# Towards Biologically Plausible Deep Learning

**Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein and Zhouhan Lin**

Montreal Institute for Learning Algorithms, University of Montreal, Montreal, QC, H3C 3J7

## Abstract

Neuroscientists have long criticised deep learning algorithms as incompatible with current knowledge of neurobiology. We explore more biologically plausible versions of deep representation learning, focusing here mostly on unsupervised learning but developing a learning mechanism that could account for supervised, unsupervised and reinforcement learning. The starting point is that the basic learning rule believed to govern synaptic weight updates (Spike-Timing-Dependent Plasticity) can be interpreted as gradient descent on some objective function so long as the neuronal dynamics push firing rates towards better values of the objective function (be it supervised, unsupervised, or reward-driven). The second main idea is that this corresponds to a form of the variational EM algorithm, i.e., with approximate rather than exact posteriors, implemented by neural dynamics. Another contribution of this paper is that the gradients required for updating the hidden states in the above variational interpretation can be estimated using an approximation that only requires propagating activations forward and backward, with pairs of layers learning to form a denoising auto-encoder. Finally, we extend the theory about the probabilistic interpretation of auto-encoders to justify improved sampling schemes based on the generative interpretation of denoising auto-encoders, and we validate all these ideas on generative learning tasks.

## 1. Introduction

Deep learning and artificial neural networks have taken their inspiration from brains, but mostly for the form of the computation performed (with much of the biology, such as the presence of spikes remaining to be accounted for). However, what is lacking currently is a credible *machine learning interpretation of the learning rules* that seem to exist in biological neurons that would explain joint training of a deep neural network, i.e., accounting for credit assign-ment through a long chain of neural connections. Solving the credit assignment problem therefore means identifying neurons and weights that are responsible for a desired outcome and changing parameters accordingly. Whereas back-propagation offers a machine learning answer, it is not biologically plausible, as discussed in the next paragraph. Finding a biologically plausible machine learning approach for credit assignment in deep networks is the main long-term question to which this paper contributes.

Let us first consider the claim that state-of-the-art deep learning algorithms rely on mechanisms that seem biologically implausible, such as gradient back-propagation, i.e., the mechanism for computing the gradient of an objective function with respect to neural activations and parameters. The following difficulties can be raised regarding the biological plausibility of back-propagation: (1) the back-propagation computation (coming down from the output layer to lower hidden layers) is purely linear, whereas biological neurons interleave linear and non-linear operations, (2) if the feedback paths known to exist in the brain (with their own synapses and maybe their own neurons) were used to propagate credit assignment by backprop, they would need precise knowledge of the derivatives of the non-linearities at the operating point used in the corresponding feedforward computation on the feedforward path[1], (3) similarly, these feedback paths would have to use exact symmetric weights (with the same connectivity, transposed) of the feedforward connections,[2] (4) real neurons communicate by (possibly stochastic) binary values (spikes), not by clean continuous values, (5) the computation would have to be precisely clocked to alternate between feedforward and back-propagation phases (since the latter needs the former's results), and (6) it is not clear where the output targets would come from. The approach proposed in this paper has the ambition to address all these issues, although some question marks as to a possible biological implementations remain, and of course many details of the biology that need to be accounted for are not covered

---

[1] and with neurons not all being exactly the same, it could be difficult to match the right estimated derivatives

[2] this is known as the *weight transport* problem (Lillicrap et al., 2014)

here.

Note that back-propagation is used not just for classical supervised learning but also for many unsupervised learning algorithms, including all kinds of auto-encoders: sparse auto-encoders (Ranzato et al., 2007; Goodfellow et al., 2009), denoising auto-encoders (Vincent et al., 2008), contractive auto-encoders (Rifai et al., 2011), and more recently, variational auto-encoders (Kingma & Welling, 2014). Other unsupervised learning algorithms exist which do not rely on back-propagation, such as the various Boltzmann machine learning algorithms (Hinton & Sejnowski, 1986; Smolensky, 1986; Hinton et al., 2006; Salakhutdinov & Hinton, 2009). Boltzmann machines are probably the most biologically plausible learning algorithms for deep architectures that we currently know, but they also face several question marks in this regard, such as the weight transport problem ((3) above) to achieve symmetric weights, and the positive-phase vs negative-phase synchronization question (similar to (5) above).

Our starting point (Sec. 2) proposes an interpretation of the main learning rule observed in biological synapses: Spike-Timing-Dependent Plasticity (STDP). Following up on the ideas presented in Hinton's 2007 talk (Hinton, 2007), we first argue that STDP could be seen as stochastic gradient descent if only the neuron was driven by a feedback signal that either increases or decreases the neuron's firing rate in proportion to the gradient of an objective function with respect to the neuron's voltage potential.

In Sec. 3 we then argue that the above interpretation suggests that neural dynamics (which creates the above changes in neuronal activations thanks to feedback and lateral connections) correspond to *inference* towards neural configurations that are more consistent with each other and with the observations (inputs, targets, or rewards). This view extends Hinton's supervised learning proposal to the unsupervised generative setting. It naturally suggests that the training procedure corresponds to a form of variational EM (Neal & Hinton, 1999) (see Sec.3), possibly based on MAP (maximum a posteriori) or MCMC (Markov Chain Monte-Carlo) approximations. In Sec. 4 we show how this mathematical framework suggests a training procedure for a deep generative network with many layers of latent variables. However, the above interpretation would still require to compute some gradients. Another contribution (Sec. 6) is to show that one can estimate these gradients via an approximation that only involves ordinary neural computation and no explicit derivatives, following previous work on target propagation (Bengio, 2014; Lee et al., 2014).

Although our primary justification for the proposed learning algorithm corresponds to a deep directed graphical model, it turns out that the proposed learning mechanism can be interpreted as training a denoising auto-encoder. As discussed in Sec. 5 these alternative interpretations of the model provide different ways to sample from it, and we found that better samples could be obtained.

## 2. STDP as Stochastic Gradient Descent

Spike-Timing-Dependent Plasticity or STDP is believed to be the main form of synaptic change in neurons (Markram & Sakmann, 1995; Gerstner et al., 1996) and it relates the expected change in synaptic weights to the timing difference between post-synaptic spikes and pre-synaptic spikes. Although it is the result of experimental observations in biological neurons, its interpretation as part of a learning procedure that could explain learning in deep networks remains unclear. This paper aims at proposing such an interpretation, starting from the proposal made by Hinton (2007), but extending these ideas towards unsupervised generative modeling of the data.

What has been observed in STDP is that the weights change if there is a pre-synaptic spike in the temporal vicinity of a post-synaptic spike: that change is positive if the post-synaptic spike happens just after the pre-synaptic spike, negative if it happens just before. As suggested in Hinton's talk, this also corresponds to a temporal derivative filter applied to the post-synaptic firing rate, *at the time of the pre-synaptic spike*. To illustrate this, consider the situation in which two neurons $N_i$ and $N_k$ impinge on neuron $N_j$, and each neuron, say $N_i$, has a voltage potential $V_i$ which, when above threshold, makes the neuron send out a spike $S_i$ with probability (called rate) $R_i$. If $R_k$ increases after a spike $S_i$, in average (over $S_k$), that will increase $V_j$ and $R_j$ and thus the probability of $N_j$'s post-synaptic spike. That will come right after the $N_i$ spike, yielding an increase in the synaptic weight $W_{ij}$ as per STDP. However, if $R_k$ decreases after a spike $S_i$, this decreases the probability of $N_j$ spiking after $N_i$'s spike, or equivalently, making the probability of $N_j$'s spike occuring before $N_i$'s spike larger than the probability of $N_j$'s spike occuring after $N_i$'s spike, i.e., making $V_j$ and $R_j$ smaller. According to STDP, this situation would then correspond to a decrease in the synaptic weight. In conclusion, these arguments suggest that STDP can be interpreted as follows:

$$\Delta W_{ij} \propto S_i \Delta V_j, \qquad (1)$$

where $\Delta$ indicates the temporal change, $S_i$ indicates the pre-synaptic spike (from neuron $i$), and $V_j$ indicates the post-synaptic voltage potential (of neuron $j$).

Clearly, the consequence is that **if the change $\Delta V_j$ corresponds to improving some objective function $J$, then STDP corresponds to approximate stochastic gradient descent in that objective function**. With this view, STDP would implement the delta rule (gradient descent on a one-layer network) if the post-synaptic activation changes in the direction of the gradient.

# 3. Variational EM with Learned Approximate Inference

To take advantage of the above statement, the dynamics of the neural network must be such that *neural activities move towards better values of some objective function J*. Hence we would like to define such an objective function in a way that is consistent with the actual neural computation being performed (for fixed weights $W$), in the sense that the expected temporal change of the voltage potentials approximately corresponds to increases in $J$. In this paper, we are going to consider the voltage potentials as the central variables of interest which influence $J$ and consider them as latent variables $V$ (denoted $h$ below to keep machine learning interpretation general), while we will consider the actual spike trains $S$ as non-linear noisy corruptions of $V$, a form of quantization (with the "noise level" controlled either by the integration time or the number of redundant neurons in an ensemble (Legenstein & Maass, 2014). This view makes the application of the denoising auto-encoder theorems discussed in Sec. 5 more straightforward.

The main contribution of this paper is to propose and give support to the hypothesis that $J$ comes out of a *variational bound* on the likelihood of the data. Variational bounds have been proposed to justify various learning algorithms for generative models (Hinton et al., 1995) (Sec. 7). To keep the mapping to biology open, consider such bounds and the associated criteria that may be derived from them, using an abstract notation with *observed variable x* and *latent variable h*. If we have a model $p(x, h)$ of their joint distribution, as well as some approximate inference mechanism defining a conditional distribution $q^*(H|x)$, the observed data log-likelihood $\log p(x)$ can be decomposed as

$$\log p(x) = \log p(x) \sum_h q^*(h|x)$$
$$= \sum_h q^*(h|x) \log \frac{p(x,h)q^*(h|x)}{p(h|x)q^*(h|x)}$$
$$= E_{q^*(H|x)}[\log p(x,H)] + \mathbf{H}[q^*(H|x)]$$
$$\qquad + \mathbf{KL}(q^*(H|x)||p(H|x)), \quad (2)$$

where $\mathbf{H}[]$ denotes entropy and $\mathbf{KL}(||)$ the Kullback-Leibler (KL) divergence, and where we have used sums but integrals should be considered when the variables are continuous. Since both the entropy and the KL-divergence are non-negative, we can either bound the log-likelihood via

$$\log p(x) \geq E_{q^*(H|x)}[\log p(x,H)] + \mathbf{H}[q^*(H|x)], \quad (3)$$

or if we care only about optimizing $p$,

$$\log p(x) \geq E_{q^*(H|x)}[\log p(x,H)]. \quad (4)$$

The idea of variational bounds as proxies for the log-likelihood is that *as far as optimizing p is concerned, i.e., dropping the entropy term which does not depend on p,*

*the bound becomes tight when $q^*(H|x) = p(H|x)$*. This suggests that $q^*(H|x)$ should approximate $p(H|x)$. Fixing $q^*(H|x) = p(H|x)$ and optimizing $p$ with $q$ fixed is the EM algorithm. Here (and in general) this is not possible so we consider variational methods in which $q^*(H|x)$ approximates but does not reach $p(H|x)$.

We propose to decompose $q^*(H|x)$ in two components: *parametric initialization* $q_0(H|x) = q(H|x)$ and *iterative inference*, implicitly defining $q^*(H|x) = q_T(H|x)$ via a deterministic or stochastic update, or transition operator

$$q_t(H|x) = A(x) \, q_{t-1}(H|x). \quad (5)$$

The variational bound suggests that $A(x)$ should gradually bring $q_t(H|x)$ closer to $p(H|x)$. At the same time, to make sure that a few steps will be sufficient to approach $p(H|x)$, one may add a term in the objective function to make $q_0(H|x)$ closer to $p(H|x)$, as well as to encourage $p(x, h)$ to favor solutions $p(H|x)$ that can be easily approximated by $q_t(H|x)$ even for small $t$.

For this purpose, consider as training objective a regularized variational MAP-EM criterion (for a given $x$):

$$J = \log p(x,h) + \alpha \log q(h|x), \quad (6)$$

where $h$ is a *free variable* (for each $x$) initialized from $q(H|x)$ and then iteratively updated to approximately maximize $J$. The total objective function is just the average of $J$ over all examples after having performed inference (the approximate maximization over $h$ for each $x$). A reasonable variant would not just encourage $q = q_0$ to generate $h$ (given $x$), but all the $q_t$'s for $t > 0$ as well. Alternatively, the iterative inference could be performed by *stochastically* increasing $J$, i.e., via a Markov chain which may correspond to probabilistic inference with spiking neurons (Pecevski et al., 2011). The corresponding variational MAP or variational MCMC algorithm would be as in Algorithm 1. For the stochastic version one would inject noise when updating $h$. Variational MCMC (de Freitas et al., 2001) can be used to approximate the posterior, e.g., as in the model from Salimans et al. (2014). However, a rejection step does not look very biologically plausible (both for the need of returning to a previous state and for the need to evaluate the joint likelihood, a global quantity). On the other hand, a *biased* MCMC with no rejection step, such as the stochastic gradient Langevin MCMC of Welling & Teh (2011) can work very well in practice.

# 4. Training a Deep Generative Model

There is strong biological evidence of a distinct pattern of connectivity between cortical areas that distinguishes between "feedforward" and "feedback" connections (Douglas et al., 1989) at the level of the microcircuit of cortex (i.e., feedforward and feedback connections do not land in the same type of cells). Furthermore, the feedforward connec-

**Algorithm 1** Variational MAP (or MCMC) SGD algorithm for gradually improving the agreement between the values of the latent variables $h$ and the observed data $x$. $q(h|x)$ is a learned parametric initialization for $h$, $p(h)$ is a parametric prior on the latent variables, and $p(x|h)$ specifies how to generate $x$ given $h$. Objective function $J$ is defined in Eq. 6 Learning rates $\delta$ and $\epsilon$ respectively control the optimization of $h$ and of parameters $\theta$ (of both $q$ and $p$).

---
Initialize $h \sim q(h|x)$
**for** $t = 1$ to $T$ **do**
   $h \leftarrow h + \delta \frac{\partial J}{\partial h}$    (optional: add noise for MCMC)
**end for**
$\theta \leftarrow \theta + \epsilon \frac{\partial J}{\partial \theta}$

---

tions form a directed acyclic graph with nodes (areas) updated in a particular order, e.g., in the visual cortex (Felleman & Essen, 1991). So consider Algorithm 1 with $h$ decomposed into multiple layers, with the conditional independence structure of a directed graphical model structured as a chain, both for $p$ (going down) and for $q$ (going up):

$$p(x,h) = p(x|h^{(1)}) \left( \prod_{k=1}^{M-1} p(h^{(k)}|h^{(k+1)}) \right) p(h^{(M)})$$

$$q(h|x) = q(h^{(1)}|x) \prod_{k=1}^{M-1} q(h^{(k+1)}|h^{(k)}). \tag{7}$$

This clearly decouples the updates associated with each layer, for both $h$ and $\theta$, making these updates "local" to the layer $k$, based on "feedback" from layer $k-1$ and $k+1$. Nonetheless, thanks to the iterative nature of the updates of $h$, all the layers are interacting via both feedforward ($q(h^{(k)}|h^{(k-1)})$) and feedback ($p(h^{(k)}|h^{(k-1)})$) paths. Denoting $x = h^{(0)}$ to simplify notation, the $h$ update would thus consist in moves of the form

$$h^{(k)} \leftarrow h^{(k)} + \delta \frac{\partial}{\partial h^{(k)}} \left( \log(p(h^{(k-1)}|h^{(k)})p(h^{(k)}|h^{(k+1)})) \right.$$
$$\left. + \alpha \log(q(h^{(k)}|h^{(k-1)})q(h^{(k+1)}|h^{(k)})) \right), \tag{8}$$

where $\alpha$ is as in Eq. 6. No back-propagation is needed for the above derivatives when $h^{(k)}$ is on the left hand side of the conditional probability bar. Sec. 6 deals with the right hand side case. For the left hand side case, e.g., $p(h^{(k)}|h^{(k+1)})$ a conditional Gaussian with mean $\mu$ and variance $\sigma^2$, the gradient with respect to $h^{(k)}$ is simply $\frac{\mu - h^{(k)}}{\sigma^2}$. Note that there is an interesting interpretation of such a deep model: the layers above $h^{(k)}$ provide a complex implicitly defined prior for $p(h^{(k)})$.

# 5. Alternative Interpretations as Denoising Auto-Encoder

By inspection of Algorithm 1, one can observe that this algorithm trains $p(x|h)$ and $q(h|x)$ to form complementary pairs of an *auto-encoder* (since the input of one is the target of the other and vice-versa). Note that from that point of view any of the two can act as encoder and the other as decoder for it, depending on whether we start from $h$ or from $x$. In the case of multiple latent layers, each pair of conditionals $q(h^{(k+1)}|h^{(k)})$ and $p(h^{(k)}|h^{(k+1)})$ forms a symmetric auto-encoder, i.e., either one can act as the encoder and the other as the corresponding decoder, since they are trained with the same $(h^{(k)}, h^{(k+1)})$ pairs (but with reversed roles of input and target).

In addition, if noise is injected, e.g., in the form of the quantization induced by a spike train, then the trained auto-encoders are actually denoising auto-encoders, which means that both the encoders and decoders are *contractive*: in the neighborhood of the observed $(x,h)$ pairs, they map neighboring "corrupted" values to the "clean" $(x,h)$ values.

## 5.1. Joint Denoising Auto-Encoder with Latent Variables

This suggests considering a special kind of "joint" denoising auto-encoder which has the pair $(x,h)$ as "visible" variable, an auto-encoder that implicitly estimates an underlying $p(x,h)$. The transition operator[3] for that joint visible-latent denoising auto-encoder is the following in the case of a single hidden layer:

$$(\tilde{x}, \tilde{h}) \leftarrow \text{corrupt}(x, h)$$
$$h \sim q(h|\tilde{x}) \qquad x \sim p(x|\tilde{h}), \tag{9}$$

where the corruption may correspond to the stochastic quantization induced by the neuron non-linearity and spiking process. In the case of a middle layer $h^{(k)}$ in a deeper model, the transition operator must account for the fact that $h^{(k)}$ can either be reconstructed from above or from below, yielding, with probability say $\frac{1}{2}$,

$$h^{(k)} \sim p(h^{(k)}|\tilde{h}^{(k+1)}), \tag{10}$$

and with one minus that probability,

$$h^{(k)} \sim q(h^{(k)}|\tilde{h}^{(k-1)}). \tag{11}$$

Since this interpretation provides a different model, it also provides a different way of *generating* samples. Especially for shallow, we have found that better samples could be obtained in this way, i.e., running the Markov chain with the above transition operator for a few steps.

There might be a geometric interpretation for the improved quality of the samples when they are obtained in this way,

---

[3]See Theorem 1 from Bengio et al. (2013) for the generative interpretation of denoising auto-encoders: it basically states that one can sample from the model implicitly estimated by a denoising auto-encoder by simply alternating noise injection (corruption), encoding and decoding, these forming each step of a generative Markov chain.

compared to the directed generative model that was defined earlier. Denote $q^*(x)$ the empirical distribution of the data, which defines a joint $q^*(h,x) = q^*(x)q^*(h|x)$. Consider the likely situation where $p(x,h)$ is not well matched to $q^*(h,x)$ because for example the parametrization of $p(h)$ is not powerful enough to capture the complex structure in the empirical distribution $q^*(h)$ obtained by mapping the training data through the encoder and inference $q^*(h|x)$. Typically, $q^*(x)$ would concentrate on a manifold and the encoder would not be able to completely unfold it, so that $q^*(h)$ would contain complicated structure with pockets or manifolds of high probability. If $p(h)$ is a simple factorized model, then it will generate values of $h$ that do not correspond well to those seen by the decoder $p(x|h)$ when it was trained, and these out-of-manifold samples in $h$-space are likely to be mapped to out-of-manifold samples in $x$-space. One solution to this problem is to increase the capacity of $p(h)$ (e.g., by adding more layers on top of $h$). Another is to make $q(h|x)$ more powerful (which again can be achieved by increasing the depth of the model, but this time by inserting additional layers below $h$). Now, there is a cheap way of obtaining a very deep directed graphical model, by unfolding the Markov chain of an MCMC-based generative model for a fixed number of steps, i.e., considering each step of the Markov chain as an extra "layer" in a deep directed generative model, with shared parameters across these layers. As we have seen that there is such an interpretation via the joint denoising auto-encoder over both latent and visible, this idea can be immediately applied. We know that each step of the Markov chain operator moves its input distribution closer to the stationary distribution of the chain. So if we start from samples from a very broad (say factorized) prior $p(h)$ and we iteratively encode/decode them (injecting noise appropriately as during training) by successively sampling from $p(x|h)$ and then from $q(h|x)$, the resulting $h$ samples should end up looking more like those seen during training (i.e., from $q^*(h)$).

### 5.2. Latent Variables as Corruption

There is another interpretation of the training procedure, also as a denoising auto-encoder, which has the advantage of producing a generative procedure that is the same as the inference procedure except for $x$ being unclamped.

We return again to the generative interpretation of the denoising criterion for auto-encoders, but this time we consider the non-parametric process $q^*(h|x)$ as a kind of corruption of $x$ that yields the $h$ used as input for reconstructing the observed $x$ via $p(x|h)$. Under that interpretation, a valid generative procedure consists at each step in first performing inference, i.e., sampling $h$ from $q^*(h|x)$, and second sampling from $p(x|h)$. Iterating these steps generates $x$'s according to the Markov chain whose stationary distribution is an estimator of the data generating distribution that produced the training $x$'s (Bengio et al., 2013).

This view does not care about how $q^*(h|x)$ is constructed, but it tells us that if $p(x|h)$ is trained to maximize reconstruction probability, then we can sample in this way from the implicitly estimated model.

We have also found good results using this procedure (Algorithm 2 below), and from the point of view of biological plausibility, it would make more sense that "generating" should involve the same operations as "inference", except for the input being observed or not.

## 6. Targetprop instead of Backprop

In Algorithm 1 and the related stochastic variants Eq. 8 suggests that back-propagation (through one layer) is still needed when $h^{(k)}$ is on the right hand side of the conditional probability bar, e.g., to compute $\frac{\partial p(h^{(k-1)}|h^{(k)})}{\partial h^{(k)}}$. Such a gradient is also the basic building block in back-propagation for supervised learning: we need to back-prop through one layer, e.g. to make $h^{(k)}$ more "compatible" with $h^{(k-1)}$. This provides a kind *error signal*, which in the case of unsupervised learning comes from the sensors, and in the case of supervised learning, comes from the layer holding the observed "target".

Based on recent theoretical results on denoising auto-encoders, we propose the following estimator (up to a scaling constant) of the required gradient, which is related to previous work on "target propagation" (Bengio, 2014; Lee et al., 2014) or targetprop for short. To make notation simpler, we focus below on the case of two layers $h$ and $x$ with "encoder" $q(h|x)$ and "decoder" $p(x|h)$, and we want to estimate $\frac{\partial \log p(x|h)}{\partial h}$. We start with the special case where $p(x|h)$ is a Gaussian with mean $g(h)$ and $q(h|x)$ is Gaussian with mean $f(x)$, i.e., $f$ and $g$ are the deterministic components of the encoder and decoder respectively. The proposed estimator is then

$$\widehat{\Delta h} = \frac{f(x) - f(g(h))}{\sigma_h^2}, \qquad (12)$$

where $\sigma_h^2$ is the variance of the noise injected in $q(h|x)$.

Let us now justify this estimator. Theorem 2 by Alain & Bengio (2013) states that in a denoising auto-encoder with reconstruction function $r(x) = \text{decode}(\text{encode}(x))$, a well-trained auto-encoder estimates the log-score via the difference between its reconstruction and its input:

$$\frac{r(x) - x}{\sigma^2} \to \frac{\partial \log p(x)}{\partial x},$$

where $\sigma^2$ is the variance of injected noise, and $p(x)$ is the implicitly estimated density. We are now going to consider two denoising auto-encoders and apply this theorem to them. First, we note that the gradient $\frac{\partial \log p(x|h)}{\partial h}$ that we wish to estimate can be decomposed as follows:

$$\frac{\partial \log p(x|h)}{\partial h} = \frac{\partial \log p(x,h)}{\partial h} - \frac{\partial \log p(h)}{\partial h}.$$

Hence it is enough to estimate $\frac{\partial \log p(x,h)}{\partial h}$ as well as $\frac{\partial \log p(h)}{\partial h}$. The second one can be estimated by considering the auto-encoder which estimates $p(h)$ implicitly and for which $g$ is the encoder (with $g(h)$ the "code" for $h$) and $f$ is the decoder (with $f(g(h))$ the "reconstruction" of $h$). Hence we have that $\frac{f(g(h))-h}{\sigma_h^2}$ is an estimator of $\frac{\partial \log p(h)}{\partial h}$.

The other gradient can be estimated by considering the joint denoising auto-encoder over $(x,h)$ introduced in the previous section. The (noise-free) reconstruction function for that auto-encoder is

$$r(x,h) = (g(h), f(x)).$$

Hence $\frac{f(x)-h}{\sigma_h^2}$ is an estimator of $\frac{\partial \log p(x,h)}{\partial h}$. Combining the two estimators, we get

$$\frac{(f(x)-h)}{\sigma_h^2} - \frac{(f(g(h))-h)}{\sigma_h^2} = \frac{f(x) - f(g(h))}{\sigma_h^2},$$

which corresponds to Eq. 12.



*Figure 1.* The optimal $h$ for maximizing $p(x|h)$ is $\tilde{h}$ s.t. $g(\tilde{h}) = x$. Since the encoder $f$ and decoder $g$ are approximate inverses of each other, their composition makes a small move $\Delta x$. Eq. 12 is obtained by assuming that by considering an $\tilde{x}$ at $x - \Delta$ and applying $f \circ g$, one would approximately recover $x$, which should be true if the changes are small and the functions smooth (see Lee & Bengio (2014) for a detailed derivation).

Another way to obtain the same formula from a geometric perspective is illustrated in Figure 1. It was introduced in Lee & Bengio (2014) in the context of a backprop-free algorithm for training a denoising auto-encoder.

## 7. Related Work

The main inspiration for the proposed framework is the biological implementation of back-propagation proposed by Hinton (2007). In that talk, Hinton suggests that STDP corresponds to a gradient update step with the gradient on the voltage potential corresponding to its temporal derivative. To obtain the supervised back-propagation update in the proposed scenario would require symmetric weights and synchronization of the computations in terms of feedforward and feedback phases.

Our proposal extends these ideas to include unsupervised learning, avoids the need for symmetric weights, and exploits inference to obtain targets and a probabilistic interpretation as the optimization of a variational bound on the

---

**Algorithm 2** Inference, training and generative procedures used in Experiment 1, for a model with three layers $x$, $h_1$, $h_2$. $f_i()$ is the feedforward map from layer $i-1$ to layer $i$ and $g_i()$ is the feedback map from layer $i$ to layer $i-1$, with $x = h_0$ being layer 0.

Define INFERENCE($x$, $N$=15, $\delta$=0.1, $\alpha$=0.001):
Feedforward pass: $h_1 \leftarrow f_1(x)$, $\quad h_2 \leftarrow f_2(h_1)$
**for** $t = 1$ to $N$ **do**
$\quad h_2 \leftarrow h_2 + \delta(f_2(h_1) - f_2(g_2(h_2)))$
$\quad h_1 \leftarrow h_1 + \delta(f_1(x) - f_1(g_1(h_1))) + \alpha(g_2(h_2) - h_1)$
**end for**
**Return** $h_1, h_2$

Define TRAIN()
**for** $x$ in training set **do**
$\quad$ do INFERENCE($x$)
$\quad$ train each layer (both $f_l$ and $g_l$) by taking Gaussian-corrupted value of other layer as input and the clean inferred value as target (i.e. applying the delta rule). For the top sigmoid layer, we sample 3 binary values and average them as a spike-like corruption.
**end for**
Compute the mean and variance of the $h_2$ values inferred in the training set. Multiply the variances by 4. Define $p(h_2)$ as sampling from this Gaussian.

Define GENERATE():
Sample $h_2$ from $p(H_2)$
**for** $t = 1$ to 3 **do**
$\quad h_1, h_2 \leftarrow$ INFERENCE($x$,$\alpha = 0.3$)
$\quad x \leftarrow g_1(h_1)$
**end for**
**Return** $x$

---

likelihood. There is also an interesting connection with an earlier proposal for a more biologically plausible implementation of supervised back-propagation (Xie & Seung, 2003) which also relies on iterative inference (a deterministic relaxation in that case), but needs symmetric weights.

Another important inspiration is Predictive Sparse Decomposition (PSD) (Kavukcuoglu et al., 2008). PSD is a special case of Algorithm 1 when there is only one layer and the encoder $q(h|x)$, decoder $p(x|h)$, and prior $p(h)$ have a specific form which makes $p(x,h)$ a sparse coding model and $q(h|x)$ a fast parametric approximation of the correct posterior. Our proposal extends PSD by providing a justification for the training criterion as a variational bound, by generalizing to multiple layers of latent variables, and by providing associated generative procedures.

The combination of a parametric approximate inference machine (the encoder) and a generative decoder (each with possibly several layers of latent variables) is an old theme

Figure 2. MNIST samples generated by GENERATE from Algorithm 2 after training with TRAIN.
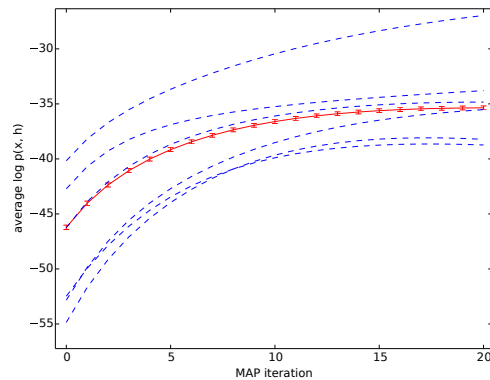


Figure 3. Increase of $\log p(x, h)$ over 20 iterations of the INFERENCE algorithm 2, showing that the targetprop updates increase the joint likelihood. The solid red line shows the average and the standard error over the full testset containing 10,000 digits. Dashed lines show $\log p(x, h)$ for individual datapoints.

that was started with the Wake-Sleep algorithm (Hinton et al., 1995) and which finds very interesting instantiations in the variational auto-encoder (Kingma & Welling, 2014; Kingma et al., 2014) and the reweighted wake-sleep algorithm (Bornschein & Bengio, 2014). Two important differences with the approach proposed here is that here we avoid back-propagation thanks to an inference step that approximates the posterior. In this spirit, see the recent work introducing MCMC inference for the variational auto-encoder Salimans et al. (2014).

The proposal made here also owes a lot to the idea of target propagation introduced in Bengio (2014); Lee et al. (2014), to which it adds the idea that in order to find a target that is consistent with both the input and the final output target, it makes sense to perform iterative inference, reconciling the bottom-up and top-down pressures. Addressing the weight transport problem (the weight symmetry constraint) was also done for the supervised case using *feedback alignment* (Lillicrap et al., 2014): even if the feedback weights do not exactly match the feedforward weights, the latter learn to align to the former and "back-propagation" (with the wrong feedback weights) still works.

The targetprop formula avoiding back-propagation through one layer is actually the same as proposed by Lee & Bengio (2014) for backprop-free auto-encoders. What has been added here is a justification of this specific formula based on the denoising auto-encoder theorem from Alain & Bengio (2013), and the empirical validation of its ability to climb the joint likelihood for variational inference.

## 8. Experimental Validation

Figure 2 shows generated samples obtained after training on MNIST with Algorithm 2 (derived from the considerations of Sec.s 4, 5 and 6). The network has two hid-

den layers, $h_1$ with 1000 softplus units and $h_2$ with 100 sigmoid units (which can be considered biologically plausible (Glorot et al., 2011)). We trained for 20 epochs, with minibatches of size 100 to speed-up computation using GPUs. Results can be reproduced from code at Using the Parzen density estimator previously used for that data, we obtain an estimated log-likelihood LL=236 (using a standard deviation of 0.2 for the Parzen density estimator, chosen with the validation set), which is about the same or better as was obtained for contractive auto-encoders (Rifai et al., 2011) (LL=121), deeper generative stochastic networks (Bengio et al., 2014) (LL=214) and generative adversarial networks (Goodfellow et al., 2014) (LL=225). In accordance with Algorithm 2, the variances of the conditional densities are 1, and the top-level prior is ignored during most of training (as if it was a very broad, uniform prior) and only set to the Gaussian by the end of training, before generation (by setting the parameters of $p(h_2)$ to the empirical mean and variance of the projected training examples at the top level). Figure 3 shows that the targetprop updates (instead of the gradient updates) allow the inference process to indeed smoothly increase the joint likelihood. Note that if we sample using the directed graphical model $p(x|h)p(h)$, the samples are not as good and LL=126, suggesting as discussed in Sec. 5 that additional inference and encode/decode iterations move $h$ towards values that are closer to $q^*(h)$ (the empirical distribution of inferred states from training examples). The experiment illustrated in Figure 4 shows that the proposed inference mechanism can be used to fill-in missing values with a trained model. The model is the same that was trained using Algorithm 2 (with samples shown in Figure 2). 20 iterations steps of encode/decode as described below were performed, with a call to INFERENCE (to maximize $p(x, h)$ over $h$) for each step, with a slight modification. Instead of using $f_1(x) - f_1(g_1(h))$ to account for
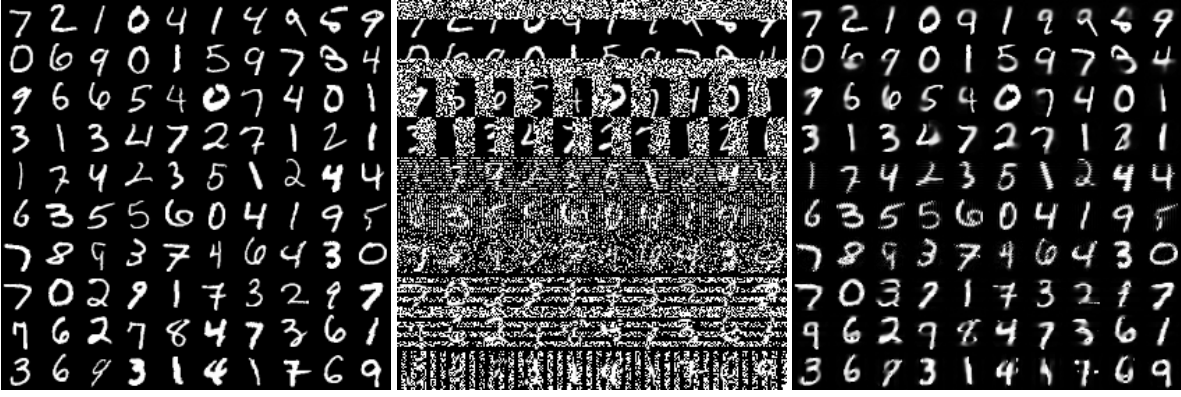
*Figure 4.* Examples of filling-in (in-painting) missing (initially corrupted) parts of an image. Left: original MNIST test examples. Middle: initial state of the inference, with half of the pixels randomly sampled (with a different corruption pattern in each row of the figure). Right: reconstructions using a variant of the INFERENCE procedure of Algorithm 2 for the case when some inputs are clamped.

the pressure of $x$ upon $h$ (towards maximizing $p(x|h)$), we used $f_1(x^v, g^m(h)) - f(g(h))$, where $x^v$ is the part of $x$ that is visible (clamped) while $g^m(h)$ is the part of the output of $g(h)$ that concerns the missing (corrupted) inputs. This formula was derived from the same consideration as for Eq. 12, but where the quantity of interest is $\frac{\partial \log p(x^v|h)}{\partial h}$ rather than $\frac{\partial \log p(x|h)}{\partial h}$, and we consider that the reconstruction of $h$, given $x^v$, fills-in the missing inputs $(x^m)$ from $g^m(h)$.

## 9. Future Work and Conclusion

We consider this paper as an exploratory step towards explaining a central aspect of the brain's learning algorithm: *credit assignment through many layers*. As argued by Bengio (2014); Lee et al. (2014), departing from back-propagation could be useful not just for biological plausibility but from a machine learning point of view as well: by working on the "targets" for the intermediate layers, we may avoid the kind of reliance on smoothness and derivatives that characterizes back-propagation, as these techniques can in principle work even with highly non-linear transformations for which gradients are often near 0, e.g., with stochastic binary units (Lee et al., 2014). Besides the connection between STDP and variational EM, an important contribution of this paper is to show that the "targetprop" update which estimates the gradient through one layer can be used for inference, yielding systematic improvements in the joint likelihood and allowing to learn a good generative model. Another interesting contribution is that the variational EM updates, with noise added, can also be interpreted as training a denoising auto-encoder over both visible and latent variables, and that iterating from the associated Markov chain yields better samples than those obtained from the directed graphical model estimated by variational EM.

Many directions need to be investigated to follow-up on the work reported here. An important element of neural circuitry is the strong presence of lateral connections between nearby neurons in the same area. In the proposed framework, an obvious place for such lateral connections is to implement the prior on the joint distribution between nearby neurons, something we have not explored in our experiments. For example, Garrigues & Olshausen (2008) have discussed neural implementations of the inference involved in sparse coding based on the lateral connections.

Although we have found that "injecting noise" helped training a better model, more theoretical work needs to be done to explore this replacement of a MAP-based inference by an MCMC-like inference, which should help determine how and how much of this noise should be injected.

Whereas this paper focused on unsupervised learning, these ideas could be applied to supervised learning and reinforcement learning as well. For reinforcement learning, an important role of the proposed algorithms is to learn to predict rewards, although a more challenging question is how the MCMC part could be used to simulate future events. For both supervised learning and reinforcement learning, we would probably want to add a mechanism that would give more weight to minimizing prediction (or reconstruction) error for some of the observed signals (e.g. $y$ is more important to predict than $x$).

Finally, a lot needs to be done to connect in more detail the proposals made here with biology, including neural implementation using spikes with Poisson rates as the source of signal quantization and randomness, taking into account the constraints on the sign of the weights depending on whether the pre-synaptic neuron is inhibitory or excitatory, etc. In addition, although the operations proposed here are backprop-free, they may still require some kinds of synchronizations (or control mechanism) and specific connectivity to be implemented in brains.

## Acknowledgments

## References

Alain, Guillaume and Bengio, Yoshua. What regularized auto-encoders learn from the data generating distribution. In *ICLR'2013*. also arXiv report 1211.4246, 2013.

Bengio, Yoshua. How auto-encoders could provide credit assignment in deep networks via target propagation. Technical report, arXiv:1407.7906, 2014.

Bengio, Yoshua, Yao, Li, Alain, Guillaume, and Vincent, Pascal. Generalized denoising auto-encoders as generative models. In *NIPS'2013*, 2013.

Bengio, Yoshua, Thibodeau-Laufer, Eric, Alain, Guillaume, and Yosinski, Jason. Deep generative stochastic networks trainable by backprop. In *Proceedings of the 30th International Conference on Machine Learning (ICML'14)*, 2014.

Bornschein, Jörg and Bengio, Yoshua. Reweighted wake-sleep. Technical report, arXiv:1406.2751, 2014.

de Freitas, Nando, Højen-Sørensen, Pedro, Jordan, Michael I., and Russell, Stuart. Variational MCMC. In *UAI'2001*, pp. 120–127, 2001.

Douglas, R. J., Martin, K. A., and Whitteridge, D. A canonical microcircuit for neocortex. *Neural Computation*, 1 (480--488), 1989.

Felleman, Daniel J. and Essen, David C. Van. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.

Garrigues, Pierre and Olshausen, Bruno. Learning horizontal connections in a sparse coding model of natural images. In Platt, J.C., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems 20 (NIPS'07)*, pp. 505–512. MIT Press, Cambridge, MA, 2008.

Gerstner, W., Kempter, R., van Hemmen, J.L., and Wagner, H. A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 386:76–78, 1996.

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *AISTATS'2011*, 2011.

Goodfellow, Ian, Le, Quoc, Saxe, Andrew, and Ng, Andrew. Measuring invariances in deep networks. In *NIPS'2009*, pp. 646–654, 2009.

Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial networks. In *NIPS'2014*, 2014.

Hinton, G. E. and Sejnowski, T. J. Learning and relearning in Boltzmann machines. In Rumelhart, D. E. and McClelland, J. L. (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, pp. 282–317. MIT Press, Cambridge, MA, 1986.

Hinton, Geoffrey E. How to do backpropagation in a brain. Invited talk at the NIPS'2007 Deep Learning Workshop, Dec. 2007.

Hinton, Geoffrey E., Dayan, Peter, Frey, Brendan J., and Neal, Radford M. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1558–1161, 1995.

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

Kavukcuoglu, Koray, Ranzato, Marc'Aurelio, and LeCun, Yann. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU, 2008. Tech Report CBLL-TR-2008-12-01.

Kingma, D.P., Rezende, D.J., Mohamed, S., and Welling, M. Semi-supervised learning with deep generative models. In *NIPS'2014*, 2014.

Kingma, Durk P. and Welling, Max. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Lee, Dong-Hyun and Bengio, Yoshua. Backprop-free auto-encoders. NIPS'2014 Deep Learning workshop, 2014.

Lee, Dong-Hyun, Zhang, Saizheng, Biard, Antoine, and Bengio, Yoshua. Target propagation. arXiv 1412.7525, 2014.

Legenstein, Robert and Maass, Wolfgang. Ensembles of spiking neurons with noise support optimal probabilistic inference in a dynamically changing environment. *PLOS Computational Biology*, 10(10), 2014.

Lillicrap, Timothy P., Cownden, Daniel, Tweed, Douglas B., and Akerman, Colin J. Random feedback weights support learning in deep neural networks. arXiv:1411.0247, 2014.

Markram, H. and Sakmann, B. Action potentials propagating back into dendrites triggers changes in efficacy. *Soc. Neurosci. Abs*, 21, 1995.

Neal, R.M. and Hinton, G.E. A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I. (ed.), *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

Pecevski, Dejan, Buesing, Lars, and Maass, Wolfgang. Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLOS Computational Biology*, 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, Vincent, Pascal, Muller, Xavier, Glorot, Xavier, and Bengio, Yoshua. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML'2011*, 2011.

Salakhutdinov, R. and Hinton, G.E. Deep Boltzmann machines. In *AISTATS'2009*, 2009.

Salimans, T., Kingma, D. P., and Welling, M. Markov Chain Monte Carlo and Variational Inference: Bridging the Gap. arXiv:1410.6460, 2014.

Smolensky, Paul. Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D. E. and McClelland, J. L. (eds.), *Parallel Distributed Processing*, volume 1, chapter 6, pp. 194–281. MIT Press, Cambridge, 1986.

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

Welling, Max and Teh, Yee-Whye. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML'2011*, pp. 681–688, 2011.

Xie, Xiaohui and Seung, H. Sebastian. Equivalence of backpropagation and contrastive Hebbian learning in a layered network. *Neural Computation*, 2003.