Symmetric Equilibrium Learning of VAEs

Boris Flach
Czech Techn. University in Prague

Dmitrij Schlesinger
Dresden University of Technology

Alexander Shekhovtsov
Czech Techn. University in Prague

Abstract

We view variational autoencoders (VAE) as decoder–encoder pairs, which map distributions in the data space to distributions in the latent space and vice versa. The standard learning approach for VAEs is the maximisation of the evidence lower bound (ELBO). It is asymmetric in that it aims at learning a latent variable model while using the encoder as an auxiliary means only. Moreover, it requires a closed form a-priori latent distribution. This limits its applicability in more complex scenarios, such as general semi-supervised learning and employing complex generative models as priors. We propose a Nash equilibrium learning approach, which is symmetric with respect to the encoder and decoder and allows learning VAEs in situations where both the data and the latent distributions are accessible only by sampling. The flexibility and simplicity of this approach allows its application to a wide range of learning scenarios and downstream tasks.

1 INTRODUCTION

Variational autoencoders (Kingma and Welling, 2014; Rezende et al., 2014) are a well established and well analysed approach of learning latent variable models of the form $p(x) = \sum_z p(z)p(x|z)$. Given a distribution $\pi(x)$, $x \in \mathcal{X}$ in the data space and an assumed distribution $p(z)$, $z \in \mathcal{Z}$ in the latent space, a VAE combines a pair of parametrised distributions $p_\theta(x \mid z)$, $q_\varphi(z \mid x)$, which are usually modelled in terms of deep networks. The standard way to learn this encoder–decoder pair is to maximise the evidence lower bound of the data log-likelihood,

$$L_B(\theta, \varphi) = \mathbb{E}_{\pi(x)} \big[\mathbb{E}_{q_\varphi(z \mid x)} \log p_\theta(x \mid z) - D_{\mathrm{KL}}\big(q_\varphi(z \mid x) \,\big\|\, p(z)\big)\big]. \tag{1}$$

This learning formulation is particularly well suited to situations where only the generative model $p(x)$ is of interest. The research in this area in recent years has culminated in deep hierarchical VAEs (Vahdat and Kautz, 2020) and diffusion models (Ho et al., 2020; Rombach et al., 2022), which can be viewed also as hierarchical VAEs. The encoder's role is auxiliary in the ELBO, and it is even fixed to a simple noisy shrinkage in diffusion models. However, a learned encoder is often of interest in applications on its own — it can provide compact representations, useful for downstream tasks (*e.g.* for semantic hashing, Dadaneh et al. 2020). Furthermore, while only samples from $\pi(x)$ are needed in (1), an explicit model of $p(z)$ is required in order to compute (and differentiate) the KL-divergence term. Although solutions to the latter problem have been proposed, they come with some other limitations (discussed in detail in Section 5).

The asymmetries of the standard VAE learning approach pointed above make it difficult to use it in semi-supervised training scenarios and in situations where both spaces $\mathcal{X}$ and $\mathcal{Z}$ are complex and possibly structured, as for instance in semantic segmentation with images x and segmentations z. Learning an encoder–decoder pair in such a scenario would naturally allow solving inference problems in both directions between x and z as well as to build more complex models. The requirement to model $p(z)$ by a simple and tractable density becomes then a significant limitation.

In this work, we propose a symmetric learning approach inspired by game theory, which leads to a simple learning algorithm. The method can handle implicitly given marginal distributions $\pi(x)$ and $\pi(z)$. It does not require gradients of parametric discrete expectations like the gradient of ELBO w.r.t. the encoder parameters, and therefore no reparametrisation is needed. Consequently, handling discrete or continuous variables is simple. The method gives a novel view of the well-known wake-sleep algorithm (Hinton et al.,

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

1995), as discussed in Section 5. It can be applied to models with structured latent spaces, like hierarchical VAE, and extended to models consisting of 3 or more groups of variables. In the latter case, the model consists of several inference networks – one for each group of variables. They are learned jointly and can address an extended range of tasks at inference time, as we demonstrate experimentally.

The rest of the paper is organised as follows. In the next two sections we derive and analyse our novel learning approach. In the following section we exemplify its application to advanced models and learning setups. In the final experimental section we compare it with ELBO learning, show that it provides comparable model estimates, and demonstrate its applicability to more complex models not addressable by ELBO.

2 PROBLEM FORMULATION

We propose a generic learning approach, whose primary goal is to learn a decoder $p(x \,|\, z)$ and an encoder $q(z \,|\, x)$ in the following training scenarios:

Semi-supervised learning: We assume training samples $x \sim \pi(x)$ and $z \sim \pi(z)$ and possibly also joint samples $(x, z) \sim \pi(x, z)$, i.i.d. drawn from an unknown distribution $\pi(x, z)$ and its marginals.

Unsupervised learning: Only samples of $x \sim \pi(x)$ are observed. In this case the space $\mathcal{Z}$ is a free modelling choice.

Similar to VAE learning, the choice of the models for the decoder and encoder is dictated by the need to be able to evaluate (or at least differentiate) their respective log-densities and to sample from them. We will assume that the decoder and encoder belong to parametric exponential families of the form

$$p_\theta(x \,|\, z) \propto \exp\big[\langle \phi(x), f_\theta(z)\rangle\big], \tag{2a}$$
$$q_\varphi(z \,|\, x) \propto \exp\big[\langle \psi(z), g_\varphi(x)\rangle\big], \tag{2b}$$

where $\phi\colon \mathcal{X} \to \mathbb{R}^n$ and $\psi\colon \mathcal{Z} \to \mathbb{R}^m$ are fixed sufficient statistics. The mappings f and g are usually modelled by deep networks, parametrised by θ, φ. Notice that variables x, z can be either discrete or continuous depending on the chosen exponential family. Common choices are *e.g.* Bernoulli or Gaussian models.

3 SYMMETRIC EQUILIBRIUM LEARNING

We present our general approach and theoretical analysis for the semi-supervised learning task from the previous section, which naturally calls for a symmetric formulation.

For simplicity of exposition, let us assume that only marginal empirical distributions $\pi(x)$ and $\pi(z)$ are given, but no joint observations are available. The goal is to learn an encoder–decoder pair $q_\varphi(z \,|\, x)$ and $p_\theta(x \,|\, z)$ by (i) optimising the likelihood of the observed data and (ii) enforcing the encoder and decoder consistency at the same time. We formulate the learning task symmetrically as finding a *Nash equilibrium* for a two-player game. The strategy of the first player is represented by the decoder p_θ. Similarly, the strategy of the second player is represented by the encoder q_φ. The utility function of a player is the likelihood of the training data w.r.t. its strategy. Thereby, training examples are completed by the strategy of the other player. For example, the missing information in the examples $x \sim \pi(x)$ for the decoder likelihood is completed by the encoder strategy: $z \sim q_\varphi(z \,|\, x)$. Proceeding in the same way for the encoder, we obtain the utility functions

$$L_p(\theta, \varphi) = \mathbb{E}_{\pi(x)} \mathbb{E}_{q_\varphi(z \,|\, x)}[\log p_\theta(x \,|\, z)], \tag{3a}$$
$$L_q(\theta, \varphi) = \mathbb{E}_{\pi(z)} \mathbb{E}_{p_\theta(x \,|\, z)}[\log q_\varphi(z \,|\, x)]. \tag{3b}$$

As we will see later, the game aims at maximising the decoder likelihood and the encoder likelihood of the training data simultaneously, whereby the mutual completion reinforces decoder-encoder consistency.

A Nash equilibrium of the game is a pair (θ_*, φ_*) such that

$$\begin{aligned} L_p(\theta_*, \varphi_*) &\geqslant L_p(\theta, \varphi_*),\ \forall \theta, \\ L_q(\theta_*, \varphi_*) &\geqslant L_q(\theta_*, \varphi),\ \forall \varphi, \end{aligned} \tag{4}$$

i.e. a point at which neither player can improve its objective function. Towards finding an equilibrium we consider a simple gradient algorithm, in which each player tries to improve its utility w.r.t. to its strategy

$$\theta := \theta + \alpha \nabla_\theta L_p(\theta, \varphi); \quad \varphi := \varphi + \alpha \nabla_\varphi L_q(\theta, \varphi). \tag{5}$$

These updates may be executed in parallel or sequentially. Stochastic unbiased estimates of the required gradients are readily obtained by differentiating Monte-Carlo estimates of expectations (3) with as few as a single sample. Unlike in ELBO, the expectation $L_p(\theta, \varphi)$ does not need to be differentiated with respect to the encoder parameters and similarly for $L_q(\theta, \varphi)$. There is no need for the reparametrization trick in case of continuous variables or specialised gradient estimators through discrete samples in case of discrete variables.

Uniqueness It is well known that nonzero-sum games can have multiple and even infinitely many Nash equilibria. It is therefore crucial to analyse uniqueness of the solution as well as the convergence properties of the algorithm (5).

Extending the decoder and encoder to joint models via

$$p_\theta(x,z) = p_\theta(x \mid z)\pi(z); \quad q_\varphi(x,z) = q_\varphi(z \mid x)\pi(x) \quad (6)$$

the game utilities (3) can be compactly written as

$$\mathbb{E}_{q_\varphi(x,z)} \log p_\theta(x,z); \qquad \mathbb{E}_{p_\theta(x,z)} \log q_\varphi(x,z). \qquad (7)$$

This game is hard to analyse because of non-linear mappings involved.

To allow for theoretical analysis we will enlarge the spaces of feasible joint distributions by considering the following canonical exponential families

$$p_u(x,z) = \pi(z) \exp\bigl[\langle \phi(x,z), u\rangle - A(u)\bigr], \qquad (8a)$$
$$q_v(x,z) = \pi(x) \exp\bigl[\langle \psi(x,z), v\rangle - B(v)\bigr], \qquad (8b)$$

where $\phi(x,z), \psi(x,z)$ are sufficient statistics on (x,z), u and v are free parameter vectors and A and B are cumulant functions ensuring normalisation. The models (8) are log-linear in u and v by design. At the same time, with sufficiently complex $\phi(x,z)$ and $\psi(x,z)$ they can represent or approximate all models from the original families which were parametrised in terms of neural networks.

We explain this model relaxation for the case of binary valued vectors z and x. The components of the vector of natural parameters $f_\theta(z)$ in (2) and the corresponding cumulant function are then pseudo-Boolean functions and can be written as polynomials in the components of z. The same holds for the components of the sufficient statistic vector $\phi(x)$. This means that if we take the components of $\phi(x,z)$ in the relaxed class to contain all base monomials, then for any θ there would be a corresponding parameter vector u making the models equal. Notice that only under this correspondence the exponent part in (8a) matches the conditional distribution $p_\theta(x|z)$ while this is not true for a generic u.

Theorem 1. *The two-player game with utility functions*

$$L_p(u,v) = \mathbb{E}_{q_v(x,z)} \log p_u(x,z), \qquad (9a)$$
$$L_q(u,v) = \mathbb{E}_{p_u(x,z)} \log q_v(x,z) \qquad (9b)$$

and strategies given by exponential family distributions (8) *has a unique, asymptotically stable equilibrium.*

The proof is given in Appendix A. The idea is to construct a dual formulation of the game, which maximises the entropy under moment matching constraints. In this reformulation, it is then easy to prove the diagonal strict concavity condition (Rosen, 1965) – a sufficient condition for uniqueness. Following theorems 7-10 in (Rosen, 1965), the theorem implies that the simple gradient ascent algorithm (5) converges to the unique equilibrium point.

The theorem applies to log-linear models (8) with free natural parameters u and v and guarantees that the proposed algorithm converges to a unique equilibrium in this case. This has direct applicability to *e.g.* EF-Harmonium models, which are however outside of our scope. Its value for VAEs defined in terms of neural networks is rather indirect: if the algorithm works in the lifted space, it gives more confidence that it would also make sense in a subspace with a non-linear parametrisation.

Consistency Finally, we discuss the question of encoder–decoder consistency. We say that models $p(x \mid z)$ and $q(z \mid x)$ are *consistent* if there exists a joint distribution $m(x,z)$ of which they are conditional distributions (see also Liu et al. 2021). Since we model $p_\theta(x \mid z)$ and $q_\varphi(z \mid x)$ independently, they are in general inconsistent. Enforcing the consistency strictly, while keeping the models in exponential families (2), leads to a joint $m(x,z)$ necessarily collapsing to an EF-Harmonium (Arnold and Strauss 1991, Shekhovtsov et al. 2022), which is a severe limitation. However, encouraging consistency could serve as a useful regularisation and can improve learning efficiency.

We observe that our game formulation implicitly encourages consistency.

Proposition 1. *With the definition of joint distributions $p_\theta(x,z)$ and $q_\varphi(x,z)$ in* (6) *and their respective marginals, the game* (3) *is equivalent to the game with utilities:*

$$L_p' = \mathbb{E}_{\pi(x)}\bigl[\log p_\theta(x) - D_{\mathrm{KL}}(q_\varphi(z \mid x) \,\|\, p_\theta(z \mid x))\bigr],$$
$$L_q' = \mathbb{E}_{\pi(z)}\bigl[\log q_\varphi(z) - D_{\mathrm{KL}}(p_\theta(x \mid z) \,\|\, q_\varphi(x \mid z))\bigr].$$

See details in Appendix A. The utility L_p' is an alternative decomposition of ELBO into the data likelihood part and the encoder–posterior divergence, encouraging consistency. The utility L_q' is a symmetric counterpart. The difference to ELBO learning is that L_p' is optimised over θ only and not over φ and vice-versa for L_q'.

Similar to ELBO learning, there is no guarantee that the proposed learning approach will result in a consistent decoder–encoder pair defining a unique joint distribution. The necessity for such a joint distribution might be however dictated by the application for which the VAE is learned. Or it might arise if the learned VAE is only a part of a larger model, which requires such a joint distribution. In such cases we may consider the distribution (*e.g.* Liu et al. 2021)

$$m(x,z) = \tfrac{1}{2} m(z) p(x \mid z) + \tfrac{1}{2} m(x) q(z \mid x) \qquad (10)$$

with implicitly defined marginals $m(x)$ and $m(z)$. They must satisfy $m(x) = \sum_z m(x,z)$ and $m(z) = \sum_x m(x,z)$, which leads to the equations

$$m(x) = \textstyle\sum_z p(x \mid z) m(z), \tag{11a}$$
$$m(z) = \textstyle\sum_x q(z \mid x) m(x). \tag{11b}$$

While it is usually not possible to compute these marginals in closed form, it is nevertheless possible to sample from them and from the joint $m(x,z)$ as the limiting distributions of a Markov chain that alternates sampling of $x \sim p(x|z)$ and $z \sim q(z|x)$, as considered by Lamb et al. (2017).

4 ADVANCED MODELS AND LEARNING SETUPS

In this section we exemplify the application of the proposed learning approach to several practically relevant learning setups and more complex models.

Semi-Supervised Learning with Mixed Data We extend the model and learning setup from Section 3 in two respects. First, we assume that in addition to empirical distributions $\pi(x)$ and $\pi(z)$ we also have complete training examples, *i.e.*, matching pairs (x,z), forming an empirical distribution $\pi(x,z)$. Note that here π-s are empirical distributions, hence *e.g.* $\pi(x)$ need not be a marginal of $\pi(x,z)$. Second, we assume that the decoder's joint distribution is defined using its own parametrised prior for z, *i.e.* $p_\theta(x,z) = p_\theta(z) p_\theta(x \mid z)$.

The utility function of the decoder sums the p-likelihoods of the training set, of which the likelihoods of examples $(x,z) \sim \pi(x,z)$ and $z \sim \pi(z)$, are tractable. The missing information in examples $x \sim \pi(x)$ with intractable p-likelihood is completed by the encoder strategy $q_\varphi(z \mid x)$. Proceeding in the same way for the encoder, we get the utility functions

$$\begin{aligned} L_p(\theta,\varphi) &= \mathbb{E}_{\pi(x,z)}[\log p_\theta(x,z)] + \mathbb{E}_{\pi(z)}[\log p_\theta(z)] + \\ &\quad + \mathbb{E}_{\pi(x)} \mathbb{E}_{q_\varphi(z \mid x)}[\log p_\theta(x,z)], \qquad (12a) \\ L_q(\theta,\varphi) &= \mathbb{E}_{\pi(x,z)}[\log q_\varphi(z \mid x)] + \\ &\quad + \mathbb{E}_{\pi(z)} \mathbb{E}_{p_\theta(x \mid z)}[\log q_\varphi(z \mid x)]. \qquad (12b) \end{aligned}$$

Although we follow the symmetric approach as before, the utilities (12) are not entirely symmetric due to the model asymmetry: $p_\theta(x,z)$ has its own parametrised prior $p_\theta(z)$, whereas $q_\varphi(z \mid x)$ lacks a prior model for x.

Unsupervised Learning By unsupervised learning we will understand the case when only $x \sim \pi(x)$ is observed. The choice and interpretation of the $\mathcal{Z}$ space and the respective distribution is then completely free. We are interested in learning a decoder model $p_\theta(x,z) = p_\theta(x \mid z) p_\theta(z)$ and an encoder $q_\varphi(z \mid x)$ approximating $p_\theta(z \mid x)$.

The utility function for the decoder is given by its likelihood for the examples $x \sim \pi(x)$, completed by the encoder. To form a likelihood for the encoder, we consider examples generated by the decoder model. The resulting utility functions are

$$\begin{aligned} L_p(\theta,\varphi) &= \mathbb{E}_{\pi(x)} \mathbb{E}_{q_\varphi(z \mid x)}[\log p_\theta(x,z)], \\ L_q(\theta,\varphi) &= \mathbb{E}_{p_\theta(x,z)}[\log q_\varphi(z \mid x)]. \qquad (13) \end{aligned}$$

In comparison with ELBO approach, the required stochastic gradients of the log-likelihoods are easy to compute, as discussed in Section 3. Notice that the algorithm applies also in case when $p(z)$ is fixed and implicit, *i.e.* accessible by sampling only.

Hierarchical VAEs Finally, we show that our unsupervised learning approach generalises to hierarchical / autoregressive VAEs. We assume that the hidden state z consists of parts $z_0, z_1, \ldots, z_m$, and $x \sim \pi(x)$ can be observed. Such models come in two variants. In the first one the factorisation order of the encoder is reverse to the factorisation order of the decoder. Examples are *e.g.* Helmholtz machines (Hinton et al., 1995) and deep belief networks (Hinton et al., 2006). Here, we will consider the second variant, in which the encoder and decoder have the same order of factorisation:

$$p(x,z) = p(z_0) \prod_{i=1}^{m} p(z_i \mid z_{<i})\, p(x \mid z), \tag{14a}$$

$$q(z \mid x) = q(z_0 \mid x) \prod_{i=1}^{m} q(z_i \mid z_{<i}, x). \tag{14b}$$

The encoder of such models can share parameters with the decoder, in particular Sønderby et al. (2016) proposed to define the encoder by

$$q_{\theta,\varphi}(z_i \mid z_{<i}, x) \propto p_\theta(z_i \mid z_{<i}) f_i(z_i; d_i(x,\varphi)), \tag{15}$$

where f_i is a factorised function of z_i and $d_i(x,\varphi)$ are the hidden layer outputs of a deterministic encoder network $x \mapsto d_m \mapsto d_{m-1} \cdots \mapsto d_0$, parameterised by φ. The strategy of the first player is represented by the decoder parameters θ, while the strategy of the second player is represented by the encoder parameters φ. The utility functions for unsupervised learning are as in (13). Thanks to the factorisation of the decoder and encoder, they decompose into sums over the blocks $p(z_i \mid z_{<i})$ and $q(z_i \mid z_{<i}, x)$ and are tractable.

The model can be also learned "partially" semi-supervised by assuming that besides training examples $x \sim \pi(x)$ we have access to a (usually smaller) set of training examples $(x, z_0) \sim \pi(x, z_0)$. This is relevant, for example, when z_0 represents some hidden state(s) like classes or segmentations, on which we want to condition the decoder $p(x, z)$. The additional training examples will add

$$\mathbb{E}_{\pi(x,z_0)}\mathbb{E}_{q(z_{>0}\,|\,z_0,x)}[\log p(x, z)], \tag{16a}$$
$$\mathbb{E}_{\pi(x,z_0)}[\log q(z_0\,|\,x)] \tag{16b}$$

to the respective utility functions.

5 RELATED WORK

Wake-Sleep The learning algorithm (5) with utility functions (13) in the unsupervised case turns out to be equivalent to the wake-sleep (WS) algorithm first proposed by Hinton et al. (1995). However, we arrived at it from a conceptually new game-theoretic formulation, allowing for new analysis and generalisation to other settings (semi-supervised, partial observation scenarios). In Appendix B we give a brief overview of the original WS and follow-up works.

Implicit Prior An important advantage of the proposed method is allowing prior $\pi(z)$ to be implicit, *i.e.* accessible via samples only. Several works have extended VAEs to handle implicit encoders and priors. Mescheder et al. (2017) and Huszár (2017) estimate the log-density ratio $\log \frac{q_\varphi(z\,|\,x)}{\pi(z)}$ in ELBO by learning a logistic regression discriminator. Similar to GANs, this requires an inner loop with possibly complex discriminator. Molchanov et al. (2019) allow both the encoder and the prior to be an intractable mixture of tractable densities. At the training time, a finite sample from the mixture is used to form a density estimate of $\pi(z)$ and a lower bound on ELBO. These approaches are substantially more complex than ours and have further limitations. The prior can be made completely implicit, by assuming that the encoder-decoder model is consistent and hence defines a joint distribution and its marginals symmetrically. Towards this end Liu et al. (2021) explicitly optimise consistency and an expression that matches likelihood when assuming consistency.

Symmetric Learning Asymmetry of ELBO formulation has motivated several approaches, alternative to ours. Dumoulin et al. (2017) minimises Jensen-Shannon divergence between joint encoder $q(x, z) = \pi(x)q(x|z)$ and decoder $p(x, z)$. To estimate this divergence, a discriminator of joint samples is learned alongside, as in GANs. Pu et al. (2017) use a similar approach to minimise the symmetrised KL divergence. Lamb et al. (2017) learns the MCMC encoder–decoder sampler by using a discriminator between data-clamped and free-running chains. An important difference to our work is that the game in these approaches is between the discriminator and the model, not between decoder and encoder.

Unsupervised and Semi-Supervised VAEs Unsupervised equilibrium learning with utilities (13) can be reinterpreted to facilitate theoretical comparison with ELBO alongside Proposition 1. Furthermore, hierarchical model with observed z_0 (16) is closely related to semi-supervised learning with ELBO (Kingma et al., 2014). These connections are detailed in Appendix C.

6 EXPERIMENTS

Hierarchical VAE (MNIST) The goal of this experiment is to compare the symmetric equilibrium learning and ELBO learning on a simple dataset – MNIST images binarised by a suitably chosen threshold. We consider two hierarchical VAE model variants, each with two groups of binary valued latent variables $z_0 \in \mathcal{B}^{30}$ and $z_1 \in \mathcal{B}^{100}$. The decoder model is $p(x, z_0, z_1) = p(z_0)p(z_1\,|\,z_0)p(x\,|\,z_1)$, where we assume a uniform distribution $p(z_0)$. The encoder for the first model variant (similar to ladder VAEs) factorises in the same order as the decoder, i.e. $q(z_0, z_1\,|\,x) = q(z_0\,|\,x)q(z_1\,|\,z_0, x)$ and shares parameters with the decoder as described in Sec. 3. The encoder in the second model variant factorises in reverse order, i.e. $q(z_0, z_1\,|\,x) = q(z_1\,|\,x)q(z_0\,|\,z_1)$ and shares no parameters with the decoder. The networks used in the encoders and decoders are standard deep convolutional networks of decreasing and increasing spatial resolution respectively. More details are provided in Appendix E[1]. Training such models with ELBO requires a specialised gradient estimator for differentiating expectations in q w.r.t. its parameters. We use the estimator by Gregor et al. (2014), which is superior to straight-through and comparable to complex unbiased estimators for VAEs (Gu et al., 2016). Notice again, that no such approximation is required for the symmetric equilibrium learning.

Besides validating the generative capabilities of two resulting hierarchical VAEs, we want to analyse the consistency of their decoder–encoder pairs. We therefore generate images (i) from the decoder model p and (ii) from the limiting distribution $m(x)$ (see Sec. 3 for explanation). Fig. 1 and Table 1 indicate that the models obtained by symmetric learning achieves better consistency having at the same time slightly better FID scores. This is confirmed by tSNE embeddings of

[1]The code is available under
https://github.com/dschles70/symvae-aistats2024

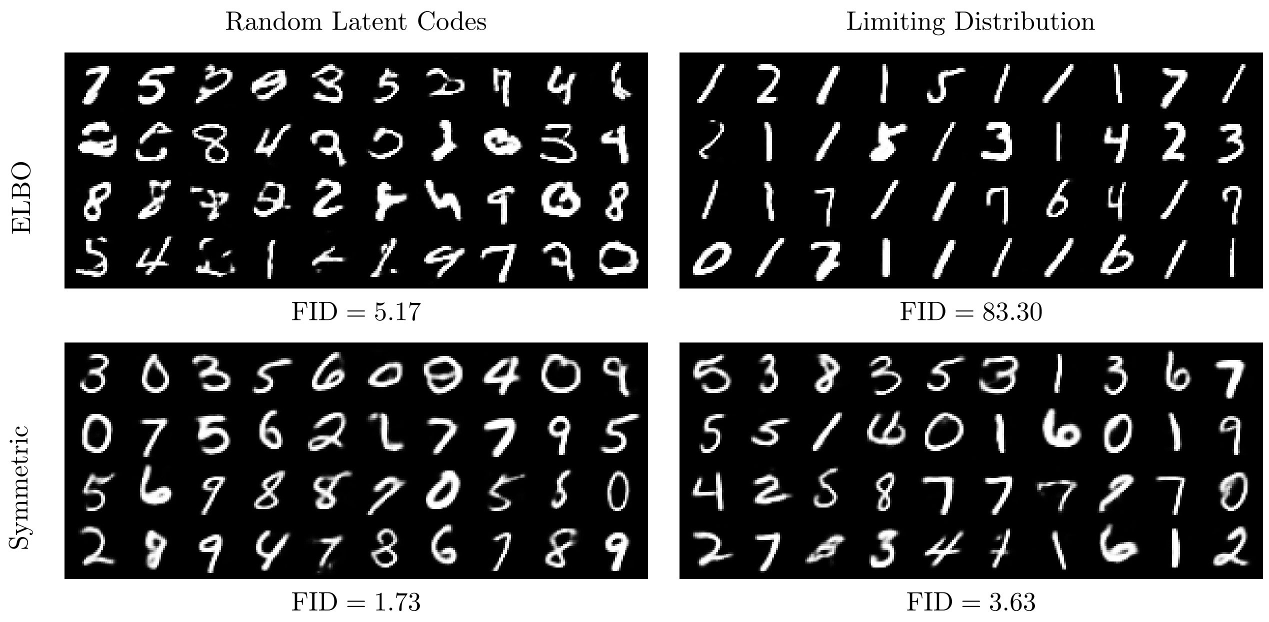

Figure 1: Ladder VAE (MNIST): FID scores and images generated from random latent codes and from limiting distributions of models learned by maximising ELBO and by symmetric equilibrium learning (images are shown by probabilities for better visibility).

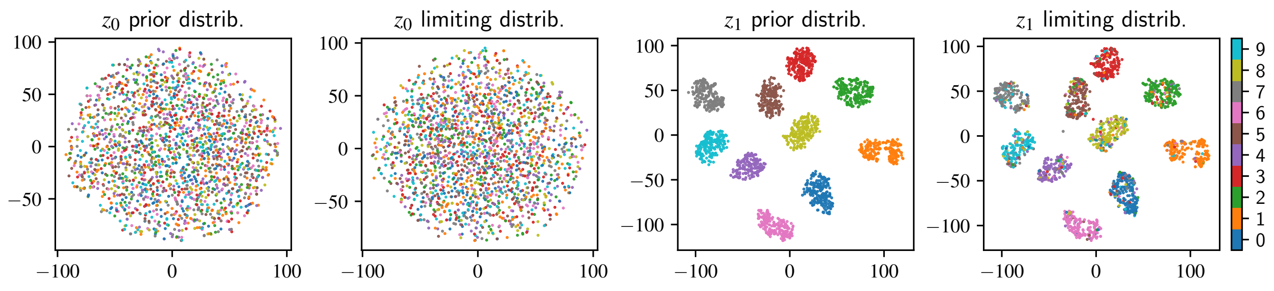

Figure 2: MNIST: tSNE embeddings for the VAE with class labels. Points are coloured by digit classes. See text for explanation.

z samples from the two models (see Appendix E).

Table 1: MNIST FID scores

model / alg.	rand. latent	limiting
LVAE, ELBO	5.17	83.30
LVAE, symmetric	1.73	3.63
RVAE, ELBO	5.83	29.59
RVAE, symmetric	0.81	5.40

To further strengthen this finding, we conducted similar experiments for the Fashion-MNIST dataset. Results and details are given in Appendix F.

The next experiment aims to show that the internal representations of a hierarchical VAE can be learned to have good generative and discriminative capabilities at the same time, even without "supervised" terms in the encoder objective as in (16). For this we extend z_0 by ten additional binary variables, which encode the class labels (one hot encoding). This means that $z_0 = (l, c)$ combines latent variables l with class labels c. We learn the model by symmetric learning from labelled examples (x, c), but use the following utility functions

$$L_p(\theta, \varphi) = \mathbb{E}_{\pi(x,c)} \mathbb{E}_{q_\varphi(l \,|\, x)} \mathbb{E}_{q_\varphi(z_{>0} \,|\, x, z_0)} [\log p_\theta(x, z)],$$
$$L_q(\theta, \varphi) = \mathbb{E}_{p_\theta(x,z)} [\log q_\varphi(z \,|\, x)]. \tag{17}$$

This means that the class information is used only when learning the decoder (notice that $q_\varphi(c, l \,|\, x)$ factorises w.r.t. to c and l). The encoder is learned solely on examples generated from the decoder, *i.e.* without any discriminative terms. The learned encoder achieves 99% classification accuracy on the MNIST validation set, with almost no decrease of the FID scores for the generated images (2.9 when sampled from the decoder and 4.0 when sampled from the limiting distribution). We also analyse tSNE embeddings of samples of the latent part l of $z_0 = (l, c)$ and samples of z_1, both from the prior distribution $p(z)$ and from the limiting distribution $m(z \mid c)$. Fig. 2 reveals that the latent part of z_0 is fully class agnostic, whereas z_1 is clearly clustered w.r.t. the digit classes. This can be interpreted as follows. The latent part l of $z_0 = (l, c)$ is "transversal" to the class labels c and presumably encodes image properties like stroke width, slant etc., whereas the internal representations in z_1 are clustered by digit classes and encode the appearance properties separately for each class.

Semantic Segmentation (CelebA) The following experiments illustrate the flexibility of the proposed approach on an application which is not accessible by ELBO learning. We consider the task of semantic segmentation with the goal to build a generative image segmentation model which can (i) generate image and segmentation pairs, (ii) segment given images, and (iii) generate images given a segmentation.

We use the CelebA-HQ dataset (Karras et al., 2018) and downscale its images and segmentations to 64×64 pixels for simplicity. Let $x \in \mathbb{R}^{3\times 64\times 64}$ be an image and $s \in \{1, \ldots, K\}^{64\times 64}$ be a segmentation (a categorical variable for each pixel). In order to model a distribution $p(x, s)$, we might try to learn a VAE with a decoder $p_\theta(x, s \mid z)$ and encoder $q_\varphi(z \mid x, s)$, assuming *e.g.* a uniform prior distribution for the vector of binary latent variables $z \in \mathcal{B}^m$. However, this alone will not meet our goals because we can not access the resulting distributions $p(s \mid x)$ and $p(x \mid s)$. We propose to model $p_\theta(x, s \mid z)$ as limiting distribution of a pair of parametrised conditional probability distributions $p_{\theta_1}(s \mid x, z)$ and $p_{\theta_2}(x \mid s, z)$ (see (10)). This means that the marginal probability distributions $p_\theta(x \mid z)$ and $p_\theta(s \mid z)$ are defined implicitly through the corresponding marginalisation constraints.

To summarise, the whole model consists of three learnable conditional probability distributions $p_{\theta_1}(s \mid x, z)$, $p_{\theta_2}(x \mid s, z)$ and $q_\varphi(z \mid x, s)$. This defines a nested game with three players. Their respective strategies are represented by θ_1, θ_2 and φ. Their utility functions are

$$L_{\theta_1}(\theta_1, \theta_2, \varphi) = \mathbb{E}_{\pi(x,s)}\mathbb{E}_{q_\varphi(z \mid x,s)}\Big[\log p_{\theta_1}(s \mid x, z) + \mathbb{E}_{p_{\theta_2}(x' \mid s,z)} \log p_{\theta_1}(s \mid x', z)\Big], \tag{18a}$$

$$L_{\theta_2}(\theta_1, \theta_2, \varphi) = \mathbb{E}_{\pi(x,s)}\mathbb{E}_{q_\varphi(z \mid x,s)}\Big[\log p_{\theta_2}(x \mid s, z) + \mathbb{E}_{p_{\theta_1}(s' \mid x,z)} \log p_{\theta_2}(x \mid s', z)\Big], \tag{18b}$$

$$L_\varphi(\theta_1, \theta_2, \varphi) = \mathbb{E}_{\pi(z)}\mathbb{E}_{p_\theta(x,s \mid z)}\Big[\log q_\varphi(z \mid x, s)\Big], \tag{18c}$$

where Gibbs sampling is applied for obtaining pairs $(x, s) \sim p_\theta(x, s \mid z)$ in the last utility function. (See Appendix D for detailed explanation).

To ease the training, we start by pre-training model parts for $p(s)$ and $p(x \mid s)$ separately. For the former we introduce latent variables $z_1 \in \mathcal{B}^{50}$, which should encode segmentation shapes, and define $p(s) = \sum_{z_1} p(z_1) \cdot p_{\theta_1}(s \mid z_1)$ with uniform prior $p(z_1)$. The model for $p(x \mid s)$ is a latent variable model $p(x \mid s) = \sum_{z_2} p(z_2) \cdot p_{\theta_2}(x \mid s, z_2)$ with latent variables $z_2 \in \mathcal{B}^{100}$, also uniformly distributed a-priori, which should encode appearance properties, like *e.g.* segment colours, textures, characteristic shadows etc. Both $p_{\theta_1}(s \mid z_1)$ and $p_{\theta_2}(x \mid s, z_2)$ are equipped with corresponding encoders, *i.e.* $q_{\varphi_1}(z_1 \mid s)$ and $q_{\varphi_2}(z_2 \mid x, s)$, and trained by symmetric learning, which is straightforward. All conditional probability distributions p and q are implemented as moderate complexity feed-forward networks, which output the parameters of the corresponding probability distribution. For example, $p_{\theta_2}(x \mid s, z_2) = \mathcal{N}(\mu_{\theta_2}(s, z_2), \sigma_{\theta_2}(s, z_2))$ is a diagonal normal distribution with means μ and standard deviations σ provided by the corresponding network.

Results for the learned $p_{\theta_2}(x \mid s)$ are illustrated in Fig. 3 in the following way. We consider pairs of training examples, each consisting of an image and its segmentation. The first example is encoded by $q_{\varphi_2}(z_2 \mid x, s)$ and the sampled latent code z_2 is used to decode the segmentation of the second example to an image by using $p_{\theta_2}(x \mid s, z_2)$.

After pre-training we extend the model part $p_{\theta_1}(s \mid z_1)$, learned in the previous step, to represent $p_{\theta_1}(s \mid x, z)$ by adding an "additional branch", *i.e.* we define

$$p(s \mid x, z) \propto \exp\big\langle f_1(z_1) + f_2(x, z_2), s_{oh}\big\rangle, \tag{19}$$

where f_1 is the pre-trained network, s_{oh} denotes the segmentation in one-hot encoding and f_2 is the additional network, which makes s dependent on x and z_2 as well. Its initial weights are chosen so that it outputs zeros at the beginning.

Finally, the model (18) is initialised by the pre-trained components and trained towards a Nash equilibrium for the three player game as explained above. Fig. 4 shows a few results. The model achieves 95.2% segmentation accuracy on the training set and 90.7% segmentation accuracy on the validation set.

An important property of the obtained model is its ability to complete missing information for any subset of its variables. Given a partial observation – *e.g.* an image part, or a segmentation part, or a combination of such parts – we can complete the missing data

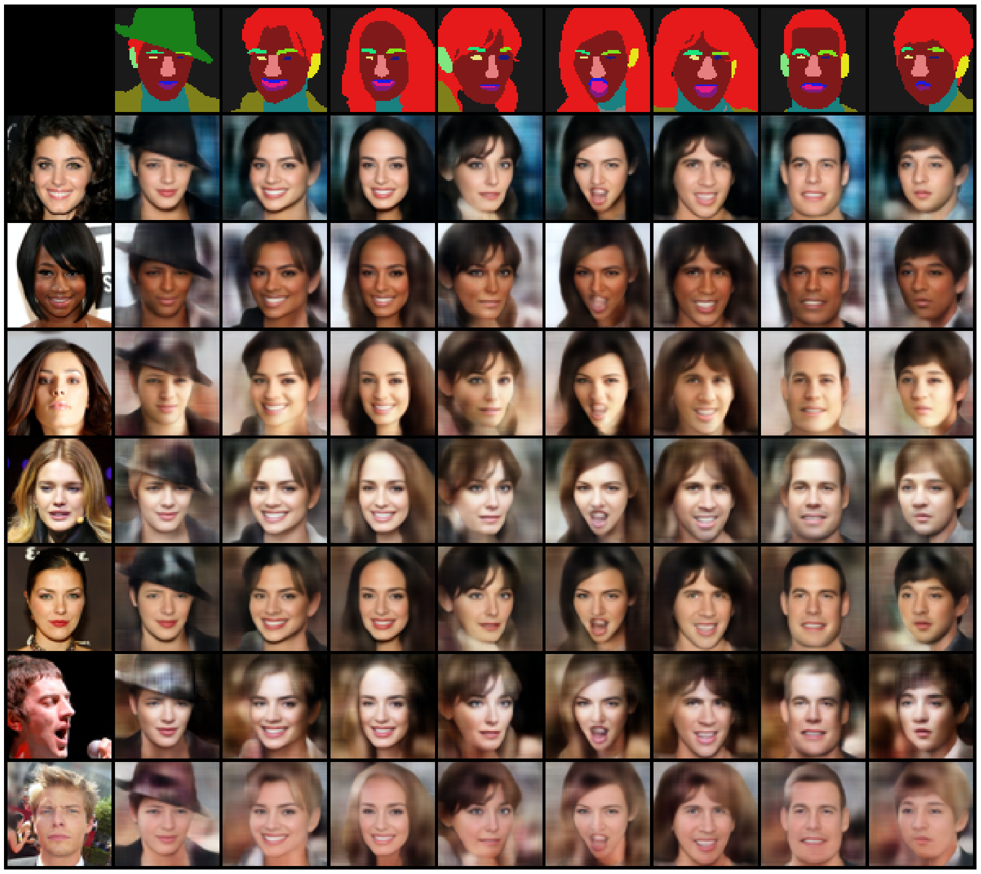

Figure 3: Given images and segmentations (x_i, s_i) from the training set (x_i are shown in the leftmost column), latent codes z_{2i} are sampled from $q_{\varphi_2}(z_2 \,|\, x_i, s_i)$. Given segmentations s_j shown in the top row, images $x_{i,j}$ are sampled from $p_{\theta_2}(x \,|\, s_j, z_{2i})$. Images are shown by mean values of the respective Gaussians for better visibility.

by sampling from the corresponding limiting distribution. We illustrate this property on the example of inference from incomplete images x. Let $x = (x_o, x_h)$ consist of two parts: an observed part x_o and a hidden part x_h. In order to segment such an image by the maximum marginal decision strategy, we need to compute the marginal probabilities $p(s_i \,|\, x_o)$ for each pixel i. They can be estimated by Gibbs sampling, which alternately draws all unobserved random variables, including x_h. We accumulate segmentation label frequencies for each pixel during the sampling and finally decide for the label with the highest occurrence. A few results are presented in Fig. 5. As compared to the segmentation from complete images, the segmentation accuracy drops from 95.2% to 92.8% for the training set and from 90.7% to 88.8% for the validation set. We consider this accuracy drop as minor, because the segmentations inferred for the hidden image parts need not necessarily coincide with the ground truth – they should only be “plausible”, which is seen in the figure. Although not the primary goal of this experiment, Gibbs sampling allows at the same time to reconstruct the image content in the hidden parts (*i.e.* in-painting). For this we employ a mean-marginal decision, *i.e.* we average all sampled image values observed during Gibbs sampling. Although the results are sometimes not perfect (see the last row in Fig. 5), it is however enough to infer reasonable segmentations.

7 CONCLUSION

We propose an alternative learning approach for variational autoencoders. For this we view VAEs as decoder–encoder pairs and derive a symmetric learning formulation inspired by game theory, which leads to a simple learning algorithm for finding a Nash equilibrium. We prove its uniqueness under fairly general assumptions. The proposed method can be applied for various learning scenarios and for models with complex, possibly structured latent spaces. This includes implicit distributions in the latent space as well as discrete latent variables. We show experimentally that the models learned by this method are comparable to those obtained by ELBO learning and demonstrate its applicability for tasks that are not accessible by stan-

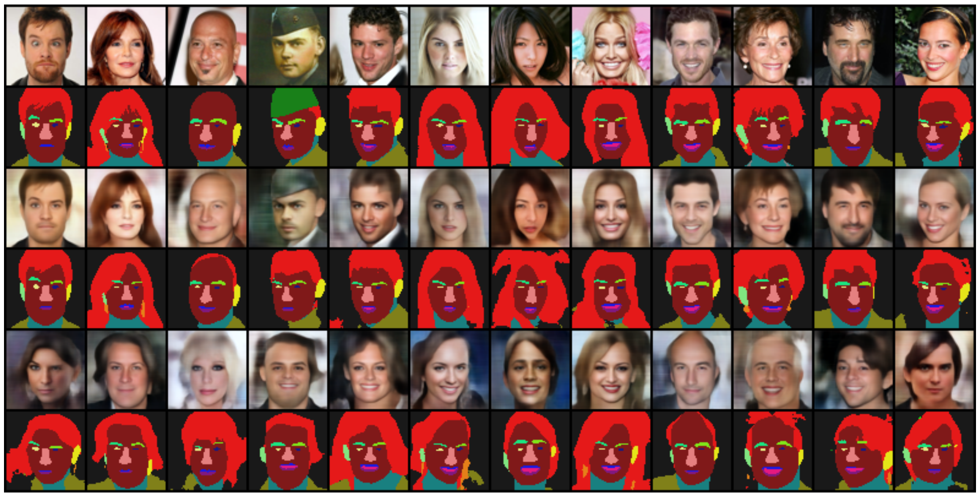

Figure 4: First two rows: training data (x, s). Third and fourth rows: reconstructed images, and segmentations sampled from $p_\theta(x \mid s, z_2)$ and from $p_\theta(s \mid x, z)$ with $z \sim q_\varphi(z \mid x, s)$. Last two rows: sampling image–segmentation pairs from the full limiting distribution.

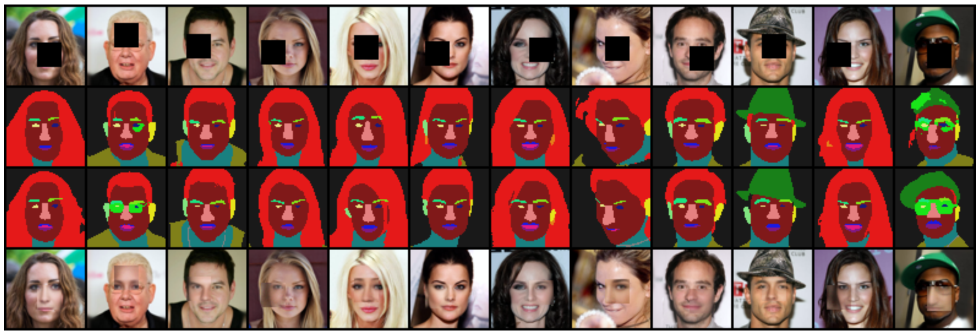

Figure 5: Segmentation from incomplete data. First row: original images from the validation set with hidden parts depicted as black squares. Second row: predicted segmentations, Third row: ground truth segmentations. Fourth row: "in-painting" – average over all images obtained during generation (with clamped visible part).

dard VAE learning.

Acknowledgements

We would like to thank our colleagues Tomas Werner and Denis Barucic for their continued interest in this work and their valuable comments and discussions which helped to improve the manuscript. We also thank the reviewers for their critical remarks, which encouraged us to present more experiments and to resolve remaining unclarities. B.F. gratefully acknowledges support by the Czech OP VVV project "Research Center for Informatics" (CZ.02.1.01/0.0/0.0/16019/0000765). D.S. was supported by the German Federal Ministry of Education and Research (BMBF) project 01/S18026A-F and by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) project 01MN23021A. A.S. was supported by the Czech Science Foundation grant GA24-12697S. The authors would like to thank the Center for Information Services and HPC (ZIH) at TU Dresden for providing computing resources.

References

B. C. Arnold and D. J. Strauss. Bivariate distributions with conditionals in prescribed exponential families. *Journal of the Royal Statistical Society Series B (Methodological)*, 53(2), 1991.

Jorg Bornschein and Yoshua Bengio. Reweighted wake-sleep. *ArXiv, 1406.2751*, 2015.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *ICLR*, 2016.

Siamak Zamani Dadaneh, Shahin Boluki, Mingzhang Yin, Mingyuan Zhou, and Xiaoning Qian. Pairwise supervised hashing with Bernoulli variational autoencoder and self-control gradient estimator. In *UAI*, volume 124, 2020.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2017.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *ICML*, 2014.

Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. Muprop: Unbiased backpropagation for stochastic neural networks. In *ICLR*, May 2016.

Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214), May 1995.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7), jul 2006.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, volume 33, 2020.

Ferenc Huszár. Variational inference using implicit distributions. *ArXiv*, abs/1702.08235, 2017.

Shiro Ikeda, Shun-ichi Amari, and Hiroyuki Nakahara. Convergence of the wake-sleep algorithm. In *NeurIPS*, volume 11, 1998.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. In *NeurIPS*, NIPS'14, 2014.

Alex M Lamb, Devon Hjelm, Yaroslav Ganin, Joseph Paul Cohen, Aaron C Courville, and Yoshua Bengio. Gibbsnet: Iterative adversarial inference for deep graphical models. In *NeurIPS*, volume 30, 2017.

Tuan Anh Le, Adam R. Kosiorek, N. Siddharth, Yee Whye Teh, and Frank Wood. Revisiting reweighted wake-sleep for models with stochastic control flow. In *UAI*, volume 115, 2020.

Chang Liu, Haoyue Tang, Tao Qin, Jintao Wang, and Tie-Yan Liu. On the generative utility of cyclic conditionals. In *NeurIPS*, 2021.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In *ICML*, 2017.

Dmitry Molchanov, Valery Kharitonov, Artem Sobolev, and Dmitry P. Vetrov. Doubly semi-implicit variational inference. In *AISTATS*, volume 89, 2019.

Yuchen Pu, Weiyao Wang, Ricardo Henao, Liqun Chen, Zhe Gan, Chunyuan Li, and Lawrence Carin. Adversarial symmetric variational autoencoder. In *NeurIPS*, volume 30, 2017.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

J. B. Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, 33(3), 1965. doi: 10.2307/1911749.

Alexander Shekhovtsov, Dmitrij Schlesinger, and Boris Flach. VAE approximation error: ELBO and exponential families. In *ICLR*, 2022.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *NeurIPS*, volume 29, 2016.

Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *NeurIPS*, 2020.

Eszter Vértes and Maneesh Sahani. Flexible and accurate inference and learning for deep generative models. In *NeurIPS*, volume 31, 2018.

Li Wenliang, Theodore Moskovitz, Heishiro Kanagawa, and Maneesh Sahani. Amortised learning by wake-sleep. In *ICML*, volume 119, 13–18 Jul 2020.

A PROOFS

In this section we provide proofs of formal claims regarding uniqueness and consistency-enforcement.

Theorem 1. *The two-player game with utility functions*

$$L_p(u,v) = \mathbb{E}_{q_v(x,z)} \log p_u(x,z), \tag{9a}$$
$$L_q(u,v) = \mathbb{E}_{p_u(x,z)} \log q_v(x,z) \tag{9b}$$

and strategies given by exponential family distributions (8) *has a unique, asymptotically stable equilibrium.*

Proof. We repeat here the model assumptions (8) for convenience

$$p_u(x,z) = \pi(z) \exp\big[\langle \phi(x,z), u\rangle - A(u)\big] \tag{20a}$$
$$q_v(x,z) = \pi(x) \exp\big[\langle \psi(x,z), v\rangle - B(v)\big]. \tag{20b}$$

Our proof relies on the classic result of (Rosen, 1965), who shows that games satisfying *diagonal strict concavity* (DSC), a condition stronger than concavity, have unique Nash equilibria.

Since the log-partition function of an exponential family is convex in its natural parameters, it follows that the game utilities are concave in their own strategies. A sufficient condition for the stronger DSC criterion is that the symmetrised Jacobian of the mapping

$$\begin{bmatrix} u \\ v \end{bmatrix} \mapsto \begin{bmatrix} \nabla_u L_p(u,v) \\ \nabla_v L_q(u,v) \end{bmatrix} \tag{21}$$

is negative definite. The most convenient way to prove this condition is to "dualise" the game. Maximising $L_p(u,v)$ w.r.t. u is equivalent to finding the exponential family model, whose expected sufficient statistic $\mathbb{E}_{p_u}[\phi(x,z)]$ coincides with $\mathbb{E}_{q_v}[\phi(x,z)]$. This follows from

$$\nabla_u \mathbb{E}_{q_v(x,z)} \log p_u(x,z) = \mathbb{E}_{q_v(x,z)}[\phi(x,z)] - \mathbb{E}_{p_u(x,z)}[\phi(x,z)]. \tag{22}$$

The corresponding dual task reads

$$F_p(p) = \sum_{x,z} p(x,z)\big[\log p(x,z) - \log \pi(z)\big] \to \min_p \tag{23a}$$

$$\text{s.t.} \begin{cases} \mathbb{E}_p[\phi(x,z)] = \mathbb{E}_q[\phi(x,z)] \\ \sum_{x,z} p(x,z) = 1. \end{cases} \tag{23b}$$

This can be seen by noticing that (23) is a convex optimisation task with linear constraints. Hence, we can apply Fenchel duality

$$\inf_p \big\{ F_p(p) \mid Ap = b \big\} = \sup_\gamma \big\{ \langle b, \gamma\rangle - F_p^*(A^T\gamma) \big\}, \tag{24}$$

where F_p^* denotes the Fenchel conjugate function of F_p. For our case, we have $b = (\mathbb{E}_q[\phi(x,z)], 1)$ and the corresponding dual variables $\gamma = (u, \lambda)$. The Fenchel conjugate of the function $f(p) = p\log p - p \log \pi$ is $f^*(w) = \pi e^{w-1}$. Substituting all terms in the rhs of (24) and solving for λ, we get the task $\mathbb{E}_{q_v(x,z)} \log p_u(x,z) \to \max_u$.

Applying the same dualisation for $L_q(u,v)$, we obtain the following "dual" game. The strategy of the first player is represented by $p(x,z)$ and the strategy of the second player is represented by $q(x,z)$. The utility functions $-F_p(p)$ and $-F_q(q)$ of the players depend on their respective strategy only. The game has additional linear constraints, where we assume existence of an interior feasible point (p,q). The assertion of the theorem follows from Theorems 3,4,9 in (Rosen, 1965), if we prove that the symmetrised Jacobian of the mapping

$$\begin{bmatrix} p \\ q \end{bmatrix} \mapsto \begin{bmatrix} \nabla_p F_p(p) \\ \nabla_q F_q(q) \end{bmatrix} \tag{25}$$

is positive definite. This is trivial since the Jacobian is diagonal with elements $1/p(x,z)$ in the first half of the diagonal and elements $1/q(x,z)$ in its second half. □

Proposition 1. *With the definition of joint distributions $p_\theta(x,z)$ and $q_\varphi(x,z)$ in* (6) *and their respective marginals, the game* (3) *is equivalent to the game with utilities:*

$$L'_p = \mathbb{E}_{\pi(x)}\big[\log p_\theta(x) - D_{\mathrm{KL}}(q_\varphi(z\,|\,x) \,\|\, p_\theta(z\,|\,x))\big],$$
$$L'_q = \mathbb{E}_{\pi(z)}\big[\log q_\varphi(z) - D_{\mathrm{KL}}(p_\theta(x\,|\,z) \,\|\, q_\varphi(x\,|\,z))\big].$$

Proof. For completeness, we include the fact that L'_p is an alternative form of the ELBO. It is verified as follows:

$$\log p_\theta(x) - D_{\mathrm{KL}}(q_\varphi(z\,|\,x) \,\|\, p_\theta(z\,|\,x)) \tag{26a}$$
$$= \log p_\theta(x) - \mathbb{E}_{q_\varphi(z\,|\,x)}\Big[\log \frac{q_\varphi(z\,|\,x)}{p_\theta(z\,|\,x)}\Big] \tag{26b}$$
$$= \mathbb{E}_{q_\varphi(z\,|\,x)}\Big[\log p_\theta(x) - \log \frac{q_\varphi(z\,|\,x)}{p_\theta(z\,|\,x)}\Big] \tag{26c}$$
$$= \mathbb{E}_{q_\varphi(z\,|\,x)}\Big[\log \frac{p_\theta(x) p_\theta(z\,|\,x)}{q_\varphi(z\,|\,x)}\Big] \tag{26d}$$
$$= \mathbb{E}_{q_\varphi(z\,|\,x)}\Big[\log \frac{p_\theta(x|z)\pi(z)}{q_\varphi(z\,|\,x)}\Big] \tag{26e}$$
$$= \mathbb{E}_{q_\varphi(z\,|\,x)}\Big[\log p_\theta(x|z)\Big] - D_{\mathrm{KL}}(q_\varphi(z\,|\,x) \,\|\, \pi(z)).$$

Therefore,

$$L'_p = L_p - \mathbb{E}_{\pi(x)}[D_{\mathrm{KL}}(q_\varphi(z\,|\,x) \,\|\, \pi(z))]. \tag{27}$$

Therefore, for a fixed φ, utilities L'_p and L_p share all local and global minima in θ. It is straightforward to see that (θ_*, ϕ_*) is an equilibrium of the game with utilities L'_p and L'_q iff it is an equilibrium of the game with utilities (3). □

B WAKE SLEEP

In this section we give a brief overview of the original wake-sleep (WS) algorithm and follow-up works.

Hinton et al. (1995) considered a multilayer network of stochastic neurons. The "recognition" (encoder) connections are used to convert the input vector into a representation in one or more layers of hidden units. The "generative" (decoder) connections are then used to reconstruct an approximation to the input vector from its underlying representation. In the wake phase of WS, given an observed sample x from the training dataset, a sample of hidden states z is obtained from the encoder network and the decoder is learned on the joint sample (x, z). In the sleep phase a joint sample is drawn from the decoder model and the encoder is learned.

The model was initially assuming binary units and factorised encoder and decoder. In case of a hierarchical encoder–decoder model, the learning decouples over layers and no back-propagation is needed. Extended to a deep exponential family model (Vértes and Sahani, 2018), it is equivalent to a hierarchical VAE with the reverse encoder structure.

Bornschein and Bengio (2015) et al. uses importance sampling, similar to IWAE (Burda et al., 2016), to tighten the bounds for the decoder and introduces a wake-phase (importance weighted) update of the encoder, tightening the ELBO (as in VAE) as well.

Vértes and Sahani (2018) and Wenliang et al. (2020) showed that the encoder in WS can be specified implicitly by its mean parameters, which allows for non-conditionally independent encoders. This makes encoders more flexible so that higher quality decoder can be trained but impairs inference.

The advantage of not requiring differentiation through discrete sampling has been explored by Le et al. (2020) for models with stochastic control flow.

To our knowledge, prior work has neither extended WS to semi-supervised setting nor discussed the question of *why* it is a reasonable algorithm. The only analysis attempt by Ikeda et al. (1998) is limited to a strictly consistent encoder-decoder in a simple special case.

C (DIS-)SIMILARITIES TO ELBO

In this section we elaborate on similarities and difference between symmetric learning and ELBO learning in unsupervised as well as semi-supervised case (Kingma et al., 2014).

Unsupervised Recall, that in the unsupervised case we consider utility functions

$$\begin{aligned} L_p(\theta, \varphi) &= \mathbb{E}_{\pi(x)} \mathbb{E}_{q_\varphi(z \mid x)} [\log p_\theta(x, z)], \\ L_q(\theta, \varphi) &= \mathbb{E}_{p_\theta(x,z)} [\log q_\varphi(z \mid x)]. \end{aligned} \tag{28}$$

As discussed in Proposition 1, the decoder utility can be equivalently replaced with the common ELBO $L_B(\theta, \phi)$ (both have the same dependence on θ). The difference to VAE of Kingma and Welling (2014) is therefore only in the encoder learning. In VAE the encoder is learned to tighten ELBO, *i.e.* to minimise the so-called *reverse* KL divergence in the expectation over the data distribution:

$$\mathbb{E}_{\pi(x)} \big[D_{\mathrm{KL}}(q_\varphi(z|x) \,\|\, p_\theta(z|x)) \big]. \tag{29}$$

In the equilibrium learning, minimising L_q in (28) w.r.t. encoder is equivalent to minimising

$$\mathbb{E}_{p_\theta(x)} \big[D_{\mathrm{KL}}(p_\theta(z|x) \,\|\, q_\varphi(z|x)) \big], \tag{30}$$

which is a *forward* KL divergence between the same conditional distributions, and the expectation is over the generative model $p_\theta(x) = \sum_z p_\theta(z) p_\theta(x|z)$. The choice of the encoder as the true posterior, $q(z|x) = p(z|x)$, when possible (i.e. for consistent models), is optimal to both ELBO and symmetric learning. But in general, L_q leads to different preferred solutions.

Semi-Supervised Semi-supervised learning of VAE was previously considered by Kingma et al. (2014). It can be seen that the hierarchical model (14a) is a generalisation of the generative model of Kingma et al. (2014): the state z consists of two parts (z_0, z_1), where z_0 is the image label, available only for a part of images. Similar to unsupervised case, when learning the decoder for a fixed encoder, the learning objective (Kingma et al., 2014, Eq. 8) is equivalent to our L_p.

Only the learning of encoder differs. In their formulation the encoder minimises

$$\begin{aligned} &\mathbb{E}_{\pi(x)} D_{\mathrm{KL}}(q(z|x) \,\|\, p(z|x)) \\ &+ \mathbb{E}_{\pi(x,z_0)} D_{\mathrm{KL}}(q(z_1|x, z_0) \,\|\, p(x, z)) \\ &- \alpha \mathbb{E}_{\pi(x,z_0)} \log q(z_0|x), \end{aligned} \tag{31}$$

where α is an empirical coefficient. In case when there are no unlabelled pairs, the first term disappears and the ELBO learning approach (Kingma et al., 2014) decouples into learning of a conditional VAE (decoder and encoder conditioned on z_0: $p(x|z_1, z_0)$, $q(z_1|x, z_0)$) and *an independent* discriminative learning of the encoder part $q(z_0|x)$ from the labelled data only. Thus, the generative counterpart of the model has no effect on learning of the recognition part (unless there is a parameter sharing).

In our formulation the encoder maximises

$$\mathbb{E}_{p(x,z)} \log q(z|x) + \mathbb{E}_{\pi(x,z_0)} \log q(z_0|x). \tag{32}$$

This objective is more homogeneous because both terms correspond to forward KL divergences. When there are no unlabelled training pairs, the objective stays the same and the encoder part $q(z_0|x)$ still needs to fulfil two goals: to approximate the posterior of the decoder $p(z_0|x)$ (in the expectation over the generated distribution $p(x)$, like in the unsupervised case) and to approximate the empirical distribution $\pi(z_0|x)$ (in the expectation over $\pi(x)$). A weighting coefficient might be appropriate here as well to balance the two objectives. Our semi-supervised MNIST experiment in Section 6 with utilities (17) shows that even when switching off the discriminative counterpart, the encoder still efficiently learns to classify.

D LEARNING MODELS WITH IMPLICIT MARGINALS

Here we give a more detailed derivation of the learning in situations, where a joint model is given by means of its conditional distributions only, *i.e.* marginal distributions are given implicitly. In particular, we used it in our experiments with CelebA to define and learn $p(x, s \mid z)$, where x are images, s are segmentations, and z are latent variables. Since everything is conditioned on z we will omit it for clarity and use x and s as variables of interest to be inline with our experiments.

With the above agreement, we want to learn two conditional probability distributions $p_\theta(x \mid s)$ and $q_\varphi(s \mid x)$. As both images and segmentations are rather complex, it is desirable to avoid making any assumptions about the prior (marginal) distributions $p(s)$ and $q(x)$. Towards this end, we consider the MCMC process starting from a random state and alternately sampling using $p_\theta(x \mid s)$ and $q_\varphi(s \mid x)$. This process defines two limiting joint distributions, depending on which variable was sampled last:

$$m(s) p_\theta(x \mid s) \quad \text{and} \quad m(x) q_\varphi(s \mid x), \tag{33}$$

where $m(x)$ and $m(s)$ are solutions to the stationary equations

$$m(x) = \sum_s p_\theta(x \mid s) m(s) \tag{34a}$$

$$m(s) = \sum_x q_\varphi(s \mid x) m(x). \tag{34b}$$

It is natural to consider the mixture of these two limiting distributions

$$m(x, s) = \frac{1}{2}\Big[m(s) p_\theta(x \mid s) + m(x) q_\varphi(s \mid x)\Big], \tag{35}$$

as we suggest in (10). Our goal therefore will be to maximise the likelihood of the data $\pi(x, s)$ under this mixture joint model. The likelihood can be lower-bounded w.r.t. mixture components as

$$\begin{aligned}&\mathbb{E}_{\pi(x,s)} \log m(x, s)\\ &\quad \geq \frac{1}{2}\Big[\mathbb{E}_{\pi(s)} \log m(s) + \mathbb{E}_{\pi(x,s)} \log p_\theta(x \mid s) + \\ &\qquad \mathbb{E}_{\pi(x)} \log m(x) + \mathbb{E}_{\pi(x,s)} \log q_\varphi(s \mid x)\Big].\end{aligned} \tag{36}$$

Note that this lower bound is tight if the mixture components coincide, *i.e.* $p_\theta(x \mid s)$ and $q_\varphi(s \mid x)$ are consistent. The terms in (36) corresponding to p_θ and q_φ are tractable under assumption (2). However, $m(x)$ and $m(s)$ are not given in closed form and depend on both θ and φ. We approximate their defining equations (34) as

$$m_\theta(x) = \sum_s p_\theta(x \mid s) \pi(s) \tag{37a}$$

$$m_\varphi(s) = \sum_x q_\varphi(s \mid x) \pi(x) \tag{37b}$$

and use these expressions in the mixture model (35). With this approximation, (36) sums the data likelihood terms with respect to separate model components $p_\theta(x \mid s)$, $m_\theta(x)$, $q_\varphi(s \mid x)$ and $m_\varphi(s)$. Hence, optimising this sum decouples into optimising the two objectives

$$\begin{aligned}L_p &= \mathbb{E}_{\pi(x,s)} \log p_\theta(x \mid s) + \mathbb{E}_{\pi(x)} \log m_\theta(x),\\ L_q &= \mathbb{E}_{\pi(x,s)} \log q_\varphi(s \mid x) + \mathbb{E}_{\pi(s)} \log m_\varphi(s)\end{aligned} \tag{38}$$

independently in θ and φ, respectively. It remains only to explain how to handle $\log m_\theta(x)$ and $\log m_\varphi(s)$, which are still intractable. Substituting (37) and introducing a lower bound for $\log m_\theta(x)$ w.r.t. summation over s gives

$$\begin{aligned}\mathbb{E}_{\pi(x)} \log m_\theta(x) \geq \mathbb{E}_{\pi(x)} \mathbb{E}_{q_\varphi(s \mid x)} \Big[&\log p_\theta(x \mid s) + \\ &\log \pi(s) - \log q_\varphi(s \mid x)\Big].\end{aligned} \tag{39}$$

If we consider the equilibrium learning approach, the objective L_p is to be optimised only w.r.t. its own parameters θ, and therefore we can drop $\log \pi(s)$ and $\log q_\varphi(s \mid x)$ terms. Applying similar steps to $\mathbb{E}_{\pi(s)} \log m_\varphi(s)$ leads to the following effective equilibrium learning objectives:

$$\begin{aligned}\tilde{L}_p(\theta, \varphi) = \mathbb{E}_{\pi(x,s)} &\log p_\theta(x \mid s) + \\ &+ \mathbb{E}_{\pi(x)} \mathbb{E}_{q_\varphi(s \mid x)} \log p_\theta(x \mid s),\end{aligned} \tag{40}$$

$$\begin{aligned}\tilde{L}_q(\theta, \varphi) = \mathbb{E}_{\pi(x,s)} &\log q_\varphi(s \mid x) + \\ &+ \mathbb{E}_{\pi(s)} \mathbb{E}_{p_\theta(x \mid s)} \log q_\varphi(s \mid x).\end{aligned} \tag{41}$$

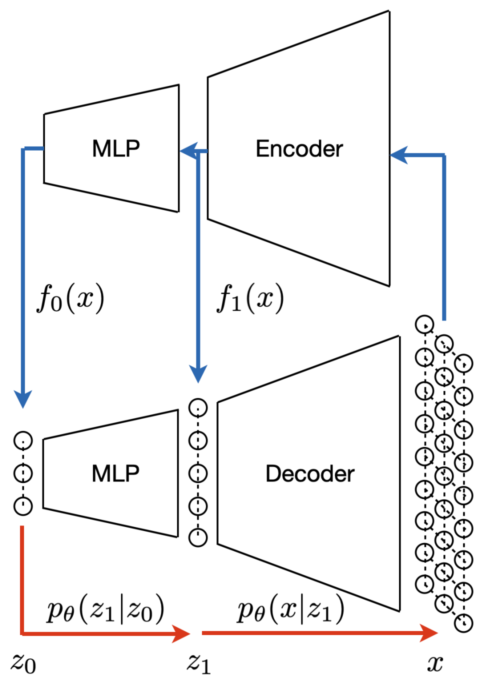

Figure 6: MNIST network architecture.

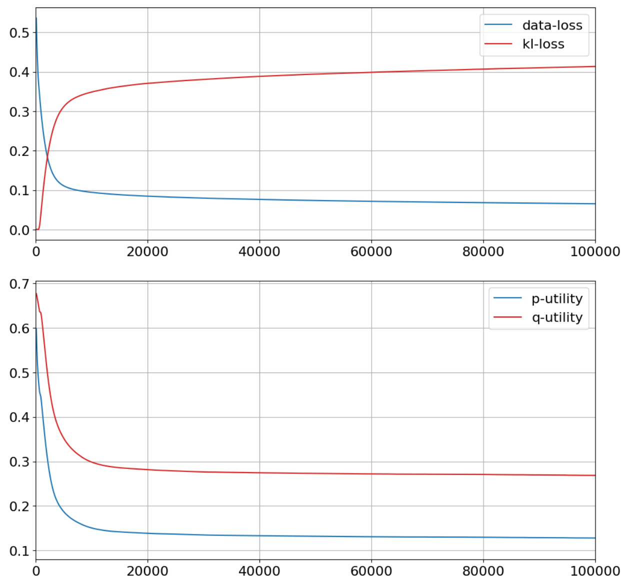

Figure 7: MNIST training with ELBO and symmetric learning. *Top*: data-term and KL-term for ELBO learning, *bottom*: negative utilities for symmetric learning.

Note that the first terms in these utilities correspond to the pseudo-likelihood objective, whereas the mutual completion in the second terms additionally enforces consistency.

E ADDITIONAL DETAILS FOR MNIST EXPERIMENTS

Here, we provide additional implementation details for the HVAE models used by symmetric learning and by ELBO optimisation in the first MNIST experiment. The first model variant is defined by the decoder $p_\theta(z_0, z_1, x) = p(z_0)p_\theta(z_1 \mid z_0)p_\theta(x \mid z_1)$ and the encoder $q_\varphi(z_0, z_1, x) = \pi(x)q_\varphi(z_0 \mid x)q_\varphi(z_1 \mid z_0, x)$, where $p(z_0)$ is uniform and $\pi(x)$ is the data distribution. The network architecture is shown in Fig. 6. The one-dimensional components are connected by a Multi-layer Perceptron (MLP) architecture. We used two hidden layers, 600 hidden units each in our MLPs. Connections between z_1 and x are implemented by standard convolutional encoder/decoder architectures with decreasing and increasing spatial resolutions respectively. Both encoder and decoder have 6 hidden layers, connected by 2D-convolution operations. In order to effectively reduce the spatial dimension some convolutions are performed with strides. We used the tanh activation function everywhere. The network weights are learned using the Adam-optimiser.

The hierarchical decoder consists of two "separate" networks, an MLP and a decoder, representing $p_\theta(z_1 \mid z_0)$ and $p_\theta(x \mid z_1)$ respectively. The encoder corresponding to the direct factorisation order (shown in the figure) is a multi-head network. The common part is an encoder, which produces intermediate features, whereas the heads are an MLP for $f_0(x)$ and a single fully connected layer for $f_1(x)$. Two network outputs $f_0(x)$ and $f_1(x)$ serve as multiplier to the hierarchical decoder model, so $q_\varphi(z_0) = f_0(x)$ and $q_\varphi(z_1 \mid z_0, x) \propto p_\theta(z_1 \mid z_0) \cdot f_1(x)$. For the reverse factorisation order we keep the hierarchical encoder architecture basically the same but split it into two separate networks: the encoder for $q_\varphi(z_1 \mid x)$ and the MLP for $q_\varphi(z_0 \mid z_1)$.

The learning curves for losses/utilities are shown in Fig. 7 for ELBO learning and symmetric learning respectively as a function of gradient update steps. For better clarity all values are normalised by the number of corresponding elements, e.g. we show the per-pixel data-loss in ELBO. It is clearly seen that the convergence behaviours are pretty similar in both cases: all values converge very quickly to almost their final values, followed by a long period in which they change much more slowly. However, we observed that the quality of generated images keeps improving, even after the losses/utilities have almost reached saturation. Hence, we run all our experiments with a small learning rate of 10^{-4} for 1M gradient update steps (note: only first 100k steps are shown in Fig. 7 for better visibility).

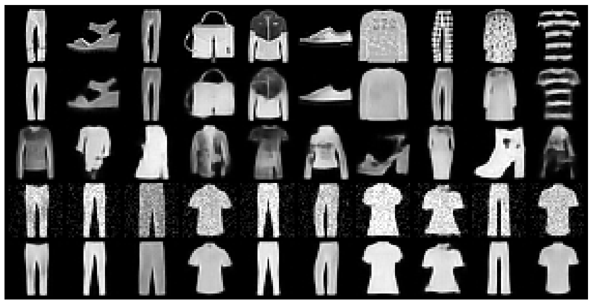
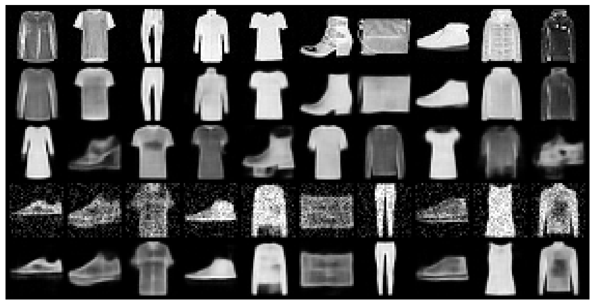

Figure 8: Fashion MNIST. Left: ELBO learning, right: symmetric learning. In each image: top row: original images, second row: means of the reconstructed images, i.e. $\mu(z_1)$ with $z_1 \sim q(z_1 \mid x)$, third row: images generated from random codes (means visualised), fourth row: sampling from limiting distribution including image noise, last row: sampling from limiting distribution, means visualised.

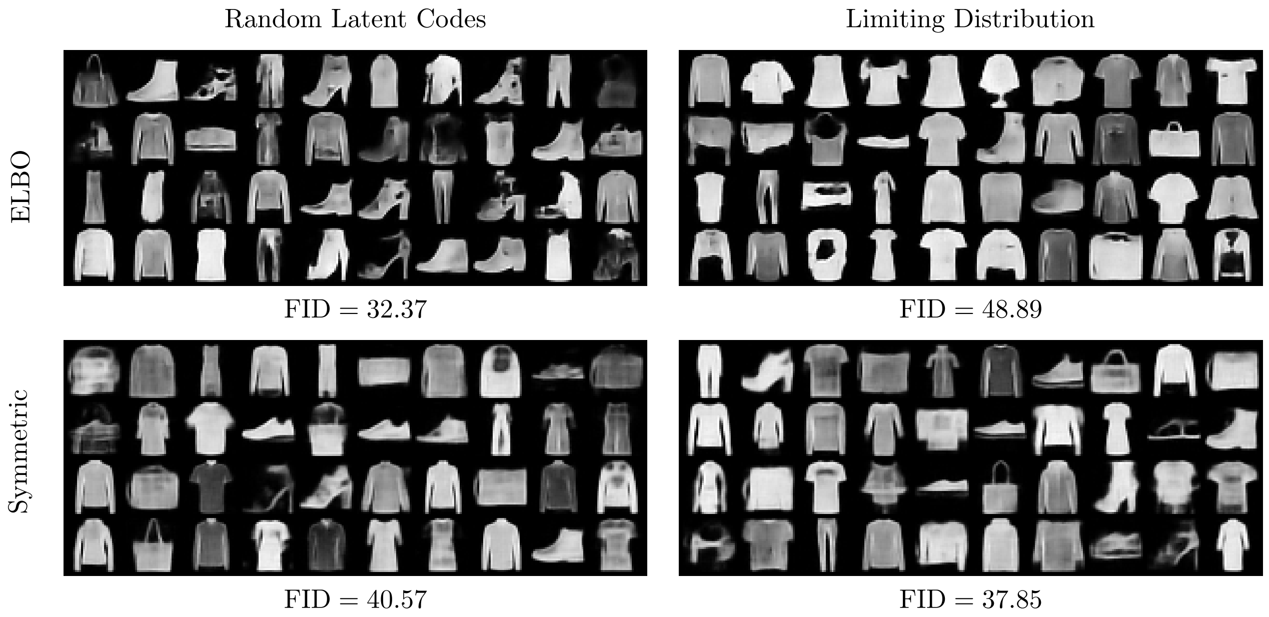

Figure 9: Fashion MNIST. FID scores and images generated from random latent codes and from limiting distributions of models learned by maximising ELBO and by symmetric equilibrium learning (images are shown by means for better visibility).

We further compare the HVAE models obtained by symmetric learning and by ELBO optimisation by embedding samples for z_0 and z_1 from (i) the *prior* distributions $p(z_0)$, $p_\theta(z_1)$, (ii) the *posterior* distributions $q_\varphi(z_0)$, $q_\varphi(z_1)$, and (iii) the *limiting* distributions $m_{\theta,\varphi}(z_0)$ and $m_{\theta,\varphi}(z_1)$ for each of the two models by tSNE. Fig. 10 shows that all three samples match well for the model learned by symmetric learning. This is however not the case for the model learned by ELBO.

F FASHION MNIST

We also tested our approach for HVAE with the direct encoder factorisation order on the Fashion MNIST dataset. The model is exactly the same as the one used in our first MNIST experiment, except:

- Images are grey-valued now. We model them by a Gaussian, where the means for all pixels are computed by a network, and the standard deviation is common for all pixels and does not depend on z, i.e. $p_{\theta,\sigma}(x \mid z_1) = \mathcal{N}(x; \mu_\theta(z_1), \sigma)$. The network architecture for $\mu_\theta(z_1)$ is the same as the decoder in the MNIST experiment, σ is learned alongside with the network weights.
- We observed that the overall results are slightly better (especially for ELBO), when using ReLU activations in $p(x \mid z_1)$ instead of tanh used for MNIST.

The results are shown in Figs. 8 and 9. They confirm

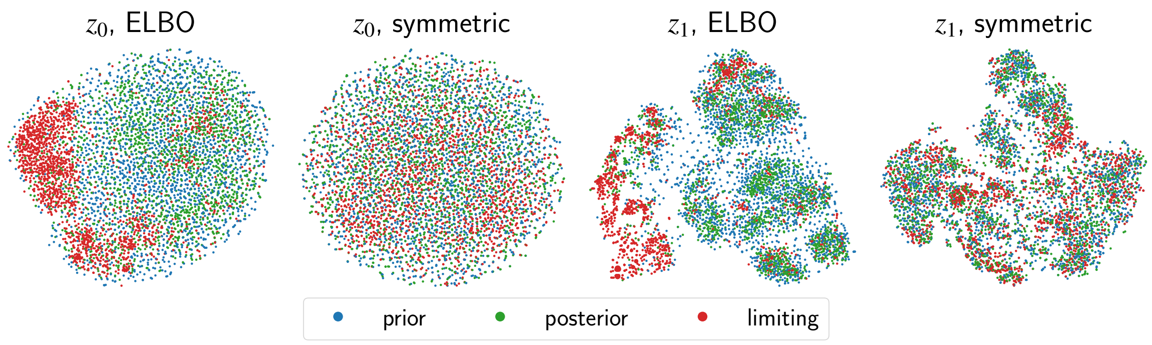

Figure 10: MNIST: tSNE embeddings of latent variables z_0 and z_1 for ELBO maximisation and symmetric learning.

our finding, that ELBO and symmetric learning are on par, whereby the latter produces more consistent encoder/decoder pairs.

CHECKLIST

Our experiments aim to illustrate theoretical findings. Hence, the description given in the paper should be enough to reproduce all presented results. Source code for selected experiments is available under https://github.com/dschles70/symvae-aistats2024. We use only standard open-source libraries as well as publicly available data in all our experiments.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not applicable]
   We analyse the algorithm convergence, which is the most crucial issue. Other aspects like *e.g.* time and space complexity are standard.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (*e.g.*, data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (*e.g.*, with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (*e.g.*, type of GPUs, internal cluster, or cloud provider). [No]

4. If you are using existing assets (*e.g.*, code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [No]
   (c) New assets either in the supplemental material or as a URL, if applicable. [No]
   (d) Information about consent from data providers/curators. [No]
   (e) Discussion of sensible content if applicable, *e.g.*, personally identifiable information or offensive content. [No]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [N/A]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [N/A]