Journal of Machine Learning Research 1–19

Hamiltonian Variational Auto-Encoder

Daniil Selikhanovych Daniil.Selihanovich@skoltech.ru
Aleksandr Artemenkov Aleksandr.Artemenkov@skoltech.ru
Igor Karpikov Igor.Karpikov@skoltech.ru
Skolkovo Institute of Science and Technology

Abstract

Hamiltonian Variational Auto-Encoder (HVAE) is a modification of Variational Auto-Encoder (VAE) proposed in Caterini et al. (2018). Let z be the latent variable, as by VAE model. HVAE differs from VAE by introducing momentum variable $\rho \sim \mathcal{N}(z|0, \sigma^2 I)$, and applying to (z, ρ) K Leapfrog steps of the Hamiltonian dynamics. Such a transform can be seen as applying to (z, ρ) K deterministic Markov kernels. It can be shown, that these kernels are optimal in some sense (see in the report).

We have implemented HVAE in PyTorch and compared it with different models on artificial dataset and MNIST. We found that HVAE is a competitive alternative to current state-of-the-art algorithms. We have proposed our own modification of HVAE, which, unfortunately, had worse performance, compared to the original.

1. Introduction

1.1. Theory

Consider the ordinary setting, where x is the observed variable and z - latent. The prior distribution of z is given by $p(z)$ and we are interested in $p(x)$, which is unknown. The posterior $p(z \mid x)$ is almost always intractable, thus, we have to consider some approximation $q(z \mid x)$. Assume we have an unbiased, w.r.t. $q(z \mid x)$, estimate of $p(x)$, denote $\hat{p}_z(x)$, which depends on both x and z and:

$$\int \hat{p}_z(x) q(z \mid x) dz = p(x).$$

For a special choice of $\hat{p}_z(x)$ the ELBO is given by:

$$\mathcal{L}(\theta, x) := \int q(z \mid x) \log \hat{p}_z(x) dz \leq \log p(x),$$

due to Jensen inequality. We want to maximize $\mathcal{L}(\theta, x)$ and It is known to have negative dependence on the variance of $\hat{p}_z(x)$, see Burda et al. (2015). Thus, we are to minimize the variance of $\hat{p}_z(x)$. If we can not compute $\mathcal{L}(\theta, x)$ analytically, we need to have an unbiased, low-variance estimate of it's gradient for optimization purposes, so, the desirable property would be:

$$\mathbb{E}_q[\nabla_\theta \log \hat{p}_z(x)] = \nabla_\theta \mathcal{L}(\theta, x),$$

which does not hold in general.

Consider the following type of $\hat{p}_z(x)$:

$$\hat{p}_z(x) = \frac{p(x, z_K) \prod_{k=0}^{K-1} r_k(z_k \mid z_{k+1})}{q_0(z_0) \prod_{k=1}^{K} q_k(z_k \mid z_{k-1})},$$

where q_k and r_k are forward and reverse Markov kernels. Denote $z = (z_0, \ldots, z_K)$, $\tilde{z} = (z_0, \ldots, z_{K-1})$. It is indeed unbiased estimate:

$$\mathbb{E}[\hat{p}_z(x)] = \int \frac{p(x, z_K) \prod_{k=0}^{K-1} r_k(z_k \mid z_{k+1})}{q_0(z_0) \prod_{k=1}^{K} q_k(z_k \mid z_{k-1})} q_0(z_0) \prod_{k=1}^{K} q_k(z_k \mid z_{k-1}) dz =$$

$$\int p(x, z_K) \int \prod_{k=0}^{K-1} r_k(z_k | z_{k+1}) d\tilde{z} dz_K = \int p(x, z_K) dz_K = p(x).$$

Proposition: for any sequence of forward Markov kernels $\{q_k\}_{k=0}^K$ we can find the sequence of inverse kernels $\{r_k\}_{k=0}^{K-1}$ such that the variance is minimized. With this choice the unbiased estimate of $\hat{p}_z(x)$ is given by:

$$\hat{p}_z(x) = \frac{p(x, z_K)}{q_K(z_K)},$$

where $q_K(z_K)$ is the marginal density of z_k under forward Markov dynamics.

Proof: express $q_0(z_0) \prod_{k=1}^{K} q_k(z_k \mid z_{k-1}) = q(z_0, \ldots, z_K)$ as we have Markov chain. Next, $q(z_0, \ldots, z_K) = q_K(z_K) q(z_0, \ldots, z_{K-1} \mid z_K)$. Note that $q := q(z_0, \ldots, z_{K-1} \mid z_K)$ and $r := \prod_{k=0}^{K-1} r_k(z_k \mid z_{k+1})$ are probability densities and select r with the same support as q, thus:

$$\mathbb{V}\big(\hat{p}_z(x)\big) = \mathbb{E}\big(\hat{p}_z^2(x)\big) - \mathbb{E}^2\big(\hat{p}_z(x)\big) = \int \frac{p^2(x, z_K)}{q_K(z_K)} \frac{r^2}{q} dz - p^2(x) \propto^+ \int \frac{p^2(x, z_K)}{q_K(z_K)} \int \frac{r^2}{q} d\tilde{z} dz_K,$$

but:

$$\int \frac{r^2}{q} d\tilde{z} = \mathbb{E}_r \left[\frac{r}{q}\right] \geq \mathbb{E}_r \left[1 + \log \frac{r}{q}\right] = 1 - \mathbb{E}_r \left[\log \frac{q}{r}\right] \geq 1 - \mathbb{E}_r \left[1 - \frac{q}{r}\right] = \mathbb{E}_r \left[\frac{q}{r}\right] = 1,$$

as $x - 1 \geq \log(x), \forall x > 0$. Thus, the minimum of variance w.r.t. r equals:

$$\int \frac{p^2(x, z_K)}{q_K(z_K)} dz_K,$$

and attained at point $r = q$. This ends the proof.

The problem here is that $q_K(z_K)$ not always admits a closed-form expression, so, we can either choose approximation or use $\{q_k\}$ for which we can compute $q_K(z_K)$.

For HVAE we introduce additional momentum variable with the following assumption on the joint density:

$$\bar{p}(x, z, \rho) := p(x, z)N(\rho \mid 0, I).$$

If use deterministic Markov kernels and $q_K(z_k, \rho_k)$ can be analytically calculated as follows:

$$q_k(z_k, \rho_k \mid z_{k-1}, \rho_{k-1}) = \delta_{\Phi_k(z_{k-1}, \rho_{k-1})}(z_k, \rho_k)$$

$$q_K(z_K, \rho_K) = q_0(z_0, \rho_0) \prod_{k=1}^{K} |\det \nabla \Phi_k(z_k, \rho_k)|^{-1}.$$

For HVAE each transform $\Phi_k : (z_{k-1}, \rho_{k-1}) \to (z_k, \rho_k)$ is simply one Leapfrog step in Hamiltonian dynamics, with momentum variable being multiplied by some positive tempering constant α_k at the end:

$$\rho_{k-1} = \rho_{k-1} - \frac{\varepsilon}{2}\nabla_{z_{k-1}}\big(-\log p(x, z_{k-1})\big),$$

$$z_k = \rho_{k-1} + \varepsilon\rho_{k-1},$$

$$\rho_k = \alpha_k\Big(\rho_{k-1} - \frac{\varepsilon}{2}\nabla_{z_k}\big(-\log p(x, z_k)\big)\Big).$$

With this the Jacobian is given by:

$$\prod_{k=1}^{K} |\det \nabla \Phi_k(z_k, \rho_k)| = \prod_{k=1}^{K} \alpha_k^l = \beta^{\frac{l}{2}},\ l = \dim(z_k)$$

.

Afterall:

$$\hat{p}(x) = \frac{\bar{p}(x, z_K, \rho_K)}{q_K(z_K, \rho_K)} = \frac{p(x, z_K)\mathcal{N}(\rho_K|0, I)}{q_0(z_0, \rho_0)}\beta_0^{\frac{l}{2}},$$

Also we assume:

$$q_0(z_0, \rho_0) = q_0(z_0)\mathcal{N}(\rho_0 \mid 0, \beta_0^{-1}I).$$

Make a substitution: $\rho_0 = \beta_0^{-\frac{1}{2}}\gamma_0$, thus:

$$\hat{p}(x) = \frac{p(x, z_K)\mathcal{N}(\rho_K \mid 0, I)}{q_0(z_0)\mathcal{N}(\gamma_0 \mid 0, I)}.$$

If we do not optimize w.r.t. parameters of q_0 we have that:

$$\mathbb{E}\left[\nabla_\theta \log\left[\frac{p(x, z_K)\mathcal{N}(\rho_K \mid 0, I)}{q_0(z_0)\mathcal{N}(\gamma_0 \mid 0, I)}\right]\right] = \nabla_\theta \mathcal{L}(\theta, x),$$

however, if we do, we can make additional reparametrization (see HVAE + IAF section) which is problem-specific.

1.2. Related works

Since VAE is not always flexible enough to express complex distributions, various alternatives were proposed in resent times. These methods, mostly, boil down to applying some transform to the samples of VAE. NF's are typically used for this task (Rezende and Mohamed, 2015), in which samples are deterministically evolve through a series of parametrized invertible transforms. However such approach does not explicitly use information about target posterior, which may be an ommision. In contrast, Hamiltonian Variational Inference (HVI, Salimans et al. (2015)), stochastically evolves the base samples according to the HMC and uses information about target posterior, but relies on defining reverse kernels, which are suboptimal. HVAE is aimed to combine advantages of both methods and can be considered as a target-informed normalizing flow.

Another important alternative is Importance Weighted Auto-Encoder (IWAE, Burda et al. (2015)), which we used in our experiments for comparison.

2. Experiments, models and results

According to original paper, we consider two datasets. We first test HVAE on an example with a tractable full log likelihood (where no neural networks are needed), and then perform larger-scale tests on the MNIST dataset. We also proposed our own modification of HVAE for Auto-Encoder setting and tested it on MNIST. Code of the project is available here: https://github.com/Daniil-Selikhanovych/h-vae.

2.1. Gaussian Model

2.1.1. Model

The generative model that we will consider first is a Gaussian likelihood with an offset and a Gaussian prior on the mean, given by

$$\begin{aligned} z &\sim \mathcal{N}(0, I_d) \\ x_i \mid z &\sim \mathcal{N}(z + \Delta, \mathbf{\Sigma}) \quad \text{independently}, \quad i \in \overline{1, N} \end{aligned}$$

where $\mathbf{\Sigma}$ is constrained to be diagonal. We will write $\mathcal{D} \equiv \{x_1, \ldots, x_N\}$ to denote an observed dataset under this model, where each $x_i \in \mathcal{X} \subseteq \mathbb{R}^d$. The goal of the problem is to learn the model parameters $\theta \equiv \{\mathbf{\Sigma}, \Delta\}$, where $\mathbf{\Sigma} = \operatorname{diag}\left(\sigma_1^2, \ldots, \sigma_d^2\right)$ and $\Delta \in \mathbb{R}^d$.

Here, we have only one latent variable generating the entire set of data. Thus, our variational lower bound is now given by

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi; \mathcal{D}) := \mathbf{E}_{z \sim q_{\theta,\phi}(\cdot \mid \mathcal{D})} \left[\log p_\theta(\mathcal{D}, z) - \log q_{\theta,\phi}(z \mid \mathcal{D})\right] \leq \log p_\theta(\mathcal{D})$$

for the variational posteroir approximation $q_{\theta,\phi}(\cdot \mid \mathcal{D})$

From the model, we see that the logarithm of the target is given by

$$\log p_\theta(\mathcal{D}, z) = \sum_{i=1}^{N} \log \mathcal{N}\left(x_i \mid z + \Delta, \mathbf{\Sigma}\right) + \log \mathcal{N}\left(z \mid 0, I_d\right).$$

2.1.2. Initialization and true setup for delta and sigma parameters

In Gaussian model we learn for numerical stability matrix

$$\log \mathbf{\Sigma}_0 \stackrel{\text{def}}{=} \operatorname{diag}(\log \sigma_1, \ldots, \log \sigma_d)$$

and vector Δ. We initialize for all models the following values of Δ and $\log \mathbf{\Sigma}_0$:

$$\Delta_0 = 0_d, \quad \log \mathbf{\Sigma}_0 \stackrel{\text{def}}{=} \operatorname{diag}(\log \sigma_1, \ldots, \log \sigma_d) = 3 \cdot 1_d,$$

where 0_d denotes the vector of d zeros and 1_d denotes the vector of d ones.

We set our true offset vector to be

$$\Delta = \frac{1}{5} \times \left(-\frac{d-1}{2}, -\frac{d+1}{2}, \ldots, \frac{d-3}{2}, \frac{d-1}{2} \right),$$

and our scale parameters to range quadratically from $\sigma_1 = 1$, reaching a minimum at $\sigma_{(d+1)/2} = 0.1$, and increasing back to $\sigma_d = 1$:

$$\operatorname{diag}(\sigma_1, \ldots, \sigma_d) = \frac{90}{(d-1)^2} \operatorname{diag}(\Delta \odot \Delta) + 0.1 I_d,$$

where $\odot$ denotes component-wise product of vectors.

2.1.3. HVAE

The potential, given by $U_\theta(z \mid \mathcal{D}) = -\log p_\theta(\mathcal{D}, z)$, has gradient

$$\nabla_z U_\theta(z \mid \mathcal{D}) = z + N \mathbf{\Sigma}^{-1}(z + \Delta - \bar{x}), \quad \bar{x} \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} x_i.$$

For this example, we will use a HVAE with variational prior equal to the true prior, i.e. $q_0 = \mathcal{N}(0, I_d)$, and fixed tempering. The set of variational parameters here is $\phi \equiv \{\varepsilon, \beta_0\}$, where $\varepsilon \in \mathbb{R}^d$ contains the per-dimension leapfrog stepsizes and $\beta_0 \in (0, 1)$ is the initial inverse temperature. For numerical stability we learn $\operatorname{logit}(\varepsilon)$, where $\operatorname{logit}(\varepsilon)$ is defined by equation:

$$\frac{\varepsilon}{\varepsilon_{max}} = \frac{1}{1 + \exp\Big(-\operatorname{logit}(\varepsilon)\Big)} \Rightarrow \operatorname{logit}(\varepsilon) \stackrel{\text{def}}{=} \log\left(\frac{\varepsilon}{\varepsilon_{max} - \varepsilon}\right)$$

and $\log T(\beta_0)$, where $\log T(\beta_0)$ is defined by equation:

$$\frac{1}{\sqrt{\beta_0}} = 1 + \exp\Big(\log T(\beta_0)\Big) \Rightarrow \log T(\beta_0) \stackrel{\text{def}}{=} \log\left(\frac{1 - \sqrt{\beta_0}}{\sqrt{\beta_0}}\right).$$

For all HVAE experiments with tempering we used

$$\varepsilon_{max} = 0.5 \times 1_d$$

and initialize

$$\varepsilon_0 = 0.005 \times 1_d, \quad \log T_0(\beta_0) = \log(4).$$

For HVAE experiments without tempering we fixed $\beta_0 = 1$ and learned only ε parameter.

2.1.4. Variational Bayes

For Variational-Bayes scheme we use mean-field approximate posterior

$$q_{\phi_V}(z \mid \mathcal{D}) = \mathcal{N}(z \mid \mu_Z, \mathbf{\Sigma}_Z),$$

where $\mathbf{\Sigma}_Z$ is diagonal and $\phi_V \equiv \{\mu_Z, \mathbf{\Sigma}_Z\}$ denotes the set of learned variational parameters. While training we also learn for numerical stability matrix

$$\log \mathbf{\Sigma}_Z \stackrel{\text{def}}{=} \operatorname{diag}\left(\log \sqrt{\mathbf{\Sigma}_Z[1,1]}, \ldots, \log \sqrt{\mathbf{\Sigma}_Z[d,d]}\right),$$

where $\mathbf{\Sigma}_Z[i,j]$ denotes (i,j) element of matrix $\mathbf{\Sigma}_Z$. and initialize for all models the following values of μ_Z and $\log \mathbf{\Sigma}_0$:

$$\mu_Z = 0_d, \quad \log \mathbf{\Sigma}_Z \stackrel{\text{def}}{=} \operatorname{diag}\left(\log \sqrt{\mathbf{\Sigma}_Z[1,1]}, \ldots, \log \sqrt{\mathbf{\Sigma}_Z[d,d]}\right) = 1_d.$$

2.1.5. Normalizing Flows

We compare the HVAE with 2 types of Normalizing flows: planar normalizing flows (NMF) (Rezende and Mohamed, 2015) and inverse autoregressive flows (IAF) (Kingma et al., 2016). In the original paper authors compared their results only with the planar normalizing flows and we showed that for considered Gaussian models IAF outperform NMF both in learning of Δ and Σ parameters. For the both types of normalizing flow the variational prior here is also set to the true prior $\mathcal{N}(0, I_d)$ as in HVAE above.

For IAF implementation we used our code from homework 4. For each dimensionality d we learn IAF with the hidden dimensionality $d + 100$ and with depth $K = 1$.

For planar normalizing flows we consider the following family of transformations:

$$f(\mathbf{z}_k) = \mathbf{z} + \mathbf{u} \cdot \tanh(\mathbf{w}^T \mathbf{z}_{k-1} + b).$$

The log variational posterior $\log q_{\phi_N}(z \mid \mathcal{D})$ is given by

$$\log q_{\phi_N}(z \mid \mathcal{D}) = \ln q_K(\mathbf{z}_K) = \ln q_0(\mathbf{z}) - \sum_{k=1}^{K} \ln \left| 1 + \mathbf{u}_k^\top \psi_k(\mathbf{z}_{k-1}) \right|,$$

where

$$\psi(\mathbf{z}) = h'\left(\mathbf{w}^\top \mathbf{z} + b\right)\mathbf{w}, \quad h(z) = \tanh(z) \to h'(z) = 1 - \tanh^2(z).$$

Here the learned variational parameters are $\phi_N \equiv \{\mathbf{u}, \mathbf{w}, b\} \in \mathbb{R}^{2d+1}$. We initialized d components of $\mathbf{w}$ as independent truncated normal continuous random variables to the interval $[-2, 2]$ with scale parameter 0.1 and fixed random seed = 12345. For b parameter we initialized the value 0.1. For numerical stability we learned not $\mathbf{u}$, but $\mathbf{u}_{reparametrized}$, where

$$\mathbf{u} = \mathbf{u}_{reparametrized} - \left(-1 + \text{Softplus}\left(\mathbf{u}_{reparametrized}^T \mathbf{w}\right) - \mathbf{u}_{reparametrized}^T \mathbf{w}\right) \times \frac{\mathbf{w}}{\|\mathbf{w}\|_2^2},$$

$$\text{Softplus}(z) \stackrel{\text{def}}{=} \ln\big(1 + \exp(z)\big).$$

We also initialized d components of $\mathbf{u}_{reparametrized}$ as independent truncated normal continuous random variables to the interval $[-2, 2]$ with scale parameter 0.1 and fixed random seed = 12345.

2.1.6. Comparison of all methods

We compared all discussed methods for dimensions $d = 25, 50, 100, 200, 300, 400$. Authors trained their models using optimization process for the whole dataset, but we found that HVAE results are better and training process is faster when the dataset is divided on batches. HVAE and normalizing flows were trained for 2000 iterations across dataset divided on batches with 256 samples. For all experiments the dataset has $N = 10,000$ points and training was done using RMSProp (Tieleman and Hinton, 2017) with a learning rate of 10^{-3} and were conducted with fix random seed = 12345. We average the results for predicted θ for 3 different generated datasets according to Gaussian model and present the mean results on the table 1 and the figure 1. We trained Variational Bayes for big dimensions $d \geq 100$ more iterations (3000 or 4000) due to the fact that 2000 iterations were not enough for the convergence of ELBO. HVAE with tempering and IAF have the best learned $\hat{\theta} = \{\Delta, \mathbf{\Sigma}\}$ for the big dimensionality $d \geq 100$. Moreover, HVAE is good for $\mathbf{\Sigma}$ prediction for all dimensions as well Variational Bayes scheme. However, Variationl Bayes suffers most on prediction Δ as the dimension increases. Planar normalizing flows suffer on prediction $\mathbf{\Sigma}$ compared to IAF. Also we compare HVAE with tempering and without tempering, see table 2 and the figure 2. We can see that the tempered methods perform better than their non-tempered counterparts; this shows that time-inhomogeneous dynamics are a key ingredient in the effectiveness of the method.

Dimensionality, d	Methods, $\|\theta_{true} - \hat{\theta}\|^2$					
	HVAE, tempering		IAF	PNF		VB
	$K = 1$	$K = 10$	$K = 1$	$K = 1$	$K = 30$	
25	22.3	22.2	23.2	37.2	37.3	17.3
50	51.3	51.2	52.0	81.8	81.9	43.8
100	101.3	101.4	108.3	163.5	163.2	112.3
200	196.8	196.7	219.0	322.5	322.3	270.1
300	331.3	331.7	283.3	558.2	520.3	679.9
400	390.2	390.5	394.3	641.7	641.8	926.5

Table 1: Comparison of averages of $\|\theta_{true} - \hat{\theta}\|_2^2$ for several variational methods and choices of dimensionality d

	HVAE, $\|\theta_{true} - \hat{\theta}\|^2$			
Dimensionality,	with tempering		no tempering, $\beta_0 = 1$	
d	$K = 1$	$K = 10$	$K = 1$	$K = 10$
25	22.3	22.2	26.1	34.8
50	51.3	51.2	59.1	76.4
100	101.3	101.4	117.5	152.5
200	196.8	196.7	228.6	298.8
300	331.3	331.7	379.5	485.5

Table 2: Comparison of averages of $\|\theta_{true} - \hat{\theta}\|_2^2$ for HVAE with/without tempering and choices of dimensionality d

Dimensionality,	Methods, $\|\Delta_{true} - \hat{\Delta}\|^2$					
	HVAE, tempering		IAF	PNF		VB
d	$K = 1$	$K = 10$	$K = 1$	$K = 1$	$K = 30$	
25	22.2	22.2	22.0	22.3	22.3	17.3
50	51.3	51.1	49.9	51.2	51.2	43.8
100	101.2	101.3	104.1	101.1	101.2	112.3
200	196.5	196.4	210.8	196.6	196.9	270.1
300	330.8	331.3	271.1	335.4	331.4	679.9
400	389.6	389.9	378.0	389.6	389.3	926.5

Table 3: Comparison of averages of $\|\Delta_{true} - \hat{\Delta}\|_2^2$ for several variational methods and choices of dimensionality d

Dimensionality,	Methods, $\|\mathbf{\Sigma}_{true} - \hat{\mathbf{\Sigma}}\|_2^2$					
	HVAE, tempering		IAF	PNF		VB
d	$K = 1$	$K = 10$	$K = 1$	$K = 1$	$K = 30$	
25	0.04	0.04	1.14	14.97	14.97	$3 \cdot 10^{-4}$
50	0.08	0.08	2.16	30.64	30.74	$8 \cdot 10^{-4}$
100	0.17	0.15	4.18	62.38	62.07	$1.5 \cdot 10^{-3}$
200	0.33	0.31	8.18	125.9	125.42	$2.9 \cdot 10^{-3}$
300	0.50	0.45	12.21	222.84	188.87	$3.9 \cdot 10^{-3}$
400	0.61	0.61	16.33	252.11	252.44	$5.2 \cdot 10^{-3}$

Table 4: Comparison of averages of $\|\mathbf{\Sigma}_{true} - \hat{\mathbf{\Sigma}}\|_2^2$ for several variational methods and choices of dimensionality d

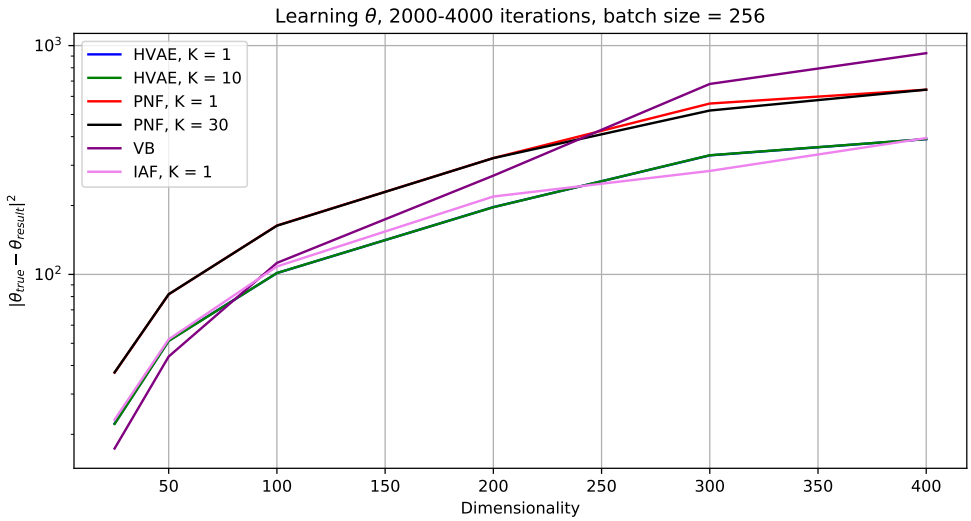


Figure 1: Comparison across all methods of learning θ, log-scale

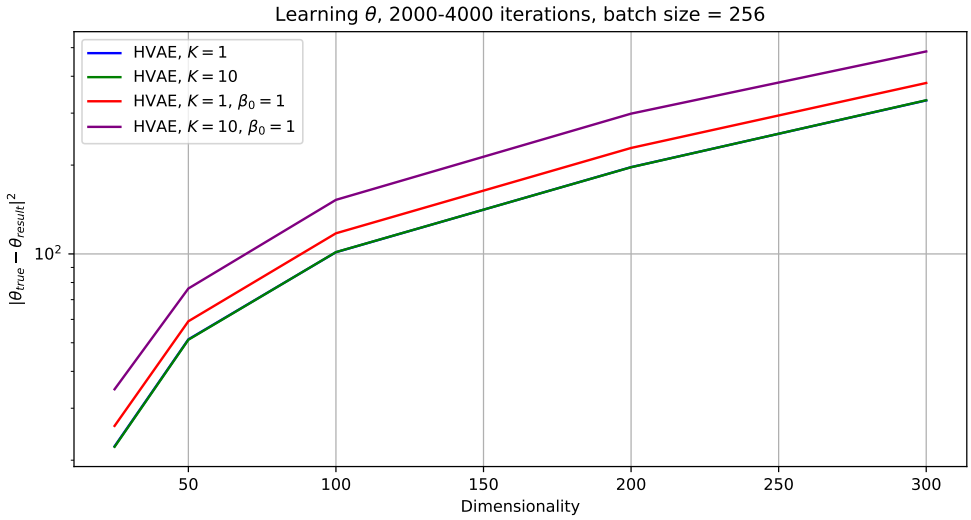


Figure 2: Comparison across HVAE with and without tempering of learning θ, log-scale

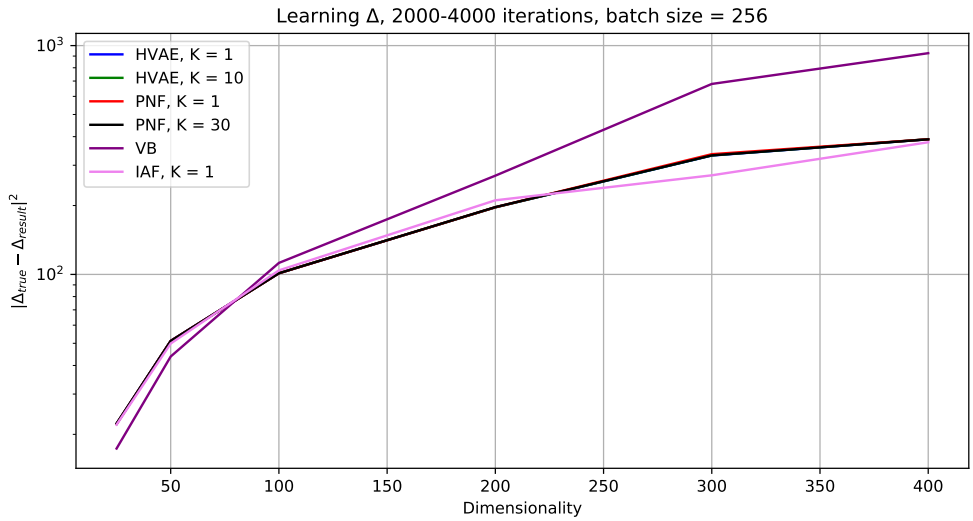


Figure 3: Comparison across all methods of learning Δ, log-scale

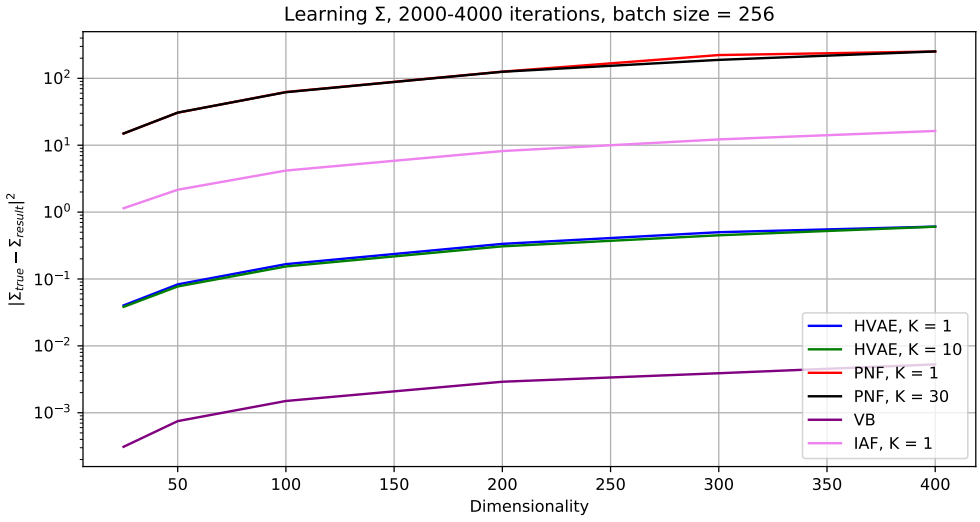


Figure 4: Comparison across all methods of learning $\mathbf{\Sigma}$, log-scale

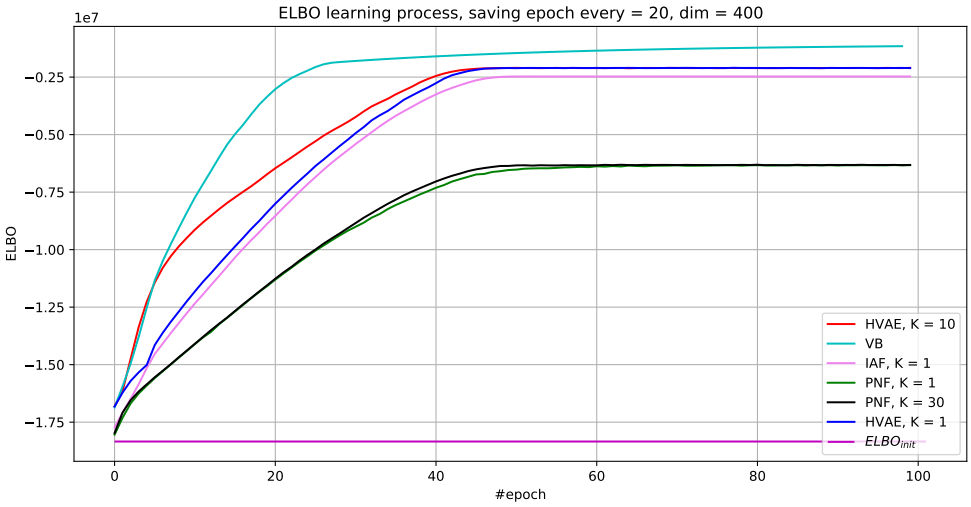


Figure 5: Comparison across all methods of learning ELBO process

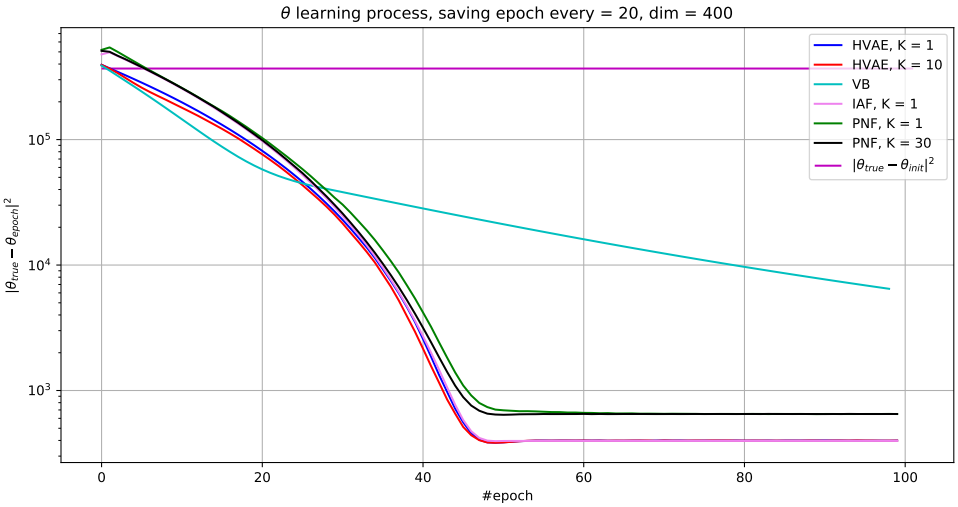


Figure 6: Comparison across all methods of learning θ processes, log-scale

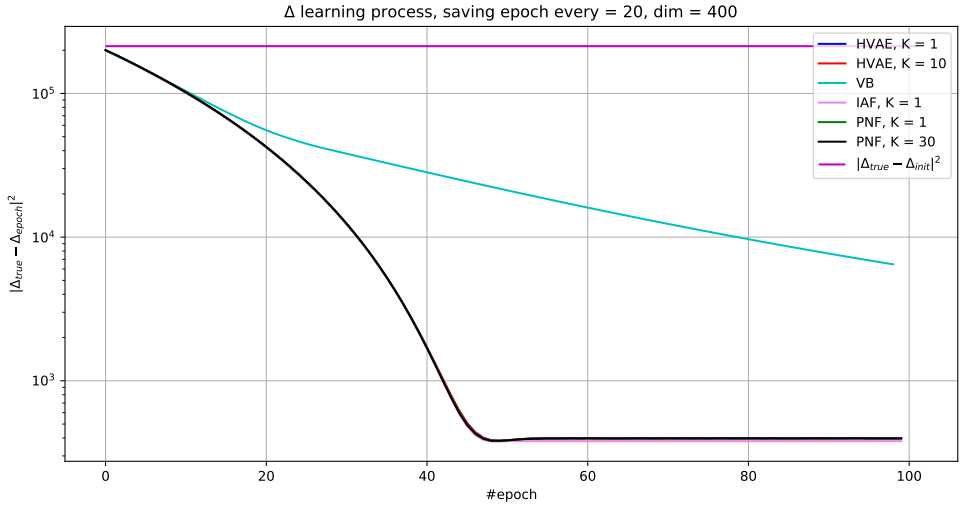


Figure 7: Comparison across all methods of learning Δ processes, log-scale

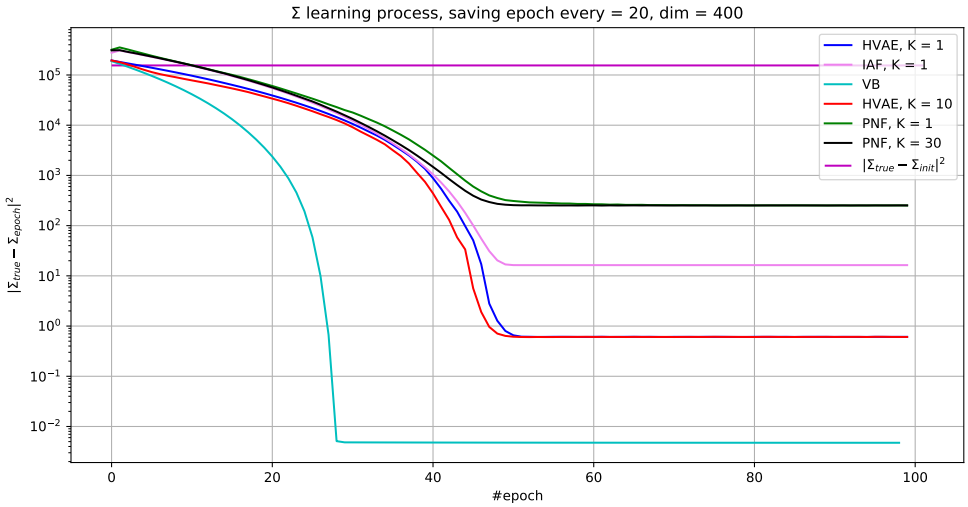


Figure 8: Comparison across all methods of learning Σ processes, log-scale

2.2. HVAE+IAF

We already know that in HVAE the ELBO is given by:

$$\mathcal{L}(\theta, x) = \mathbb{E}_{z_K, \rho_K}\left[\log \frac{p(x \mid z_K)\mathcal{N}(z_K \mid 0, I)\mathcal{N}(\rho_K \mid 0, I)}{q_K(z_K, \rho_K)}\right] = \mathbb{E}_{z_0, \gamma_0}\left[\frac{p(x, z_K)\mathcal{N}(\rho_K \mid 0, I)}{q_0(z_0)\mathcal{N}(\gamma_0 \mid 0, I)}\right].$$

In Auto-Encoder setting:

$$z_0 \sim \mathcal{N}(z_0 \mid \mu(x), \sigma(x)),$$

where μ, σ is the output of encoder network. Thus, we need to apply additional reparameterization:

$$z_0 = \mu(x) + \sigma(x)\varepsilon, \quad \varepsilon \sim \mathcal{N}(\varepsilon \mid 0, I),$$

with this:

$$\mathcal{L}(\theta, x) = \mathbb{E}_{\varepsilon, \gamma_0}\left[\log \frac{p(x \mid z_K)\mathcal{N}(z_K \mid 0, I)\mathcal{N}(\rho_K \mid 0, I)}{\left(\prod_{j=1}^{l} \sigma(x)_j\right)^{-1} \mathcal{N}(\varepsilon \mid 0, I)\mathcal{N}(\gamma_0 \mid 0, I)}\right] \propto^+$$

$$\mathbb{E}_{\varepsilon, \gamma_0}\Big[\log p(x \mid z_K) - 0.5 \cdot \|z_K\|_2^2 - 0.5 \cdot \|\rho_K\|_2^2\Big] + \sum_j \log\big(\sigma(x)_j\big),$$

and:

$$\nabla_\theta\left[\log p\Big(x \mid z_K\big(z_0(\varepsilon)\big)\Big) - 0.5 \cdot \|z_K(z_0(\varepsilon))\|_2^2 - 0.5 \cdot \|\rho_K(\rho_0(\gamma_0))\|_2^2 + \sum_j \log\big(\sigma(x)_j\big)\right],$$

is unbiased estimate of $\nabla_\theta \mathcal{L}(\theta, x)$.

Now assume, we additionally apply some flow to (z_K, ρ_K):

$$(\tilde{z}, \tilde{\rho}) = F(z_K, \rho_K),$$

$$q(\tilde{z}, \tilde{\rho}) = q_K(z_K, \rho_K) \cdot |\det \nabla F(z_K, \rho_K)|^{-1},$$

then:

$$\mathcal{L}(\theta, x) \propto^+ \mathbb{E}_{\varepsilon, \gamma_0}\Big[\log p(x \mid \tilde{z}) - 0.5 \cdot \|\tilde{z}\|_2^2 - 0.5 \cdot \|\tilde{\rho}\|_2^2 + \log|\det \nabla F(z_K, \rho_K)|\Big] + \sum_j \log(\sigma(x)_j),$$

and:

$$\nabla_\theta\Big[\log p(x \mid \tilde{z}) - 0.5 \cdot \|\tilde{z}\|_2^2 - 0.5 \cdot \|\tilde{\rho}\|_2^2 + \log|\det \nabla F(z_K, \rho_K)| + \sum_j \log\big(\sigma(x)_j\big)\Big]$$

is still an unbiased estimate of $\nabla_\theta \mathcal{L}(\theta, x)$.

In our case we applied IAF to (z_K, ρ_K) with easy-to-compute Jacobian. However, our tests on MNIST showed that such model performs much worse, compared to HVAE. You can see the results here: https://github.com/Daniil-Selikhanovych/h-vae/blob/master/demos/HVAE_IAF.ipynb. Since they are not noteworthy, we will not show any plots in this report.

2.3. MNIST

We appeal to the binarized MNIST handwritten digit dataset as an example of image generative task. The training data has the following form: $\mathcal{D} = \{x_1, \ldots, x_N\}$, where $x_i \in \mathcal{X} \subseteq \{0,1\}^d$ for $d = 28 \times 28 = 784$. We then formalize the generative model:

$$z_i \sim \mathcal{N}(0, I_\ell)$$
$$x_i \mid z_i \sim \prod_{j=1}^{d} \text{Bernoulli}\left((x_i)_j \mid \pi_\theta(z_i)_j\right)$$

,

for $i \in [N]$, where $(x_i)_j$ is the j^{th} component of $x_i, z_i \in \mathcal{Z} \equiv \mathbb{R}^\ell$ is the latent variable associated with x_i, and $\pi_\theta : \mathcal{Z} \to \mathcal{X}$ is an encoder (convolutional neural network). The VAE approximate posterior is given by $q_{\theta,\phi}(z_i \mid x_i) = \mathcal{N}(z_i \mid \mu_\phi(x_i), \mathbf{\Sigma}_\phi(x_i))$, where μ_ϕ and Σ_ϕ are separate outputs of the encoder parametrized by ϕ, and Σ_ϕ is constrained to be diagonal.

We set the dimensionalty of the latent space to $l = 64$. As we need both means and variances to parametrize the VAE posterior distribution, the output dimension of the linear layer is set to $l_{\mu,\sigma} = 128$. The encoder architecture is shown in Table 5 and the decoder architecture is shown in Table 6. We use Adam optimizer with standard parameters and learning rate set to 10^{-3}.

Layer (type)	Output Shape	Parameters Number
BatchNorm2d	[-1, 1, 28, 28]	2
Conv2d	[-1, 16, 14, 14]	416
Softplus	[-1, 16, 14, 14]	0
BatchNorm2d	[-1, 16, 14, 14]	32
Conv2d	[-1, 32, 7, 7]	12,832
Softplus	[-1, 32, 7, 7]	0
BatchNorm2d	[-1, 32, 7, 7]	64
Conv2d	[-1, 32, 4, 4]	25,632
BatchNorm2d	[-1, 32, 4, 4]	64
Reshape	[-1, 512]	0
Linear	[-1, 450]	230,850
Softplus	[-1, 450]	0
BatchNorm1d	[-1, 450]	900
Linear	[-1, 128]	57,728
	Total	328,520

Table 5: Encoder architecture.

We compare the performance of HVAE with the performance of IWAE (Burda et al. (2015)). We set the number of Monte-Carlo steps in HVAE and the number of importance samples in IWAE so that 1 training epoch requiers equal time to finish. In this setting

Layer (type)	Output Shape	Parameters Number
BatchNorm1d	[-1, 64]	128
Linear	[-1, 450]	29,250
Softplus	[-1, 450]	0
BatchNorm1d	[-1, 450]	900
Linear	[-1, 512]	230,912
Softplus	[-1, 512]	0
Reshape	[-1, 32, 4, 4]	0
BatchNorm2d	[-1, 32, 4, 4]	64
Conv2d	[-1, 32, 4, 4]	25,632
Softplus	[-1, 32, 7, 7]	0
BatchNorm2d	[-1, 32, 7, 7]	64
Conv2d	[-1, 16, 7, 7]	12,816
Softplus	[-1, 16, 14, 14]	0
BatchNorm2d	[-1, 16, 14, 14]	32
Conv2d	[-1, 16, 14, 14]	6,416
Softplus	[-1, 16, 28, 28]	0
Conv2d	[-1, 1, 28, 28]	401
Sigmoid	[-1, 1, 28, 28]	0
	Total	306,615

Table 6: Decoder architecture.

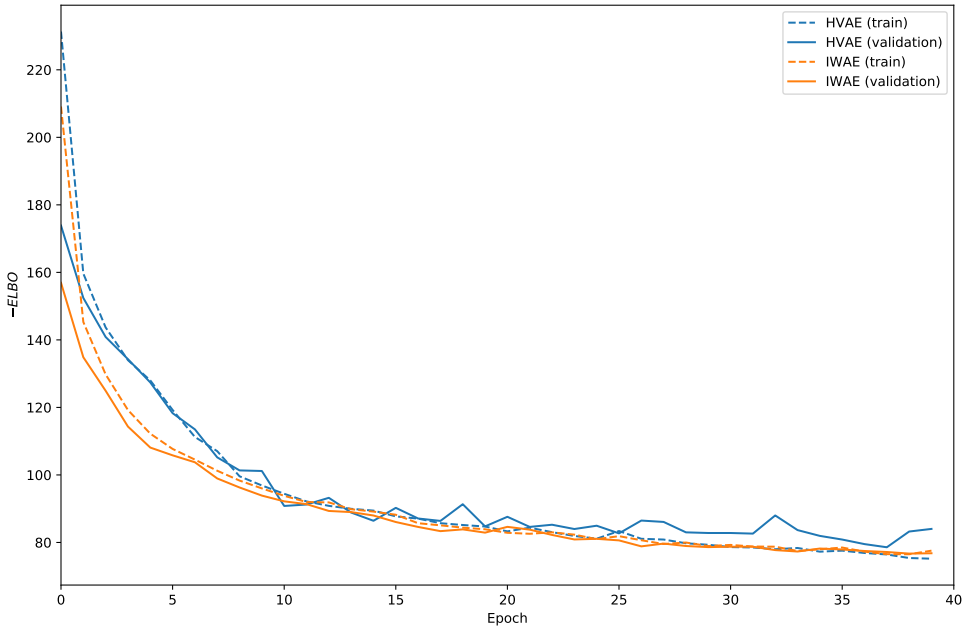
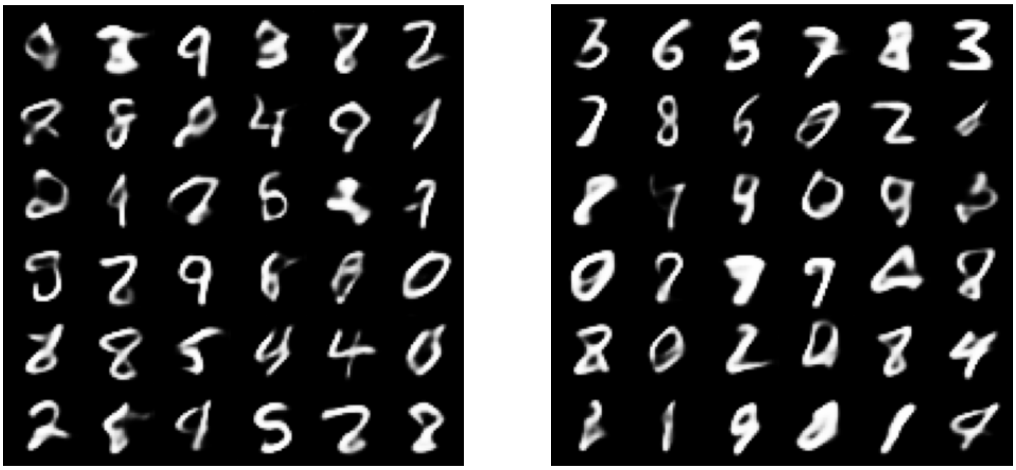


Figure 9: The training of HVAE and IWAE models.



(a) HVAE (b) IWAE

Figure 10: Images generated by both models trained on the binarized MNIST dataset.

we can compare them more fairly. We fix the number Monte-Carlo steps $K_{\text{MC}} = 15$ and the number of importance samples $K_{\text{IW}} = 40$. Both models are then optimized for 40 epochs. Corresponding plots are shown in Figure 9. It can be clearly seen that the training loss values are similar for both models, while the validation loss of HVAE is higher due to overfitting.

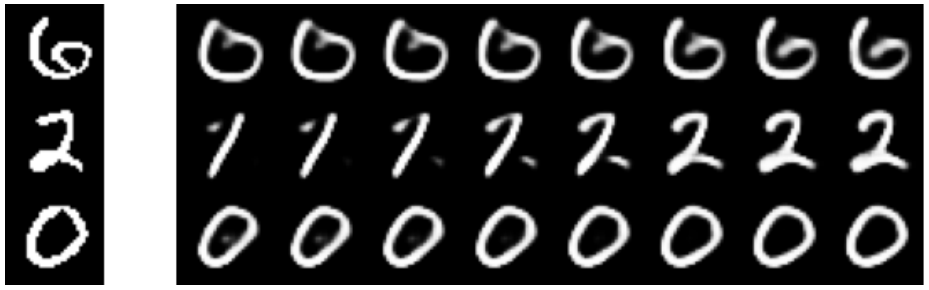
It is also important to compare the models outputs in terms of quality. The generated images are shown in Figure 10. Images generated via IWAE appear to be more blured. However, HVAE tends to generate sharper images while some of them can not be recognized as digits.

To better understand the behavior of both models, we study the decoded latent vectors of HMC chains. In Figure 11 one can clearly see that HVAE encoded vectors often correspond to the class that is different from the ground-truth, even though they are sharper. At the same time, IWAE produces reconstructions that are close to the true images.

3. Conclusion

In our work we have compared HVAE with other approaches and can summarize the results as follows:

- HVAE uses more information about the target distribution;
- tends to overfit on real datasets;
- performs comparable to other methods on artificial data;
- has some theoretical guarantees (optimal kernels, unbiased estimates of gradient);
- important alternative to normalization flows and IWAE.



(a) HVAE

(b) IWAE

Figure 11: Trajectories of HMC given a latent vector corresponding to the source image.

4. Contribution

1. Daniil Selikhanovych: experiments with Gaussian model, preparation of the presentation, preparation of the report, studying theoretical details. Github alias: Daniil-Selikhanovych.

2. Aleksandr Artemenkov: experiments on MNIST (HVAE vs IWAE), preparation of the presentation, preparation of the report, studying theoretical details. Github alias: alartum.

3. Igor Karpikov: experiments on MNIST (HVAE+IAF), preparation of the presentation, preparation of the report, studying theoretical details. Github alias: ivk22.

References

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. Hamiltonian variational auto-encoder. In *Advances in Neural Information Processing Systems*, pages 8167–8177, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

T Tieleman and G Hinton. Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *Technical Report.*, 2017.