the data distribution $\mathcal{P}(X)$, *because it $\mathcal{P}(X|\tilde{X})$ tends to be dominated by a single or few major modes* (such as the roughly Gaussian shaped density in the figure).

But how does learning the reconstruction distribution help us solve our original problem of modeling $\mathcal{P}(X)$? The two problems are clearly related, because if we knew everything about $\mathcal{P}(X)$, then our knowledge of the $\mathcal{C}(\tilde{X}|X)$ that we chose would allow us to precisely specify the optimal reconstruction function via Bayes rule: $\mathcal{P}(X|\tilde{X}) = \frac{1}{z}\mathcal{C}(\tilde{X}|X)\mathcal{P}(X)$, where z is a normalizing constant that does not depend on X. As one might hope, the relation is also true in the opposite direction; once we pick a method of adding noise, $\mathcal{C}(\tilde{X}|X)$, knowledge of the corresponding reconstruction distribution $\mathcal{P}(X|\tilde{X})$ is sufficient to recover the density of the data $\mathcal{P}(X)$.

This intuition was borne out by proofs in two recent papers. Alain & 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. (2013c) generalized this to arbitrary variables (discrete, continuous or both), arbitrary corruption (not necessarily asymptotically small), and arbitrary loss function (so long as they can be seen as a log-likelihood).

Beyond proving that $\mathcal{P}(X|\tilde{X})$ is sufficient to reconstruct the data density, Bengio et al. (2013c) also demonstrated a method of sampling from a learned, parameterized model of the density, $P_\theta(X)$, by running a Markov chain that alternately adds noise using $\mathcal{C}(\tilde{X}|X)$ and denoises by sampling from the learned $P_\theta(X|\tilde{X})$, which is trained to approximate the true $\mathcal{P}(X|\tilde{X})$. The most important contribution of that paper was demonstrating that if a learned, parameterized reconstruction function $P_\theta(X|\tilde{X})$ converges to the true $\mathcal{P}(X|\tilde{X})$, then under some relatively benign conditions the stationary distribution $\pi(X)$ of the resulting Markov chain will exist and will indeed converge to the data distribution $\mathcal{P}(X)$.

3.1 A slight extension of the generative view of denoising autoencoders

More formally, let $P_{\theta_n}(X|\tilde{X})$ be a denoising auto-encoder that has been trained on n training examples. $P_{\theta_n}(X|\tilde{X})$ assigns a probability to X, given $\tilde{X}$, when $\tilde{X} \sim \mathcal{C}(\tilde{X}|X)$. 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. (2013c).

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 ergodic Markov chain,* **then** *as* $n \to \infty$*, the asymptotic distribution* $\pi_n(X)$ *of the generated samples converges to the data-generating distribution* $\mathcal{P}(X)$*.*

In order for Theorem 1 to apply, the chain must be ergodic. One set of conditions under which this occurs is given in the aforementioned paper. We slightly restate them here:

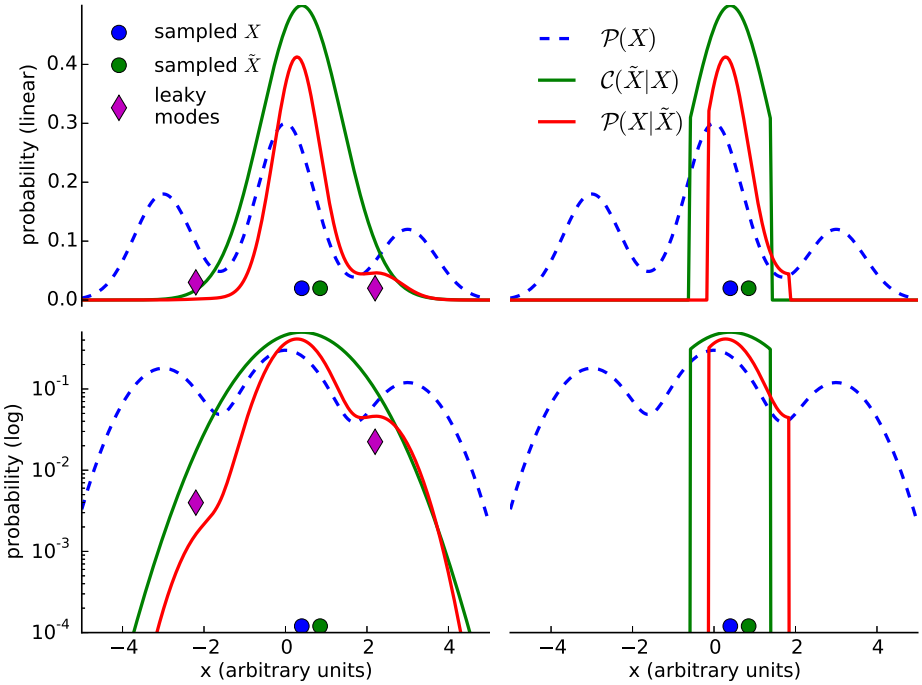

Figure 2: If $\mathcal{C}(\tilde{X}|X)$ is globally supported as required by Corollary 1 (Bengio et al., 2013c), then for $P_{\theta_n}(X|\tilde{X})$ to converge to $\mathcal{P}(X|\tilde{X})$, it will eventually have to model all of the modes in $\mathcal{P}(X)$, even though the modes are damped (see "leaky modes" on the left). However, if we guarantee ergodicity through other means, as in Corollary 2, we allow $P_{\theta_n}(X|\tilde{X})$ to model only the local structure of $\mathcal{P}(X)$ (see right).

Corollary 1. **If** *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 $n \to \infty$, **then** *the chain defined by T_n will be ergodic.*

If conditions in Corollary 1 apply, then the chain will be ergodic and Theorem 1 will apply. However, these conditions are sufficient, not necessary, and in many cases they may be artificially restrictive. In particular, Corollary 1 defines a large region V containing any possible X allowed by the model and requires that we maintain the probability of jumping between any two points in a single move to be greater than 0. While this generous condition helps us easily guarantee the ergodicity of the chain, it also has the unfortunate side effect of requiring that, in order for $P_{\theta_n}(X|\tilde{X})$ to converge to the conditional distribution $\mathcal{P}(X|\tilde{X})$, it must have the capacity to model every mode of $\mathcal{P}(X)$, exactly the difficulty we were trying to avoid. The left two plots in Figure 2 show this difficulty: because $\mathcal{C}(\tilde{X}|X) > 0$ everywhere in V, every mode of $P(X)$ will leak, perhaps attenuated, into $P(X|\tilde{X})$.

Fortunately, we may seek ergodicity through other means. The following corollary allows us to choose a $\mathcal{C}(\tilde{X}|X)$ that only makes small jumps, which in turn only requires $P_\theta(X|\tilde{X})$ to model a small part of the space V around each $\tilde{X}$.

Corollary 2. If *the data-generating distribution is contained in and non-zero in a finite-volume region V (i.e., $\forall X \notin V,\ \mathcal{P}(X) = 0$, and $\forall X \in V,\ \mathcal{P}(X) > 0$)* **and** *all pairs of points in V can be connected by a finite-length path through V* **and** *for some $\epsilon > 0$, $\forall \tilde{X} \in V, \forall X \in V$ within ϵ of each other, $\mathcal{C}(\tilde{X}|X) > 0$ and $P_\theta(X|\tilde{X}) > 0$* **and** *these statements remain true in the limit of $n \to \infty$,* **then** *the chain defined by T_n will be ergodic.*

Proof. Consider any two points X_a and X_b in V. By the assumptions of Corollary 2, there exists a finite length path between X_a and X_b through V. Pick one such finite length path P. Chose a finite series of points $x = \{x_1, x_2, \ldots, x_k\}$ along P, with $x_1 = X_a$ and $x_k = X_b$ such that the distance between every pair of consecutive points (x_i, x_{i+1}) is less than ϵ as defined in Corollary 2. Then the probability of sampling $\tilde{X} = x_{i+1}$ from $\mathcal{C}(\tilde{X}|x_i))$ will be positive, because $\mathcal{C}(\tilde{X}|X)) > 0$ for all $\tilde{X}$ within ϵ of X by the assumptions of Corollary 2. Further, the probability of sampling $X = \tilde{X} = x_{i+1}$ from $P_\theta(X|\tilde{X})$ will be positive from the same assumption on P. Thus the probability of jumping along the path from x_i to x_{i+1}, $T_n(X_{t+1} = x_{i+1}|X_t = x_i)$, will be greater than zero for all jumps on the path. Because there is a positive probability finite length path between all pairs of points in V, all states commute, and the chain is irreducible. If we consider $X_a = X_b \in V$, by the same arguments $T_n(X_t = X_a|X_{t-1} = X_a) > 0$. Because there is a positive probability of remaining in the same state, the chain will be aperiodic. Because the chain is irreducible and over a finite state space, it will be positive recurrent as well. Thus, the chain defined by T_n is ergodic. □

Although this is a weaker condition that has the advantage of making the denoising distribution even easier to model (probably having less modes), we must be careful to choose the ball size ϵ large enough to guarantee that one can jump often enough between the major modes of $\mathcal{P}(X)$ when these are separated by zones of tiny probability. ϵ must be larger than half the largest distance one would have to travel across

a desert of low probability separating two nearby modes (which if not connected in this way would make V not anymore have a single connected component). Practically, there would be a trade-off between the difficulty of estimating $\mathcal{P}(X|\tilde{X})$ and the ease of mixing between major modes separated by a very low density zone.

The generalization of the above results presented in the next section is meant to help deal with this mixing problem. It is inspired by the recent work (Bengio et al., 2013b) showing that mixing between modes can be a serious problem for RBMs and DBNs, and that well-trained deeper models can greatly alleviate it by allowing the mixing to happen at a more abstract level of representation (e.g., where some bits can actually represent which mode / class / manifold is considered).

3.2 Generalizing the denoising autoencoder to GSNs

The denoising auto-encoder Markov chain is defined by $\tilde{X}_t \sim C(\tilde{X}|X_t)$ and $X_{t+1} \sim P_\theta(X|\tilde{X}_t)$, where X_t alone can serve as the state of the chain. Instead, a GSN is associated with a Markov chain with both X_t and a latent variable H_t as state variables, of the form

$$\begin{aligned} H_{t+1} &\sim P_{\theta_1}(H|H_t, X_t) \\ X_{t+1} &\sim P_{\theta_2}(X|H_{t+1}) \end{aligned}$$

where we equivalently define $H_{t+1} = f_{\theta_1}(X_t, Z_t, H_t)$, for some independent noise source Z_t, with the condition that X_t cannot be recovered exactly from H_{t+1}.

Denoising auto-encoders are thus a special case of GSNs, and consistency can be proven in a similar way.

Theorem 2. *Let training data $X \sim \mathcal{P}(X)$ and independent noise $Z \sim \mathcal{P}(Z)$ and introduce a sequence of latent variables H defined iteratively through a function f with $H_t = f_{\theta_1}(X_{t-1}, Z_{t-1}, H_{t-1})$ for a given sequence of X_t's. Consider a model $P_{\theta_2}(X|f_{\theta_1}(X, Z_{t-1}, H_{t-1}))$ trained (over both θ_1 and θ_2) so that $P_{\theta_2}(X|H)$, for a given θ_1, is a consistent estimator of the true $\mathcal{P}(X|H)$. Consider the Markov chain defined above and assume that it converges to a stationary distribution π_n over the X and H and with marginal $\pi_n(X)$, even in the limit as the number of training examples $n \to \infty$. Then $\pi_n(X) \to \mathcal{P}(X)$ as $n \to \infty$.*

Proof. For any fixed value of θ_1 that satisfies the ergodicity of the above "model chain", consider a "ground truth" chain in which $X_t \sim \mathcal{P}(X|H_t = f_{\theta_1}(X_{t-1}, Z_{t-1}, H_{t-1}))$ instead of $X_t \sim P_{\theta_2}(X|H_t = f_{\theta_1}(X_{t-1}, Z_{t-1}, H_{t-1}))$. The stationary distribution of the ground truth chain (which we have assumed exists since we are assuming that there exists θ_2 which gives rise to the ground truth $\mathcal{P}(X|H) = P_{\theta_2}(X|H)$) has $\mathcal{P}(X)$ as marginal distribution over X, since at a fixed point of the chain we must have $\mathcal{P}(X) = \sum_H \mathcal{P}(X|H)\mathcal{P}(H)$ where $\mathcal{P}(H)$ is the marginal over H produced by the stationary distribution of the ground truth chain. Now we only need to show that as $P_{\theta_2}(X|H)$ approaches the ground truth $\mathcal{P}(X|H)$, so do the stationary distributions of the corresponding chains. Denote T_n the transition operator associated with the estimator with n examples and $\mathcal{T}$ the transition operator associated with the ground truth chain. As $P_{\theta_2}(X|H)$ approaches $\mathcal{P}(X|H)$ when n increases, the corresponding operators approach each other, i.e., $T_n \to \mathcal{T}$. Let v be the principal eigenvector of $\mathcal{T}$ (i.e.

the stationary distribution of the ground truth chain). For any matrix M and unit vector v, $||Mv||_2 \leq \sup_{||x||_2=1} ||Mx||_2 = ||M||_2$. Hence $||(\mathcal{T} - T_n)v||_2 \leq ||\mathcal{T} - T_n||_2 \to 0$, which implies that $T_n v \to \mathcal{T} v = v$, where the last equality comes from the Perron-Frobenius theorem (the leading eigenvalue is 1). Therefore the estimated stationary distribution converges to the ground truth stationary distribution, and in particular the associated marginals over X converge in the same way, i.e., $\pi_n(X) \to \mathcal{P}(X)$. □

Since we are now considering the case where f is not a fixed function (as in the denoising autoencoder case) 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. So long as f and θ_1 allow the chain to mix, the theorem remains applicable. It means that the learning procedure should not have the freedom to choose parameters that simply eliminate the uncertainty injected with Z or through the form of f. 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 $f(X, Z, H)$ is overwhelmed by the noise and has lost all information about X. In that case the theorems are still applicable while giving uninteresting results: the learner must capture the full distribution of X in $P_{\theta_2}(X|H)$ because the latter is now equivalent to $P_{\theta_2}(X)$, since $f(X, Z, H)$ no longer contains information about X. This illustrates 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 an average output, which would visually look like a fuzzy combination of actual modes. In the experiments performed here, we have only considered unimodal reconstruction distributions (with factorized outputs), because we expect that even if $\mathcal{P}(X|H)$ is not unimodal, it would be dominated by a single mode. However, future work should investigate multimodal alternatives.

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

3.3 Handling 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. One readily proves 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 Theorem 2, but using $P(X_t|H_t, X_t^{(s)} = x^{(s)})$,* **then** *the asymptotic distribution π_n of the Markov chain 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, the chain produces samples 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 X_t from $P_{\theta_2}(X_t|f_{\theta_1}(X_{t-1}, Z_{t-1}, H_{t-1}), X_t^{(s)} = X^{(s)})$, by definition of conditional distribution. Therefore, at convergence of the chain, we have that using the constrained distribution $P(X_t|f(X_{t-1}, Z_{t-1}, H_{t-1}), X_t^{(s)} = x^{(s)})$ produces a sample from π_n under the condition $X^{(s)} = x^{(s)}$. □

Practically, it means that we must choose an output (reconstruction) distribution from which it is not only easy to sample from, but also from which it is easy to sample a subset of the variables in the vector X conditioned on the rest being known. In the experiments below, we used a factorial distribution for the reconstruction, from which it is trivial to sample conditionally a subset of the input variables.

This method of dealing with missing inputs can be immediately applied to structured outputs. If $X^{(s)}$ is viewed as an "input" and $X^{(-s)}$ as an "output", then sampling from $X_{t+1}^{(-s)} \sim P(X^{(-s)}|f((X^{(s)}, X_t^{(-s)}), Z_t, H_t), X^{(s)})$ 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. Theoretical evidence comes from Alain & 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.*

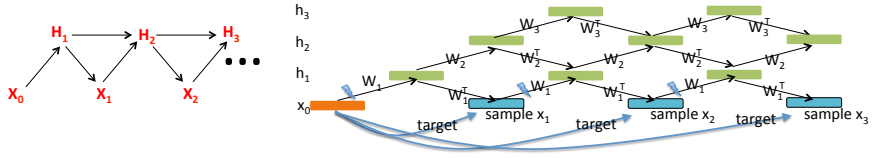

Figure 3: *Left:* Generic GSN Markov chain with state variables X_t and H_t. *Right:* GSN Markov chain inspired by the unfolded computational graph of the Deep Boltzmann Machine Gibbs sampling process, but with backprop-able stochastic units at each layer. The training example $X = x_0$ starts the chain. 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 $X = X_0$ under that distribution is also computed and used as part of the training objective.

3.4 The special case of Dependency Networks

Dependency networks (Heckerman et al., 2000) are models in which one estimates conditionals $P_i(x_i|x_{-i})$, where x_{-i} denotes $x \setminus x_i$, i.e., the set of variables other than

the i-th one, x_i. Note that each P_i may be parametrized separately, thus not guaranteeing that there exists a unique joint of which they are the conditionals. Instead of the ordered pseudo-Gibbs sampler defined in Heckerman et al. (2000), which resamples each variable x_i in the order $x_1, x_2, \ldots$, we can view dependency networks in the GSN framework by defining a proper Markov chain in which at each step one randomly chooses which variable to resample. The corruption process therefore just consists of $H = f(X, Z) = X_{-s}$ where X_{-s} is the complement of X_s, with s a randomly chosen subset of elements of X (possibly constrained to be of size 1). Furthermore, we parametrize the reconstruction distribution as $P_{\theta_2}(X = x|H) = \delta_{x_{-s}=X_{-s}} P_{\theta_2,s}(X_s = x_s|x_{-s})$ where the estimated conditionals $P_{\theta_2,s}(X_s = x_s|x_{-s})$ are not constrained to be consistent conditionals of some joint distribution over all of X.

Proposition 2. *If the above GSN Markov chain has a stationary distribution, then the dependency network defines a joint distribution (which is that stationary distribution), which does not have to be known in closed form. Furthermore, if the conditionals are consistent estimators of the ground truth conditionals, then that stationary distribution is a consistent estimator of the ground truth joint.*

The proposition can be proven by immediate application of Theorem 1 with the above definitions of the GSN. This joint stationary distribution can exist even if the conditionals are not consistent. To show that, assume that some choice of (possibly inconsistent) conditionals gives rise to a stationary distribution π. Now let us consider the set of all conditionals (not necessarily consistent) that could have given rise to that π. Clearly, the conditionals derived from π is part of that set, but there are infinitely many others (a simple counting argument shows that the fixed point equation of π introduces fewer constraints than the number of degrees of freedom that define the conditionals). To better understand why the ordered pseudo-Gibbs chain does not benefit from the same properties, we can consider an extended case (using Theorem 2), where we consider a latent variable H consisting of the index of the next variable to resample. In that case, the Markov chain on X and H would be periodic, thus violating the ergodicity assumption of the theorem. However, by introducing randomness in the choice of which variable(s) to resample next, we obtain aperiodicity and ergodicity, yielding as stationary distribution a mixture over all possible resampling orders. These results also show in a novel way (see e.g. Hyvärinen (2006) for earlier results) that training by pseudolikelihood or generalized pseudolikelihood provides a consistent estimator of the associated joint, so long as the GSN Markov chain defined above is ergodic.

4 Experimental Example of GSN

The theoretical results on Generative Stochastic Networks (GSNs) open for exploration a large class of possible parametrizations which will share the property that they can capture the underlying data distribution through the GSN Markov chain. What parametrizations will work well? Where and how should one inject noise? We present results of preliminary experiments with specific selections for each of these choices, 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 & Hinton, 2009). 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: $h_i = \eta_{\rm out} + \tanh(\eta_{\rm in} + a_i)$ 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) and $\eta_{\rm in}$ and $\eta_{\rm out}$ 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 allows information to propagate both upwards (from input to higher levels) and downwards, giving rise to the computational graph structure illustrated in Figure 3, 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. (2013c) 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 obtained after not one but several steps of the sampling Markov chain. In this context 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.

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. (2013b). Networks with 2 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). This can be seen as an

lower bound on the true log-likelihood, with the bound converging to the true likelihood as we consider more samples and appropriately set the smoothing parameter of the Parzen estimator[3] Results are summarized in Table 1. The test set Parzen log-likelihood bound 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 bound obtained with a two-layer model on MNIST is 214 ($\pm$ standard error of 1.1), while the log-likelihood bound obtained by a single-layer model (regular denoising auto-encoder, DAE in the table) is substantially worse, at -152$\pm$2.2. In comparison, Bengio et al. (2013b) report a log-likelihood bound of -244$\pm$54 for RBMs and 138$\pm$2 for a 2-hidden layer DBN, using the same setup. We have also evaluated a 3-hidden layer DBM (Salakhutdinov & Hinton, 2009), using the weights provided by the author, and obtained a Parzen log-likelihood bound of 32$\pm$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 because they generate more "prototypical" samples (we are using mean-field outputs). Figure 4 shows a single run of consecutive samples from this trained model[4], 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 bound of 1890 $\pm$29. The setup is exactly the same and was not tuned after the MNIST experiments. A DBN-2 yields a Parzen log-likelihood bound of 1908 $\pm$66, which is indistinguishable statistically, while an RBM yields 604 $\pm$ 15. One out of every 2 consecutive samples[5] from the GSN-3 model are shown in Figure 5.

5 Conclusion

We have introduced a new approach to training generative models, called Generative Stochastic Networks (GSN), that is an alternative to maximum likelihood, with the objective of avoiding the intractable marginalizations and the danger of poor approximations of these marginalizations. The training procedure is more similar to function approximation than to unsupervised learning because the reconstruction distribution is simpler than the data distribution, often unimodal (provably so in the limit of very small noise). This makes it possible to train unsupervised models that capture the data-generating distribution simply using back-prop and gradient descent (in a computational graph that includes noise injection). The proposed theoretical results state that under mild conditions (in particular that the noise injected in the networks pre-

[3]However, in this paper, to be consistent with the numbers given in Bengio et al. (2013b) we used a Gaussian Parzen density, which makes the numbers not comparable with the AIS log-likelihood upper bounds for binarized images reported in other papers for the same data.

[4]See supplementary material for longer runs

[5]Longer runs without skips are shown in the supplementary material

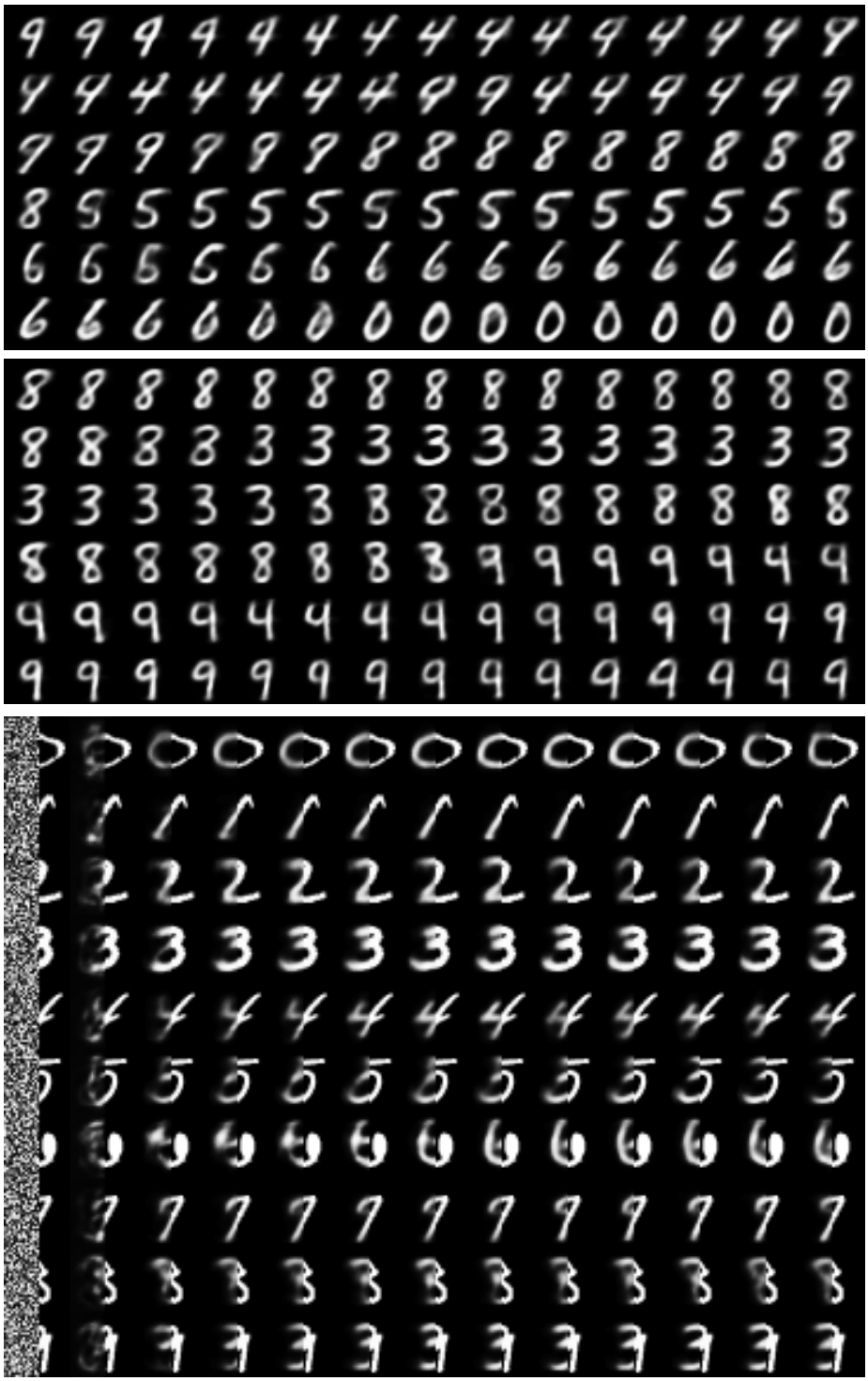

Figure 4: Top: two runs of consecutive samples (one row after the other) generated from 2-layer GSN model, showing fast mixing between classes, nice and sharp images. Note: only every fourth sample is shown; see the supplemental material for the samples in between. Bottom: conditional Markov chain, with 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 generative model to stochastically fill-in missing inputs.

Figure 5: GSN samples from a 3-layer model trained on the TFD dataset. Every second sample is shown; see the supplemental material for every sample. At the end of each row, we show the nearest example from the training set to the last sample on that row, to illustrate that the distribution is not merely copying the training set.

Table 1: Test set log-likelihood lower bound (LL) obtained by a Parzen density estimator constructed using 10000 generated samples, for different generative models trained on MNIST. The LL is not directly comparable to AIS likelihood estimates because we use a Gaussian mixture rather than a Bernouilli mixture to compute the likelihood, but we can compare with Rifai et al. (2012); Bengio et al. (2013b,c) (from which we took the last three columns). A DBN-2 has 2 hidden layers, a CAE-1 has 1 hidden layer, and a CAE-2 has 2. 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 LOWER BOUND	214	-152	-244	32	138	24
STANDARD ERROR	1.1	2.2	54	1.9	2.0	1.6

vents perfect reconstruction), training the model to denoise and reconstruct its observations (through a powerful family of renconstruction distributions) suffices to capture the data-generating distribution through a simple Markov chain. Another way to put it is that we are training the transition operator of a Markov chain whose stationary distribution estimates the data distribution, and it turns out that this is a much easier learning problem because the normalization constant for this conditional distribution is generally dominated by fewer modes. These theoretical results are extended to the case where the corruption is local but still allows the chain to mix and to the case where some inputs are missing or constrained (thus allowing to sample from a conditional distribution on a subset of the observed variables or to learned structured output models). The GSN framework is shown to lend to dependency networks a valid estimator of the joint distribution of the observed variables even when the learned conditionals are not consistent, also allowing to prove consistency of generalized pseudolikelihood training, associated with the stationary distribution of the corresponding GSN (that randomly chooses a subset of variables and then resamples it). Experiments have been conducted to validate the theory, in the case where the GSN architecture emulates the Gibbs sampling process of a Deep Boltzmann Machine, on two datasets. A quantitative evaluation of the samples confirms that the training procedure works very well (in this case allowing us to train a deep generative model without layerwise pretraining) and can be used to perform conditional sampling of a subset of variables given the rest.

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.

A Supplemental Experimental Results

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. (2013c). Networks with 2 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). This can be seen as an *lower bound on the true log-likelihood*, with the bound converging to the true likelihood as we consider more samples and appropriately set the smoothing parameter of the Parzen estimator[6]. Results are summarized in Table 1. The test set Parzen log-likelihood bound 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 bound obtained with a two-layer model on MNIST is 214 ($\pm$ standard error of 1.1), while the log-likelihood bound obtained by a single-layer model (regular denoising auto-encoder, DAE in the table) is substantially worse, at -152$\pm$2.2. In comparison, Bengio et al. (2013c) report a log-likelihood bound of -244$\pm$54 for RBMs and 138$\pm$2 for a 2-hidden layer DBN, using the same setup. We have also evaluated a 3-hidden layer DBM (Salakhutdinov & Hinton, 2009), using the weights provided by the author, and obtained a Parzen log-likelihood bound of 32$\pm$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 because they generate more "prototypical" samples (we are using mean-field outputs). Figure 6 shows two runs 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

[6]However, in this paper, to be consistent with the numbers given in Bengio et al. (2013c) we used a Gaussian Parzen density, which (in addition to being lower rather than upper bounds) makes the numbers not comparable with the AIS log-likelihood upper bounds for binarized images reported in some papers for the same data.

variations of the completion according to the trained conditional distribution.

A smaller set of experiments was also run on TFD, yielding for a GSN a test set Parzen log-likelihood bound of 1890 $\pm$29. The setup is exactly the same and was not tuned after the MNIST experiments. A DBN-2 yields a Parzen log-likelihood bound of 1908 $\pm$66, which is undistinguishable statistically, while an RBM yields 604 $\pm$ 15. A run of consecutive samples from the GSN-3 model are shown in Figure 8. Figure 7 shows consecutive samples obtained early on during training, after only 5 and 25 epochs respectively, illustrating the fast convergence of the training procedure.

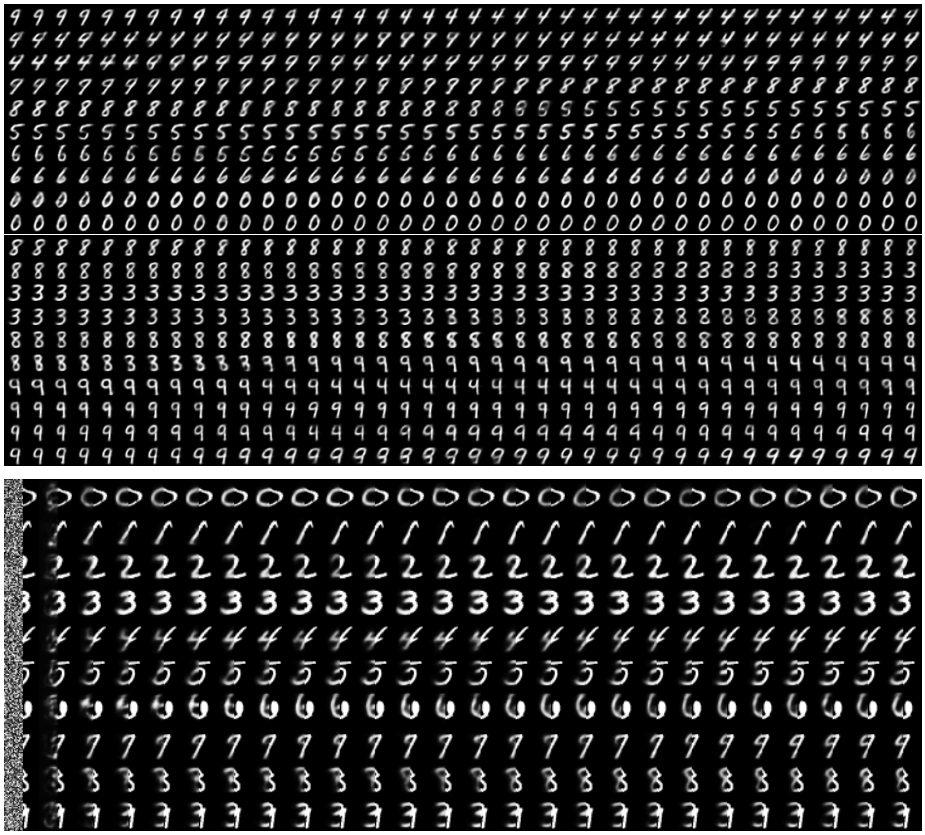

Figure 6: These are expanded plots of those in Figure 4. *Top:* two runs of consecutive samples (one row after the other) generated from a 2-layer GSN model, showing that it mixes well between classes and produces nice and sharp images. Figure 4 contained only one in every four samples, whereas here we show every sample. *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. Figure 4 showed only 13 samples in each chain; here we show 26.

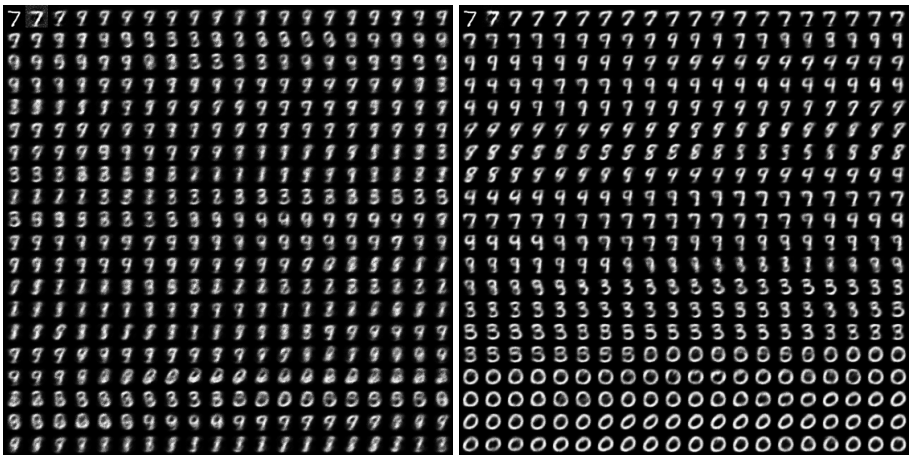

Figure 7: Left: consecutive GSN samples obtained after 10 training epochs. Right: GSN samples obtained after 25 training epochs. This shows quick convergence to a model that samples well. The samples in Figure 6 are obtained after 600 training epochs.

References

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

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

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

Bengio, Yoshua. *Learning deep architectures for AI*. Now Publishers, 2009.

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

Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Unsupervised feature learning and deep learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 2013a.

Bengio, Yoshua, Mesnil, Grégoire, Dauphin, Yann, and Rifai, Salah. Better mixing via deep representations. In *ICML'13*, 2013b.

Bengio, Yoshua, Yao, Li, Alain, Guillaume, and Vincent, Pascal. Generalized denoising auto-encoders as generative models. In *NIPS26*. Nips Foundation, 2013c.

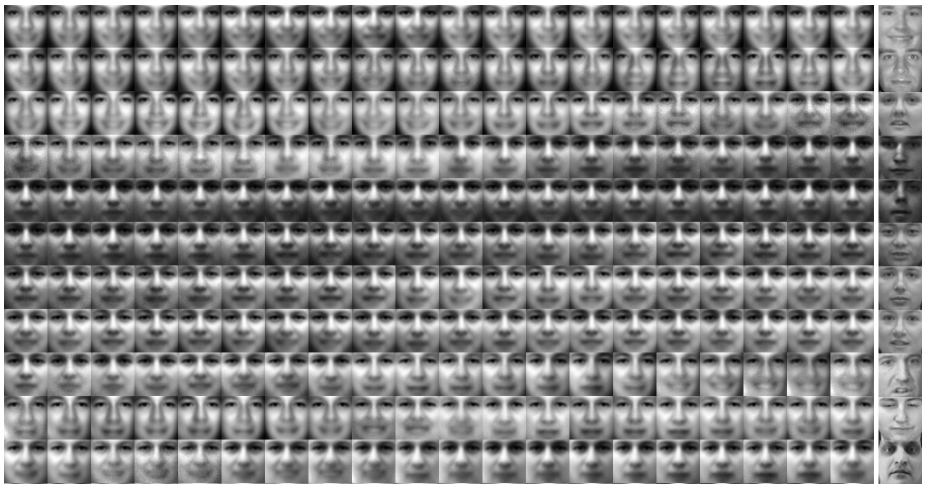

Figure 8: Consecutive GSN samples from a 3-layer model trained on the TFD dataset. At the end of each row, we show the nearest example from the training set to the last sample on that row, to illustrate that the distribution is not merely copying the training set.

Bordes, Antoine, Glorot, Xavier, Weston, Jason, and Bengio, Yoshua. A semantic matching energy function for learning with multi-relational data. *Machine Learning: Special Issue on Learning Semantics*, 2013.

Breuleux, Olivier, Bengio, Yoshua, and Vincent, Pascal. Quickly generating representative samples from an RBM-derived process. *Neural Computation*, 23(8):2053–2073, 2011.

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

Dahl, George E., Ranzato, Marc'Aurelio, Mohamed, Abdel-rahman, and Hinton, Geoffrey E. Phone recognition with the mean-covariance restricted Boltzmann machine. In *NIPS'2010*, 2010.

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

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Goodfellow, Ian J., Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Multi-prediction deep Boltzmann machines. In *NIPS26*. Nips Foundation, 2013.

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

Heckerman, David, Chickering, David Maxwell, Meek, Christopher, Rounthwaite, Robert, and Kadie, Carl. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

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

Hinton, Geoffrey E., Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinv, Ruslan. Improving neural networks by preventing co-adaptation of feature detectors. Technical report, arXiv:1207.0580, 2012.

Hyvärinen, Aapo. Consistency of pseudolikelihood estimation of fully visible boltzmann machines. *Neural Computation*, 2006.

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

Kulesza, Alex and Pereira, Fernando. Structured learning with approximate inference. In *NIPS'2007*, 2008.

LeCun, Yann, Chopra, Sumit, Hadsell, Raia, Ranzato, Marc-Aurelio, and Huang, Fu-Jie. A tutorial on energy-based learning. In Bakir, G., Hofman, T., Scholkopf, B., Smola, A., and Taskar, B. (eds.), *Predicting Structured Data*, pp. 191–246. MIT Press, 2006.

Lee, Honglak, Battle, Alexis, Raina, Rajat, and Ng, Andrew. Efficient sparse coding algorithms. In *NIPS'06*, pp. 801–808. MIT Press, 2007.

Luo, Heng, Carrier, Pierre Luc, Courville, Aaron, and Bengio, Yoshua. Texture modeling with convolutional spike-and-slab RBMs and deep extensions. In *AISTATS'2013*, 2013.

Montavon, Gregoire and Muller, Klaus-Robert. Deep Boltzmann machines and the centering trick. In Montavon, Grégoire, Orr, Genevieve, and Müller, Klaus-Robert (eds.), *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pp. 621–637. 2012.

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

Poon, Hoifung and Domingos, Pedro. Sum-product networks: A new deep architecture. In *UAI'2011*, Barcelona, Spain, 2011.

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

Rifai, Salah, Bengio, Yoshua, Dauphin, Yann, and Vincent, Pascal. A generative process for sampling contractive auto-encoders. In *ICML'12*, 2012.

Salakhutdinov, Ruslan and Hinton, Geoffrey E. Deep Boltzmann machines. In *AISTATS'2009*, pp. 448–455, 2009.

Saul, Lawrence K. and Jordan, Michael I. Exploiting tractable substructures in intractable networks. In *NIPS'95*. MIT Press, Cambridge, MA, 1996.

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

Seide, Frank, Li, Gang, and Yu, Dong. Conversational speech transcription using context-dependent deep neural networks. In *Interspeech 2011*, pp. 437–440, 2011.

Seung, Sebastian H. Learning continuous attractors in recurrent networks. In *NIPS'97*, pp. 654–660. MIT Press, 1998.

Vincent, Pascal, Larochelle, Hugo, Bengio, Yoshua, and Manzagol, Pierre-Antoine. Extracting and composing robust features with denoising autoencoders. In *ICML 2008*, 2008.