Auxiliary Deep Generative Models

Lars Maaløe[1]   LARSMA@DTU.DK
Casper Kaae Sønderby[2]   CASPERKAAE@GMAIL.COM
Søren Kaae Sønderby[2]   SKAAESONDERBY@GMAIL.COM
Ole Winther[1,2]   OLWI@DTU.DK

[1]Department of Applied Mathematics and Computer Science, Technical University of Denmark
[2]Bioinformatics Centre, Department of Biology, University of Copenhagen

Abstract

Deep generative models parameterized by neural networks have recently achieved state-of-the-art performance in unsupervised and semi-supervised learning. We extend deep generative models with auxiliary variables which improves the variational approximation. The auxiliary variables leave the generative model unchanged but make the variational distribution more expressive. Inspired by the structure of the auxiliary variable we also propose a model with two stochastic layers and skip connections. Our findings suggest that more expressive and properly specified deep generative models converge faster with better results. We show state-of-the-art performance within semi-supervised learning on MNIST, SVHN and NORB datasets.

1. Introduction

Stochastic backpropagation, deep neural networks and approximate Bayesian inference have made deep generative models practical for large scale problems (Kingma, 2013; Rezende et al., 2014), but typically they assume a mean field latent distribution where all latent variables are independent. This assumption might result in models that are incapable of capturing all dependencies in the data. In this paper we show that deep generative models with more expressive variational distributions are easier to optimize and have better performance. We increase the flexibility of the model by introducing auxiliary variables (Agakov and Barber, 2004) allowing for more complex latent distributions. We demonstrate the benefits of the increased flexibility by achieving state-of-the-art performance in the semi-supervised setting for the MNIST (LeCun et al., 1998),

SVHN (Netzer et al., 2011) and NORB (LeCun et al., 2004) datasets.

Recently there have been significant improvements within the semi-supervised classification tasks. Kingma et al. (2014) introduced a deep generative model for semi-supervised learning by modeling the joint distribution over data and labels. This model is difficult to train end-to-end with more than one layer of stochastic latent variables, but coupled with a pretrained feature extractor it performs well. Lately the Ladder network (Rasmus et al., 2015; Valpola, 2014) and virtual adversarial training (VAT) (Miyato et al., 2015) have improved the performance further with end-to-end training.

In this paper we train deep generative models with multiple stochastic layers. The *Auxiliary Deep Generative Models (ADGM)* utilize an extra set of auxiliary latent variables to increase the flexibility of the variational distribution (cf. Sec. 2.2). We also introduce a slight change to the ADGM, a 2-layered stochastic model with skip connections, the *Skip Deep Generative Model (SDGM)* (cf. Sec. 2.4). Both models are trainable end-to-end and offer state-of-the-art performance when compared to other semi-supervised methods. In the paper we first introduce toy data to demonstrate that:

(i) The auxiliary variable models can fit complex latent distributions and thereby improve the variational lower bound (cf. Sec. 4.1 and 4.3).

(ii) By fitting a complex half-moon dataset using only six labeled data points the ADGM utilizes the data manifold for semi-supervised classification (cf. Sec. 4.2).

For the benchmark datasets we show (cf. Sec. 4.4):

(iii) State-of-the-art results on several semi-supervised classification tasks.

(iv) That multi-layered deep generative models for semi-supervised learning are trainable end-to-end without pre-training or feature engineering.

2. Auxiliary deep generative models

Recently Kingma (2013); Rezende et al. (2014) have coupled the approach of variational inference with deep learning giving rise to powerful probabilistic models constructed by an inference neural network $q(z|x)$ and a generative neural network $p(x|z)$. This approach can be perceived as a variational equivalent to the deep auto-encoder, in which $q(z|x)$ acts as the encoder and $p(x|z)$ the decoder. However, the difference is that these models ensures efficient inference over continuous distributions in the latent space z with automatic relevance determination and regularization due to the KL-divergence. Furthermore, the gradients of the variational upper bound are easily defined by backpropagation through the network(s). To keep the computational requirements low the variational distribution $q(z|x)$ is usually chosen to be a diagonal Gaussian, limiting the expressive power of the inference model.

In this paper we propose a variational auxiliary variable approach (Agakov and Barber, 2004) to improve the variational distribution: The generative model is extended with variables a to $p(x, z, a)$ such that the original model is invariant to marginalization over a: $p(x, z, a) = p(a|x, z)p(x, z)$. In the variational distribution, on the other hand, a is used such that marginal $q(z|x) = \int q(z|a, x)p(a|x)da$ is a general non-Gaussian distribution. This hierarchical specification allows the latent variables to be correlated through a, while maintaining the computational efficiency of fully factorized models (cf. Fig. 1). In Sec. 4.1 we demonstrate the expressive power of the inference model by fitting a complex multimodal distribution.

2.1. Variational auto-encoder

The variational auto-encoder (VAE) has recently been introduced as a powerful method for unsupervised learning. Here a latent variable generative model $p_\theta(x|z)$ for data x is parameterized by a deep neural network with parameters θ. The parameters are inferred by maximizing the variational lower bound of the likelihood $-\sum_i \mathcal{U}_{\text{VAE}}(x_i)$ with

$$\log p_\theta(x) = \log \int_z p(x, z)dz$$
$$\geq \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x|z)p_\theta(z)}{q_\phi(z|x)} \right] \qquad (1)$$
$$\equiv -\mathcal{U}_{\text{VAE}}(x) .$$

The inference model $q_\phi(z|x)$ is parameterized by a second deep neural network. The inference and generative parameters, θ and ϕ, are jointly trained by optimizing Eq. 1 with stochastic gradient ascent, where we use the reparameterization trick for backpropagation through the Gaussian latent variables (Kingma, 2013; Rezende et al., 2014).
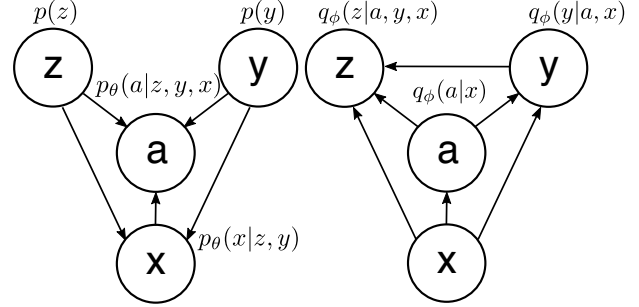
(a) Generative model P. (b) Inference model Q.

Figure 1. Probabilistic graphical model of the ADGM for semi-supervised learning. The incoming joint connections to each variable are deep neural networks with parameters θ and ϕ.

2.2. Auxiliary variables

We propose to extend the variational distribution with auxiliary variables a: $q(a, z|x) = q(z|a, x)q(a|x)$ such that the marginal distribution $q(z|x)$ can fit more complicated posteriors $p(z|x)$. In order to have an unchanged generative model, $p(x|z)$, it is required that the joint mode $p(x, z, a)$ gives back the original $p(x, z)$ under marginalization over a, thus $p(x, z, a) = p(a|x, z)p(x, z)$. Auxiliary variables are used in the EM algorithm and Gibbs sampling and have previously been considered for variational learning by Agakov and Barber (2004). Concurrent with this work Ranganath et al. (2015) have proposed to make the parameters of the variational distribution stochastic, which leads to a similar model. It is important to note that in order not to fall back to the original VAE model one has to require $p(a|x, z) \neq p(a)$, see Agakov and Barber (2004) and App. A. The auxiliary VAE lower bound becomes

$$\log p_\theta(x) = \log \int_a \int_z p(x, a, z)dadz$$
$$\geq \mathbb{E}_{q_\phi(a,z|x)} \left[\log \frac{p_\theta(a|z, x)p_\theta(x|z)p(z)}{q_\phi(a|x)q_\phi(z|a, x)} \right] \quad (2)$$
$$\equiv -\mathcal{U}_{\text{AVAE}}(x) .$$

with $p_\theta(a|z, x)$ and $q_\phi(a|x)$ diagonal Gaussian distributions parameterized by deep neural networks.

2.3. Semi-supervised learning

The main focus of this paper is to use the auxiliary approach to build semi-supervised models that learn classifiers from labeled and unlabeled data. To encompass the class information we introduce an extra latent variable y. The generative model P is defined as $p(y)p(z)p_\theta(a|z, y, x)p_\theta(x|y, z)$ (cf. Fig. 1a):

$$p(z) = \mathcal{N}(z|0, \mathrm{I}), \qquad (3)$$
$$p(y) = \text{Cat}(y|\pi), \qquad (4)$$
$$p_\theta(a|z, y, x) = f(a; z, y, x, \theta), \qquad (5)$$
$$p_\theta(x|z, y) = f(x; z, y, \theta) , \qquad (6)$$

where a, y, z are the auxiliary variable, class label, and latent features, respectively. $\text{Cat}(\cdot)$ is a multinomial distribution, where y is treated as a latent variable for the unlabeled data points. In this study we only experimented with categorical labels, however the method applies to other distributions for the latent variable y. $f(x; z, y, \theta)$ is iid categorical or Gaussian for discrete and continuous observations x. $p_\theta(\cdot)$ are deep neural networks with parameters θ. The inference model is defined as $q_\phi(a|x)q_\phi(z|a,y,x)q_\phi(y|a,x)$ (cf. Fig. 1b):

$$q_\phi(a|x) = \mathcal{N}(a|\mu_\phi(x), \text{diag}(\sigma_\phi^2(x))), \tag{7}$$

$$q_\phi(y|a,x) = \text{Cat}(y|\pi_\phi(a,x)), \tag{8}$$

$$q_\phi(z|a,y,x) = \mathcal{N}(z|\mu_\phi(a,y,x), \text{diag}(\sigma_\phi^2(a,y,x))) . \tag{9}$$

In order to model Gaussian distributions $p_\theta(a|z,y,x)$, $p_\theta(x|z,y)$, $q_\phi(a|x)$ and $q_\phi(z|a,y,x)$ we define two separate outputs from the top deterministic layer in each deep neural network, $\mu_{\phi \vee \theta}(\cdot)$ and $\log \sigma_{\phi \vee \theta}^2(\cdot)$. From these outputs we are able to approximate the expectations $\mathbb{E}$ by applying the reparameterization trick.

The key point of the ADGM is that the auxiliary unit a introduce a latent feature extractor to the inference model giving a richer mapping between x and y. We can use the classifier (9) to compute probabilities for unlabeled data x_u being part of each class and to retrieve a cross-entropy error estimate on the labeled data x_l. This can be used in cohesion with the variational lower bound to define a good objective function in order to train the model end-to-end.

VARIATIONAL LOWER BOUND

We optimize the model by maximizing the lower bound on the likelihood (cf. App. B for more details). The variational lower bound on the marginal likelihood for a single *labeled data point* is

$$\log p_\theta(x, y) = \log \int_a \int_z p(x, y, a, z) dz da$$

$$\geq \mathbb{E}_{q_\phi(a,z|x,y)} \left[\log \frac{p_\theta(x, y, a, z)}{q_\phi(a, z|x, y)} \right] \tag{10}$$

$$\equiv -\mathcal{L}(x, y) ,$$

with $q_\phi(a,z|x,y) = q_\phi(a|x)q_\phi(z|a,y,x)$. For unlabeled data we further introduce the variational distribution for y, $q_\phi(y|a,x)$:

$$\log p_\theta(x) = \log \int_a \int_y \int_z p(x, y, a, z) dz dy da$$

$$\geq \mathbb{E}_{q_\phi(a,y,z|x)} \left[\log \frac{p_\theta(x, y, a, z)}{q_\phi(a, y, z|x)} \right] \tag{11}$$

$$\equiv -\mathcal{U}(x) ,$$

with $q_\phi(a,y,z|x) = q_\phi(z|a,y,x)q_\phi(y|a,x)q_\phi(a|x)$.

The classifier (9) appears in $-\mathcal{U}(x_u)$, but not in $-\mathcal{L}(x_l, y_l)$. The classification accuracy can be improved by introducing an explicit classification loss for labeled data:

$$\mathcal{L}_l(x_l, y_l) = \tag{12}$$
$$\mathcal{L}(x_l, y_l) + \alpha \cdot \log \mathbb{E}_{q_\phi(a|x_l)}[-\log q_\phi(y_l|a, x_l)] ,$$

where α is a weight between generative and discriminative learning. The α parameter is set to $\beta \cdot \frac{N_l + N_u}{N_l}$, where β is a scaling constant, N_l is the number of labeled data points and N_u is the number of unlabeled data points. The objective function for labeled and unlabeled data is

$$\mathcal{J} = \sum_{(x_l, y_l)} \mathcal{L}_l(x_l, y_l) + \sum_{(x_u)} \mathcal{U}(x_u) . \tag{13}$$

2.4. Two stochastic layers with skip connections

Kingma et al. (2014) proposed a model with two stochastic layers but were unable to make it converge end-to-end and instead resorted to layer-wise training. In our preliminary analysis we also found that this model: $p_\theta(x|z_1)p_\theta(z_1|z_2, y)p(z_2)p(y)$ failed to converge when trained end-to-end. On the other hand, the auxiliary model can be made into a two-layered stochastic model by simply reversing the arrow between a and x in Fig. 1a. We would expect that if the auxiliary model works well in terms of convergence and performance then this two-layered model (a is now part of the generative model): $p_\theta(x|y, a, z)p_\theta(a|z, y)p(z)p(y)$ should work even better because it is a more flexible generative model. The variational distribution is unchanged: $q_\phi(z|y, x, a)q_\phi(y|a, x)q_\phi(a|x)$. We call this the *Skip Deep Generative Model (SDGM)* and test it alongside the auxiliary model in the benchmarks (cf. Sec. 4.4).

3. Experiments

The SDGM and ADGM are each parameterized by 5 neural networks (NN): (1) auxiliary inference model $q_\phi(a|x)$, (2) latent inference model $q_\phi(z|a, y, x)$, (3) classification model $q_\phi(y|a, x)$, (4) generative model $p_\theta(a|\cdot)$, and (5) the generative model $p_\theta(x|\cdot)$.

The neural networks consists of M fully connected hidden layers with h_j denoting the output of a layer $j = 1, ..., M$. All hidden layers use rectified linear activation functions. To compute the approximations of the stochastic variables we place two independent output layers after h_M, μ and $\log \sigma^2$. In a forward-pass we are propagating the input x through the neural network by

$$h_M = \text{NN}(x) \tag{14}$$

$$\mu = \text{Linear}(h_M) \tag{15}$$

$$\log \sigma^2 = \text{Linear}(h_M) , \tag{16}$$

with Linear denoting a linear activation function. We then approximate the stochastic variables by applying the reparameterization trick using the μ and $\log \sigma^2$ outputs.

In the unsupervised toy example (cf. Sec. 4.1) we applied 3 hidden layers with $\dim(h) = 20$, $\dim(a) = 4$ and $\dim(z) = 2$. For the semi-supervised toy example (cf. Sec. 4.2) we used two hidden layers of $\dim(h) = 100$ and $\dim(a, z) = 10$.

For all the benchmark experiments (cf. Sec. 4.4) we parameterized the deep neural networks with two fully connected hidden layers. Each pair of hidden layers was of size $\dim(h) = 500$ or $\dim(h) = 1000$ with $\dim(a, z) = 100$ or $\dim(a, z) = 300$. The generative model was $p(y)p(z)p_\theta(a|z, y)p_\theta(x|z, y)$ for the ADGM and the SDGM had the augmented $p_\theta(x|a, z, y)$. Both have unchanged inference models (cf. Fig. 1b).

All parameters are initialized using the Glorot and Bengio (2010) scheme. The expectation over the a and z variables were performed by Monte Carlo sampling using the reparameterization trick (Kingma, 2013; Rezende et al., 2014) and the average over y by exact enumeration so

$$\mathbb{E}_{q_\phi(a,y,z|x)}\left[f(a, x, y, z)\right] \approx \qquad (17)$$

$$\frac{1}{N_{\text{samp}}} \sum_i^{N_{\text{samp}}} \sum_y q(y|a_i, x)f(a_i, x, y, z_{yi}),$$

with $a_i \sim q(a|x)$ and $z_{yi} \sim q(z|a, y, x)$.

For training, we have used the *Adam* (Kingma and Ba, 2014) optimization framework with a learning rate of 3e-4, exponential decay rate for the 1st and 2nd moment at 0.9 and 0.999, respectively. The β constant was between 0.1 and 2 throughout the experiments.

The models are implemented in Python using Theano (Bastien et al., 2012), Lasagne (Dieleman et al., 2015) and Parmesan libraries[1].

For the MNIST dataset we have combined the training set of 50000 examples with the validation set of 10000 examples. The test set remained as is. We used a batch size of 200 with half of the batch always being the 100 labeled samples. The labeled data are chosen randomly, but distributed evenly across classes. To speed up training, we removed the columns with a standard deviation below 0.1 resulting in an input size of $\dim(x) = 444$. Before each epoch the normalized MNIST images were binarized by sampling from a Bernoulli distribution with mean parameter set to the pixel intensities.

For the SVHN dataset we used the vectorized and cropped training set $\dim(x) = 3072$ with classes from 0 to 9, com-

[1]Implementation is available in a repository named auxiliary-deep-generative-models on github.com.

bined with the *extra* set resulting in 604388 data points. The test set is of size 26032. We trained on the *small* NORB dataset consisting of 24300 training samples and an equal amount of test samples distributed across 5 classes: *animal*, *human*, *plane*, *truck*, *car*. We normalized all NORB images following Miyato et al. (2015) using image pairs of 32x32 resulting in a vectorized input of $\dim(x) = 2048$. The labeled subsets consisted of 1000 evenly distributed labeled samples. The batch size for SVHN was 2000 and for NORB 200, where half of the batch was labeled samples. To avoid the phenomenon on modeling discretized values with a real-valued estimation (Uria et al., 2013), we added uniform noise between 0 and 1 to each pixel value. We normalized the NORB dataset by 256 and the SVHN dataset by the standard deviation on each color channel. Both datasets were assumed Gaussian distributed for the generative models $p_\theta(x|\cdot)$.

4. Results

In this section we present two toy examples that shed light on how the auxiliary variables improve the distribution fit. Thereafter we investigate the unsupervised generative log-likelihood performance followed by semi-supervised classification performance on several benchmark datasets. We demonstrate state-of-the-art performance and show that adding auxiliary variables increase both classification performance and convergence speed (cf. Sec. 3 for details).

4.1. Beyond Gaussian latent distributions

In variational auto-encoders the inference model $q_\phi(z|x)$ is parameterized as a fully factorized Gaussian. We demonstrate that the auxiliary model can fit complicated posterior distributions for the latent space. To do this we consider the 2D potential model $p(z) = \exp(U(z))/Z$ (Rezende and Mohamed, 2015) that leads to the bound

$$\log Z \geq \mathbb{E}_{q_\phi(a,z)}\left[\log \frac{\exp(U(z))p_\theta(a|z)}{q_\phi(a)q_\phi(z|a)}\right]. \qquad (18)$$

Fig. 2a shows the true posterior and Fig. 2b shows a density plot of z samples from $a \sim q_\phi(a)$ and $z \sim q_\phi(z|a)$ from a trained ADGM. This is similar to the findings of Rezende and Mohamed (2015) in which they demonstrate that by using normalizing flows they can fit complicated posterior distributions. The most frequent solution found in optimization is not the one shown, but one where Q fits only one of the two equivalent modes. The one and two mode solution will have identical values of the bound so it is to be expected that the simpler single mode solution will be easier to infer.

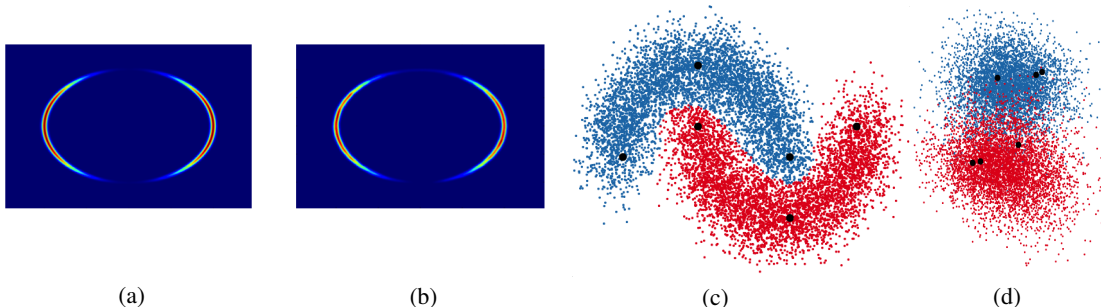| (a) | (b) | (c) | (d) |

Figure 2. (a) True posterior of the prior $p(z)$. (b) The approximation $q_\phi(z|a)q_\phi(a)$ of the ADGM. (c) Prediction on the half-moon data set after 10 epochs with only 3 labeled data points (black) for each class. (d) PCA plot on the 1st and 2nd principal component of the corresponding auxiliary latent space.

4.2. Semi-supervised learning on two half-moons

To exemplify the power of the ADGM for semi-supervised learning we have generated a 2D synthetic dataset consisting of two half-moons (top and bottom), where $(x_\text{top}, y_\text{top}) = (cos([0, \pi]), sin([0, \pi]))$ and $(x_\text{bottom}, y_\text{bottom}) = (1 - cos([0, \pi]), 1 - sin([0, \pi]) - 0.5)$, with added Gaussian noise. The training set contains 1e4 samples divided into batches of 100 with 3 labeled data points in each class and the test set contains 1e4 samples. A good semi-supervised model will be able to learn the data manifold for each of the half-moons and use this together with the limited labeled information to build the classifier.

The ADGM converges close to 0% classification error in 10 epochs (cf. Fig. 2c), which is much faster than an equivalent model without the auxiliary variable that converges in more than 100 epochs. When investigating the auxiliary variable we see that it finds a discriminating internal representation of the data manifold and thereby aids the classifier (cf. Fig. 2d).

4.3. Generative log-likelihood performance

We evaluate the generative performance of the unsupervised auxiliary model, AVAE, using the MNIST dataset. The inference and generative models are defined as

$$q_\phi(a, z|x) = q_\phi(a_1|x)q_\phi(z_1|a_1, x) \qquad (19)$$
$$\prod_{i=2}^{L} q_\phi(a_i|a_{i-1}, x)q_\phi(z_i|a_i, z_{i-1}) \, ,$$
$$p_\theta(x, a, z) = p_\theta(x|z_1)p(z_L)p_\theta(a_L|z_L) \qquad (20)$$
$$\prod_{i=1}^{L-1} p_\theta(z_i|z_{i+1})p_\theta(a_i|z_{\geq i}) \, .$$

where L denotes the number of stochastic layers.

We report the lower bound from Eq. (2) for 5000 importance weighted samples and use the same training and parameter settings as in Sønderby et al. (2016) with warm-

up[2], batch normalization and 1 Monte Carlo and IW sample for training.

	$\leq \log p(x)$
VAE+NF, L=1 (REZENDE AND MOHAMED, 2015)	-85.10
IWAE, L=1, IW=1 (BURDA ET AL., 2015)	-86.76
IWAE, L=1, IW=50 (BURDA ET AL., 2015)	-84.78
IWAE, L=2, IW=1 (BURDA ET AL., 2015)	-85.33
IWAE, L=2, IW=50 (BURDA ET AL., 2015)	-82.90
VAE+VGP, L=2 (TRAN ET AL., 2015)	-81.90
LVAE, L=5, IW=1 (SØNDERBY ET AL., 2016)	-82.12
LVAE, L=5, FT, IW=10 (SØNDERBY ET AL., 2016)	-81.74
AUXILIARY VAE (AVAE), L=1, IW=1	-84.59
AUXILIARY VAE (AVAE), L=2, IW=1	-82.97

Table 1. Unsupervised test log-likelihood on permutation invariant MNIST for the normalizing flows VAE (VAE+NF), importance weighted auto-encoder (IWAE), variational Gaussian process VAE (VAE+VGP) and Ladder VAE (LVAE) with FT denoting the finetuning procedure from Sønderby et al. (2016), IW the importance weighted samples during training, and L the number of stochastic latent layers $z_1, .., z_L$.

We evaluate the negative log-likelihood for the 1 and 2 layered AVAE. We found that warm-up was crucial for activation of the auxiliary variables. Table 1 shows log-likelihood scores for the permutation invariant MNIST dataset. The methods are not directly comparable, except for the Ladder VAE (LVAE) (Sønderby et al., 2016), since the training is performed differently. However, they give a good indication on the expressive power of the auxiliary variable model. The AVAE is performing better than the VAE with normalizing flows (Rezende and Mohamed, 2015) and the importance weighted auto-encoder with 1 IW sample (Burda et al., 2015). The results are also comparable to the Ladder VAE with 5 latent layers (Sønderby et al., 2016) and variational Gaussian process VAE (Tran et al., 2015). As shown in Burda et al. (2015) and Sønderby et al. (2016) increasing the IW samples and annealing the learning rate will likely increase the log-likelihood.

---

[2]Temperature on the KL-divergence going from 0 to 1 within the first 200 epochs of training.

	MNIST 100 LABELS	SVHN 1000 LABELS	NORB 1000 LABELS
M1+TSVM (KINGMA ET AL., 2014)	11.82% ($\pm$0.25)	55.33% ($\pm$0.11)	18.79% ($\pm$0.05)
M1+M2 (KINGMA ET AL., 2014)	3.33% ($\pm$0.14)	36.02%($\pm$0.10)	-
VAT (MIYATO ET AL., 2015)	2.12%	24.63%	9.88%
LADDER NETWORK (RASMUS ET AL., 2015)	1.06% ($\pm$0.37)	-	-
AUXILIARY DEEP GENERATIVE MODEL (ADGM)	**0.96**% ($\pm$0.02)	22.86%	10.06% ($\pm$0.05)
SKIP DEEP GENERATIVE MODEL (SDGM)	1.32% ($\pm$0.07)	**16.61**% ($\pm$0.24)	**9.40**% ($\pm$0.04)

Table 2. Semi-supervised test error % benchmarks on MNIST, SVHN and NORB for randomly labeled and evenly distributed data points. The lower section demonstrates the benchmarks of the contribution of this article.

4.4. Semi-supervised benchmarks

MNIST EXPERIMENTS

Table 2 shows the performance of the ADGM and SDGM on the MNIST dataset. The ADGM's convergence to around 2% is fast (around 200 epochs), and from that point the convergence speed declines and finally reaching 0.96% (cf. Fig. 5). The SDGM shows close to similar performance and proves more stable by speeding up convergence, due to its more advanced generative model. We achieved the best results on MNIST by performing multiple Monte Carlo samples for $a \sim q_\phi(a|x)$ and $z \sim q_\phi(z|a, y, x)$.

A good explorative estimate of the models ability to comprehend the data manifold, or in other words be as close to the posterior distribution as possible, is to evaluate the generative model. In Fig. 3a we show how the SDGM, trained on *only* 100 labeled data points, has learned to separate style and class information. Fig 3b shows random samples from the generative model.
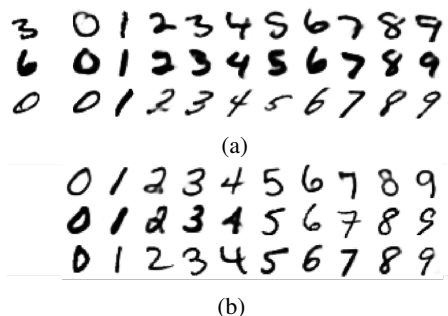
(a)

(b)

Figure 3. MNIST analogies. (a) Forward propagating a data point x (left) through $q_\phi(z|a, x)$ and generate samples $p_\theta(x|y_j, z)$ for each class label y_j (right). (b) Generating a sample for each class label from random generated Gaussian noise; hence with no use of the inference model.

Fig. 4a demonstrate the information contribution from the stochastic unit a_i and z_j (subscripts i and j denotes a unit) in the SDGM as measured by the average over the test set of the KL-divergence between the variational distribution and the prior. Units with little information content will be close to the prior distribution and the KL-divergence term

will thus be close to 0. The number of clearly activated units in z and a is quite low ~ 20, but there is a tail of slightly active units, similar results have been reported by Burda et al. (2015). It is still evident that we have information flowing through both variables though. Fig. 2d and 4b shows clustering in the auxiliary space for both the ADGM and SDGM respectively.
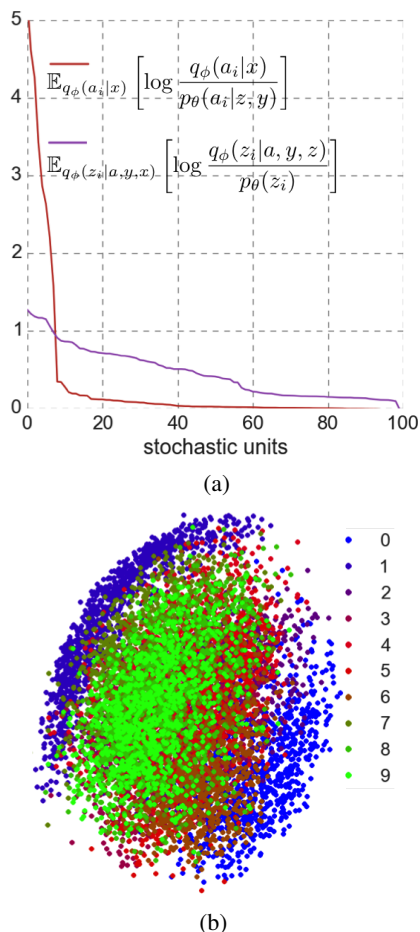
(a)

(b)

Figure 4. SDGM trained on 100 labeled MNIST. (a) The KL-divergence for units in the latent variables a and z calculated by the difference between the approximated value and its prior. (b) PCA on the 1st and 2nd principal component of the auxiliary latent space.
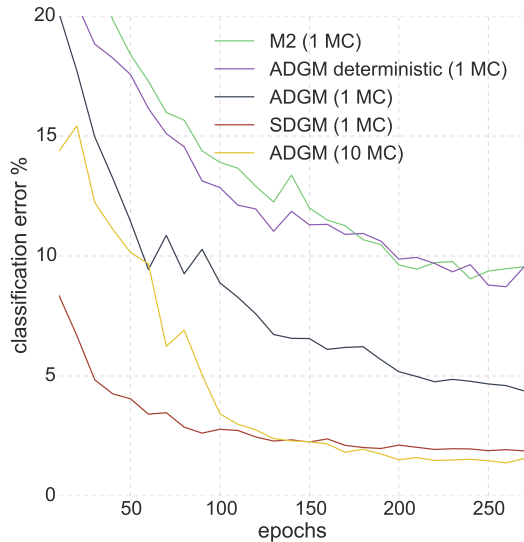
Figure 5. 100 labeled MNIST classification error % evaluated every 10 epochs between equally optimized SDGM, ADGM, M2 (Kingma et al., 2014) and an ADGM with a deterministic auxiliary variable.

In order to investigate whether the stochasticity of the auxiliary variable a or the network depth is essential to the models performance, we constructed an ADGM with a deterministic auxiliary variable. Furthermore we also implemented the M2 model of Kingma et al. (2014) using the exact same hyperparameters as for learning the ADGM. Fig. 5 shows how the ADGM outperforms both the M2 model and the ADGM with deterministic auxiliary variables. We found that the convergence of the M2 model was highly unstable; the one shown is the best obtained.

SVHN & NORB EXPERIMENTS

From Table 2 we see how the SDGM outperforms VAT with a relative reduction in error rate of more than 30% on the SVHN dataset. We also tested the model performance, when we omitted the SVHN *extra* set from training. Here we achieved a classification error of 29.82%. The improvements on the NORB dataset was not as significant as for SVHN with the ADGM being slightly worse than VAT and the SDGM being slightly better than VAT.

On SVHN the model trains to around 19% classification error in 100 epochs followed by a decline in convergence speed. The NORB dataset is a significantly smaller dataset and the SDGM converges to around 12% in 100 epochs. We also trained the NORB dataset on single images as opposed to image pairs (half the dataset) and achieved a classification error around 13% in 100 epochs.

For Gaussian input distributions, like the image data of SVHN and NORB, we found the SDGM to be more stable than the ADGM.

5. Discussion

The ADGM and SDGM are powerful deep generative models with relatively simple neural network architectures. They are trainable end-to-end and since they follow the principles of variational inference there are multiple improvements to consider for optimizing the models like using the importance weighted bound or adding more layers of stochastic variables. Furthermore we have only proposed the models using a Gaussian latent distribution, but the model can easily be extended to other distributions (Ranganath et al., 2014; 2015).

One way of approaching the stability issues of the ADGM, when training on Gaussian input distributions x is to add a *temperature* weighting between discriminative and stochastic learning on the KL-divergence for a and z when estimating the variational lower bound (Sønderby et al., 2016). We find similar problems for the Gaussian input distributions in van den Oord et al. (2016), where they restrict the dataset to ordinal values in order to apply a softmax function for the output of the generative model $p(x|\cdot)$. This discretization of data is also a possible solution. Another potential stabilizer is to add batch normalization (Ioffe and Szegedy, 2015) that will ensure normalization of each output batch of a fully connected hidden layer.

A downside to the semi-supervised variational framework is that we are summing over all classes in order to evaluate the variational bound for unlabeled data. This is a computationally costly operation when the number of classes grow. In this sense, the Ladder network has an advantage. A possible extension is to sample y when calculating the unlabeled lower bound $-\mathcal{U}(x_u)$, but this may result in gradients with high variance.

The framework is implemented with fully connected layers. VAEs have proven to work well with convolutional layers so this could be a promising step to further improve the performance. Finally, since we expect that the variational bound found by the auxiliary variable method is quite tight, it could be of interest to see whether the bound for $p(x, y)$ may be used for classification in the Bayes classifier manner $p(y|x) \propto p(x, y)$.

6. Conclusion

We have introduced a novel framework for making the variational distributions used in deep generative models more expressive. In two toy examples and the benchmarks we investigated how the framework uses the auxiliary variables to learn better variational approximations. Finally we have demonstrated that the framework gives state-of-the-art performance in a number of semi-supervised benchmarks and is trainable end-to-end.

A. Auxiliary model specification

In this appendix we study the theoretical optimum of the auxiliary variational bound found by functional derivatives of the variational objective. In practice we will resort to restricted deep network parameterized distributions. But this analysis nevertheless shed some light on the properties of the optimum. Without loss of generality we consider only auxiliary a and latent z: $p(a, z) = p(z)p(a|z)$, $p(z) = f(z)/Z$ and $q(a, z) = q(z|a)q(a)$. The results can be extended to the full semi-supervised setting without changing the overall conclusion. The variational bound for the auxiliary model is

$$\log Z \geq \mathbb{E}_{q(a,z)} \left[\log \frac{f(z)p(a|z)}{q(z|a)q(a)} \right] . \qquad (21)$$

We can now take the functional derivative of the bound with respect to $p(a|z)$. This gives the optimum $p(a|z) = q(a, z)/q(z)$, which in general is intractable because it requires marginalization: $q(z) = \int q(z|a)q(a)da$.

One may also restrict the generative model to an uninformed a-model: $p(a, z) = p(z)p(a)$. Optimizing with respect to $p(a)$ we find $p(a) = q(a)$. When we insert this solution into the variational bound we get

$$\int q(a) \, \mathbb{E}_{q(z|a)} \left[\log \frac{f(z)}{q(z|a)} \right] da . \qquad (22)$$

The solution to the optimization with respect to $q(a)$ will simply be a δ-function at the value of a that optimizes the variational bound for the z-model. So we fall back to a model for z without the auxiliary as also noted by Agakov and Barber (2004).

We have tested the uninformed auxiliary model in semi-supervised learning for the benchmarks and we got competitive results for MNIST but not for the two other benchmarks. We attribute this to two factors: in semi-supervised learning we add an additional classification cost so that the generic form of the objective is

$$\log Z \geq \mathbb{E}_{q(a,z)} \left[\log \frac{f(z)p(a)}{q(z|a)q(a)} + g(a) \right] , \qquad (23)$$

we keep $p(a)$ fixed to a zero mean unit variance Gaussian and we use deep iid models for $f(z)$, $q(z|a)$ and $q(a)$. This taken together can lead to at least a local optimum which is different from the collapse to the pure z-model.

B. Variational bounds

In this appendix we give an overview of the variational objectives used. The generative model $p_\theta(x, a, y, z)$ for the *Auxiliary Deep Generative Model* and the *Skip Deep Generative Model* are defined as

ADGM:
$$p_\theta(x, a, y, z) = p_\theta(x|y, z)p_\theta(a|x, y, z)p(y)p(z) . \qquad (24)$$

SDGM:
$$p_\theta(x, a, y, z) = p_\theta(x|a, y, z)p_\theta(a|x, y, z)p(y)p(z) . \qquad (25)$$

The lower bound $-\mathcal{L}(x, y)$ on the labeled log-likelihood is defined as

$$\log p(x, y) = \log \int_a \int_z p_\theta(x, y, a, z)dzda \qquad (26)$$

$$\geq \mathbb{E}_{q_\phi(a,z|x,y)} \left[\log \frac{p_\theta(x, y, a, z)}{q_\phi(a, z|x, y)} \right] \equiv -\mathcal{L}(x, y) ,$$

where $q_\phi(a, z|x, y) = q_\phi(a|x)q_\phi(z|a, y, x)$. We define the function $f(\cdot)$ to be $f(x, y, a, z) = \log \frac{p_\theta(x,y,a,z)}{q_\phi(a,z|x,y)}$. In the lower bound for the unlabeled data $-\mathcal{U}(x)$ we treat the discrete y[3] as a latent variable. We rewrite the lower bound in the form of Kingma et al. (2014):

$$\log p(x) = \log \int_a \sum_y \int_z p_\theta(x, y, a, z)dzda$$

$$\geq \mathbb{E}_{q_\phi(a,y,z|x)} \left[f(\cdot) - \log q_\phi(y|a, x) \right] \qquad (27)$$

$$= \mathbb{E}_{q_\phi(a|x)} \sum_y q_\phi(y|a, x)\mathbb{E}_{q_\phi(z|a,x)} \left[f(\cdot) \right] +$$

$$\mathbb{E}_{q_\phi(a|x)} \Big[\underbrace{- \sum_y q_\phi(y|a, x) \log q_\phi(y|a, x)}_{\mathcal{H}(q_\phi(y|a,x))} \Big]$$

$$= \mathbb{E}_{q_\phi(a|x)} \Big[\sum_y q_\phi(y|a, x)\mathbb{E}_{q_\phi(z|a,x)} \left[f(\cdot) \right] +$$

$$\mathcal{H}(q_\phi(y|a, x)) \Big]$$

$$\equiv -\mathcal{U}(x) ,$$

where $\mathcal{H}(\cdot)$ denotes the entropy. The objective function of $-\mathcal{L}(x, y)$ and $-\mathcal{U}(x)$ are given in Eq. (12) and Eq. (13).

---

[3]y is assumed to be multinomial but the model can easily be extended to different distributions.

Acknowledgements

References

Agakov, F. and Barber, D. (2004). An Auxiliary Variational Method. In *Neural Information Processing*, volume 3316 of *Lecture Notes in Computer Science*, pages 561–566. Springer Berlin Heidelberg.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. In *Deep Learning and Unsupervised Feature Learning, workshop at Neural Information Processing Systems.*

Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Importance Weighted Autoencoders. *arXiv preprint arXiv:1509.00519.*

Dieleman, S., Schlter, J., Raffel, C., Olson, E., Sønderby, S. K., Nouri, D., van den Oord, A., and and, E. B. (2015). Lasagne: First release.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10).*, pages 249–256.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference of Machine Learning*, pages 448–456.

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980.*

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. (2014). Semi-Supervised Learning with Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, pages 3581–3589.

Kingma, Diederik P; Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114.*

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2278–2324.

LeCun, Y., Huang, F. J., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 97–104.

Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., and Ishii, S. (2015). Distributional Smoothing with Virtual Adversarial Training. *arXiv preprint arXiv:1507.00677.*

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Deep Learning and Unsupervised Feature Learning, workshop at Neural Information Processing Systems 2011.*

Ranganath, R., Tang, L., Charlin, L., and Blei, D. M. (2014). Deep exponential families. *arXiv preprint arXiv:1411.2581.*

Ranganath, R., Tran, D., and Blei, D. M. (2015). Hierarchical variational models. *arXiv preprint arXiv:1511.02386.*

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3532–3540.

Rezende, D. J. and Mohamed, S. (2015). Variational Inference with Normalizing Flows. In *Proceedings of the International Conference of Machine Learning*, pages 1530–1538.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv preprint arXiv:1401.4082.*

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). Ladder variational autoencoders. *arXiv preprint arXiv:1602.02282.*

Tran, D., Ranganath, R., and Blei, D. M. (2015). Variational Gaussian process. *arXiv preprint arXiv:1511.06499.*

Uria, B., Murray, I., and Larochelle, H. (2013). Rnade: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, pages 2175–2183.

Valpola, H. (2014). From neural pca to deep unsupervised learning. *arXiv preprint arXiv:1411.7783.*

van den Oord, A., Nal, K., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759.*