# Learning Deep-Latent Hierarchies by Stacking Wasserstein Autoencoders

**Benoit Gaujac**
University College London

**Ilya Feige**
Faculty

**David Barber**
University College London

## Abstract

Probabilistic models with hierarchical-latent-variable structures provide state-of-the-art results amongst non-autoregressive, unsupervised density-based models. However, the most common approach to training such models based on Variational Autoencoders (VAEs) often fails to leverage deep-latent hierarchies; successful approaches require complex inference and optimisation schemes. Optimal Transport is an alternative, non-likelihood-based framework for training generative models with appealing theoretical properties, in principle allowing easier training convergence between distributions. In this work we propose a novel approach to training models with deep-latent hierarchies based on Optimal Transport, without the need for highly bespoke models and inference networks. We show that our method enables the generative model to fully leverage its deep-latent hierarchy, avoiding the well known "latent variable collapse" issue of VAEs; therefore, providing qualitatively better sample generations as well as more interpretable latent representation than the original Wasserstein Autoencoder with Maximum Mean Discrepancy divergence.

## 1. Introduction

Probabilistic latent-variable modelling is widely applicable in machine learning as a method for discovering structure directly from large, unlabelled datasets. Variational Autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) have proven to be effective for training generative models parametrised by powerful neural networks, by mapping the data into a low-dimensional embedding space. While allowing the training of expressive models, VAEs often fail when using deeper hierarchies with several stochastic latent layers.

In fact, many of the most successful probabilistic latent-variable models use only one stochastic latent layer. Auto-regressive models (Larochelle & Murray, 2011; Van Den Oord et al., 2016a;b; Salimans et al., 2017), for example, produce state-of-the-art samples and likelihood scores. However, auto-regressive models suffer from poor scalability to high-dimensional data. Flow-based models (Rezende & Mohamed, 2015; Kingma et al., 2016; Dinh et al., 2016; Kingma & Dhariwal, 2018) are another class of generative models providing competitive sample quality and are able to scale to higher-dimensional data, but they still lag behind auto-regressive models in terms of likelihood scores.

Deep-latent-variable models are highly expressive models that aim to capture the structure of data in a hierarchical manner, and could thus potentially compete with auto-regressive models for state-of-the-art performance. However, they remain hard to train. Many explanations have been proposed for this, from the use of dissimilarity measure directly in the pixel space (Larsen et al., 2016) resulting in poor sample quality, to the lack of efficient representation in the latent (Zhao et al., 2017), to simply the poor expressiveness of the models used (Zha et al., 2017; Maaløe et al., 2019).

Solutions for training deep-latent-variable models range from introducing auxiliary variables to increase the expressiveness of the posterior distribution (Maaløe et al., 2016) to replacing the simple spherical Gaussian prior with richer prior distributions which can be learned jointly with the generative model (Tomczak & Welling, 2018; Klushyn et al., 2019). Other approaches focus on building and training complex generative models and inference networks introducing latent skip connections in the generative model and inference network (Bachman, 2016), sharing generative model and inference network parameters (Sønderby et al., 2016), as well as bidirectional inference networks (Maaløe et al., 2019). Maaløe et al. (2019) managed to train very deep-hierarchical-latent models achieving near state-of-the-art sample generations, both in term of likelihood score and sample quality. However, in order to leverage their latent hierarchy (working in the VAE framework), they needed both complex, tailored inference networks, and deterministic skip connections in the generative model.

Optimal Transport (OT) (Villani, 2008) is a mathematical framework to compute distances between distributions. Recently, it has been used as a training method for generative models (Genevay et al., 2018; Bousquet et al., 2017). Tolstikhin et al. (2018) introduced Wasserstein Autoencoders (WAEs), where as with VAEs, an encoding distribution maps the data into a latent space, aiming to learn a low-dimensional representation of the data. WAEs provide a non-likelihood based framework with appealing topological properties (Arjovsky et al., 2017; Bousquet et al., 2017), in theory allowing for easier training convergence between distributions. Gaujac et al. (2018) trained a two-latent-layer generative model using a WAE, showing promising results in the capacity of the WAE framework to leverage a latent hierarchy relative to the equivalent VAE.

Following these early successes, we propose to train deep-hierarchical-latent models using the WAE framework, without the need for complex dependency paths in both the generative model and inference network. As in the works of Sønderby et al. (2016) and Maaløe et al. (2019), we believe that a deep-latent hierarchy offers the potential to improve generative models, if they could be trained properly. In order to leverage the deep-latent hierarchy, we derive a novel objective function by stacking WAEs, introducing an inference network at each level, and encoding the information up to the deepest layer in a bottom-up way. For convenience, we refer to our method as StackedWAE.

Our contributions are two-fold: first, we introduce StackedWAE, a novel objective function based on OT, designed specifically for generative modelling with latent hierarchies, and show that it is able to fully leverage its hierarchy of latents. Second, we show that StackedWAE performs significantly better when training hierarchical-latent models than the original WAE framework regularising the latent with the Maximum Mean Discrepancy (MMD) divergence (Gretton et al., 2012).

## 2. Stacked WAE

StackedWAEs are probabilistic-latent-variable models with a deep hierarchy of latents. They can be minimalistically simple in their inference and generative models, but are trained using OT in a novel way. We start by defining the probabilistic models considered in this work, then introduce OT, and finally discuss how to train probabilistic models with deep-latent hierarchies using OT, the method that we refer to as StackedWAE.

Throughout this paper, we use calligraphic letters (e.g. $\mathcal{X}$) for sets, capital letters (e.g. $X$) for random variables, and lower case letters (e.g. $x$) for their realisation. We denote probability distributions with capital letters (e.g. $P(X)$) and their densities with lower case letters (e.g. $p(x)$).

### 2.1. Generative models with deep latent hierarchies

We will consider deep-generative models with Markovian hierarchies in their latent variables. Namely, where each latent variable depends exclusively on the previous one. Denoting by $P_\Theta$ the parametric model with $N$ latent layers, where $\Theta = \{\theta_1, \dots, \theta_N\}$, we have:

$$p_\Theta(x, z_{1:N}) = p_{\theta_1}(x|z_1)\left[\prod_{n=2}^{N} p_{\theta_n}(z_{n-1}|z_n)\right]p_{\theta_N}(z_N) \tag{1}$$

where the data is $X \in \mathcal{X}$ and the latent variables are $Z_n \in \mathcal{Z}_n$ and we chose $p_{\theta_N}(z_N) = \mathcal{N}(z_N; 0_{\mathcal{Z}_N}, \mathcal{I}_{\mathcal{Z}_N})$. The corresponding graphical model for $N = 3$ is given Figure 1a.

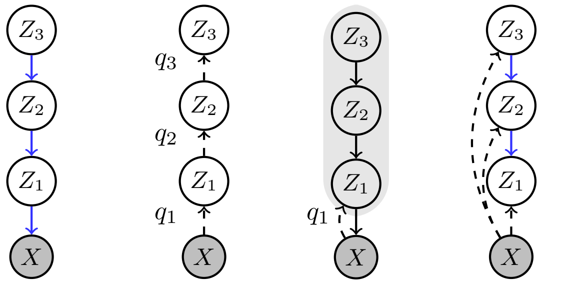
We will be using variational inference through the WAE framework of Tolstikhin et al. (2018), introducing variational distributions, $q_\Phi(z_1, \dots, z_N|x)$, to approximate the intractable posterior, analogous to VAEs. It will be shown in Section 2.3 that without loss of generality, $q_\Phi(z_1, \dots, z_N|x)$ can have a Markovian latent hierarchy when following the StackedWAE approach. That is, without loss of generality,

$$q_\Phi(z_{1:N}|x) = q_{\phi_1}(z_1|x)\prod_{n=2}^{N} q_{\phi_n}(z_n|z_{n-1}) \tag{2}$$

where each $q_{\phi_n}(z_n|z_{n-1})$ is introduced iteratively by stacking WAEs at each latent layer. The corresponding graphical model for $N = 3$ is given Figure 1b.

We focus on this simple Markovian latent-variable structure for the generative model as a proof point for StackedWAE. This simple modelling setup is famously difficult to train, as is discussed extensively in the VAE framework (see for example Burda et al. (2015); Sønderby et al. (2016); Zhao et al. (2017)). The difficulty in training such models comes from the Markovian latent structure of the generative model; in particular, the difficulty of learning structure in the deeper latents. This is because, to generate samples $x \sim p_\Theta(x)$, only the joint $p_\Theta(x, z_1)$ is needed as $p_\Theta(x) = \int_{\mathcal{Z}_1} p_{\theta_1}(x|z_1)p_\Theta(z_1)dz_1$. Learning a smooth structure in each latent layer is not a strict requirement for learning a good generative model, however, it is sought after if the latent is to be used downstream or interpreted. We find empirically (see Section 3.1) that a better generative model is also achieved when the latent hierarchy is well learnt all the way down.

Sønderby et al. (2016) sought to overcome the difficulty of learning a deep-latent hierarchy by using deterministic bottom-up inference, followed by top-down inference that shared parameters with the generative model. With additional optimisation tricks (e.g. KL annealing), their deeper-latent distributions would still go unused for sufficiently deep-latent hierarchies (discussed in Maaløe et al. (2019)). In order to get deeper hierarchies of latents, Maaløe et al. (2019) introduced additional deterministic connections in the generative model as well as bidirectional inference



(a) Our generative model (b) Our inference model (c) Comparison: inference, no stacking (d) Comparison: alternative inference

Figure 1: (a) Generative model (blue lines represent generative model parameters). (b) Inference model used in STACKEDWAE. (c) Standard WAE; the generative model has only one latent with prior $p(z_1) = \int p(z_1|z_2)p(z_2|z_3)p(z_3)dz_1dz_2dz_3$. (d) Inference model with skips connections and parameter sharing with the generative model, as in Sønderby et al. (2016).

network to facilitate the deep information flow needed to ensure the usage of the deeper-latent layers.

While the approach in Maaløe et al. (2019) is well motivated and achieves excellent results, we choose the OT framework for training deep-latent models due to its topological properties (see Arjovsky et al. (2017)). Still, the standard WAE encounters the same difficulties as the VAE in learning deep-latent hierarchies. We thus modify the original WAE objective, effectively stacking WAEs, to improve the learning of both the generative model and the inference distribution throughout the latent hierarchy.

## 2.2. Wasserstein Autoencoders

The Kantorovich formulation of the OT between the true-but-unknown data distribution $P_D$ and the model distribution $P_\Theta$, for a given cost function $c$, is defined by:

$$\mathrm{OT}_c(P_D, P_\Theta) = \inf_{\Gamma\in\mathcal{P}(P_D,P_\Theta)} \int_{\mathcal{X}\times\mathcal{X}} c(x,\tilde{x})\, d\Gamma(x,\tilde{x}) \quad (3)$$

where $\mathcal{P}(P_D, P_\Theta)$ is the space of all couplings of $P_D$ and $P_\Theta$; namely, the space of joint distributions $\Gamma$ on $\mathcal{X}\times\mathcal{X}$ whose densities $\gamma$ have marginals $p_D$ and $p_\Theta$:

$$\begin{aligned}&\mathcal{P}(P_D, P_\Theta) \quad (4)\\ &= \left\{\Gamma \,\Big|\, \int_{\mathcal{X}}\gamma(x,\tilde{x})\, d\tilde{x} = p_D(x), \int_{\mathcal{X}}\gamma(x,\tilde{x})\, dx = p_\Theta(\tilde{x})\right\}\end{aligned}$$

In the WAE framework, the space of couplings is constrained to joint distributions $\Gamma$, with density $\gamma$, of the form:

$$\gamma(x,\tilde{x}) = \int_{\mathcal{Z}_{1:N}} p_\Theta(\tilde{x}|z_{1:N})\, q_{\phi_1}(z_{1:N}|x)\, p_D(x)\, dz_{1:N} \quad (5)$$

where $q_{\phi_1}(z_{1:N}|x)$, for $x \in \mathcal{X}$, plays the same role as the variational distribution in variational inference.

Marginalising over $\tilde{x}$ in Eq. (5) automatically gives $p_D(x)$, thus satisfying the first marginal constraint of Eq. (4). However the second marginal constraint in Eq. (4) (that over $x$ giving $p_\Theta$) is not guaranteed. Due to the Markovian structure of the generative model, a sufficient condition for satisfying the second marginal constraint in Eq. (4) for all $z_1 \in \mathcal{Z}_1$, is (more detail provided in Appendix A.1):

$$\int_{\mathcal{X}} q_{\phi_1}(z_1|x)\, p_D(x)\, dx = p_{\Theta_{2:N}}(z_1) \quad (6)$$

where we denote $p_{\Theta_{2:N}}(z_1)$ the prior over $Z_1$:

$$p_{\Theta_{2:N}}(z_1) \overset{\text{def}}{=} \int_{\mathcal{Z}_{2:N}} \left[\prod_{n=2}^{N} p_{\theta_n}(z_{n-1}|z_n)\right] p_{\theta_N}(z_N)\, dz_{2:N} \quad (7)$$

Finally, to get the WAE objective of Tolstikhin et al. (2018), the constraint in Eq. (6) is relaxed using a Lagrange multiplier (more detail provided in Appendix A.2):

$$\begin{aligned}\widehat{W}_c(P_D, P_\Theta) = \inf_{Q_{\phi_1}(Z_1|X=x)} \Bigg[&\int_{\mathcal{X}\times\mathcal{X}} c(x,\tilde{x})\,\gamma(x,\tilde{x})\, dx d\tilde{x}\\ &+ \lambda_1\, \mathcal{D}_1\Big(Q_1^{\text{agg}}(Z_1),\, P_{\Theta_{2:N}}(Z_1)\Big)\Bigg] \quad (8)\end{aligned}$$

where $\mathcal{D}_1$ is any divergence function, $\lambda_1$ a relaxation parameter, $\gamma$ is defined in Eq. (5), and $Q_1^{\text{agg}}(Z_1)$ is the aggregated posterior:

$$Q_1^{\text{agg}}(Z_1) \overset{\text{def}}{=} \int_{\mathcal{X}} Q_{\phi_1}(Z_1|x)\, p_D(x)\, dx \quad (9)$$

Note that because of the Markovian latent structure of the generative model,

$$\begin{aligned}\gamma(x,\tilde{x}) &= \int_{\mathcal{Z}_{1:N}} p_\Theta(\tilde{x}|z_{1:N})\, q_{\phi_1}(z_{1:N}|x)\, p_D(x)\, dz_{1:N}\\ &= \int_{\mathcal{Z}_1} p_{\theta_1}(\tilde{x}|z_1)\, q_{\phi_1}(z_1|x)\, p_D(x)\, dz_1 \quad (10)\end{aligned}$$

Eq. (9) and (10) do not depend on $z_{>1}$, so the infimum in Eq. (8) is taken only over $q_{\phi_1}(z_1|x)$ instead of the full $q_{\phi_1}(z_1, \ldots, z_N|x)$.

While Eq. (8) is in-principle tractable (e.g. for Gaussian $q_{\phi_1}(z_1|x)$ and sample-based divergence function such as

MMD), it only depends on the first latent $Z_1$. Thus it will learn only a good approximation for the joint $p_\Theta(x, z_1) = p_{\theta_1}(x|z_1)p(z_1)$, rather than a full hierarchy with each latent living on a smooth manifold. We show empirically in Section 3.1 that Eq. (8) is indeed insufficient for learning to use the full hierarchy of latents.

### 2.3. Stacking WAEs for deep latent-variable modelling

In the limit $\lambda_1 \to \infty$, Eq. (8) does not depend on the choice of divergence $\mathcal{D}_1$. However, given the set of approximations used, a divergence that takes into account the smoothness of the full stack of latents will likely help the optimisation. We now explain how, by using the Wasserstein distance itself for $\mathcal{D}_1$, we can derive an objective that naturally pairs up inference and generation at every level in the deep-latent hierarchy. After all, the divergence in Eq. (8) is between the intractable aggregate distribution $Q_1^{\text{agg}}(Z_1)$ from which we can only access samples, and an analytically-known distribution $P_{\Theta_{2:N}}(Z_1)$, which is analgous to where we started with the Wasserstein distance between $P_D$ and the full $P_\Theta$.

Specifically, we choose $\mathcal{D}_1$ in Eq. (8) to be the relaxed Wasserstein distance $\widehat{W}_{c_1}$, which following the same arguments as before, requires the introduction of a new variational distribution $Q_{\phi_2}(Z_2|Z_1)$:

$$\mathcal{D}_1\Big(Q_1^{\text{agg}}(Z_1),\, P_{\Theta_{2:N}}(Z_1)\Big) = \inf_{Q_{\phi_2}(Z_2|Z_1=z_1)} \Bigg[\lambda_2\, \mathcal{D}_2\Big(Q_2^{\text{agg}}(Z_2),\, P_{\Theta_{3:N}}(Z_2)\Big) + \int_{\mathcal{Z}_1\times\mathcal{Z}_1} c_1(z_1,\tilde{z}_1)\,\gamma_1(z_1,\tilde{z}_1)\,dz_1 d\tilde{z}_1\Bigg] \tag{11}$$

where we re-used the notation of Eq. (6) for the prior $P_{\Theta_{3:N}}$, and similarly with Eq. (9) for the aggregated posterior,

$$Q_2^{\text{agg}}(Z_2) \overset{\text{def}}{=} \int_{\mathcal{Z}_1} Q_{\phi_2}(Z_2|z_1)\, q_1^{\text{agg}}(z_1)\, dz_1 \tag{12}$$

The joint is given by

$$\gamma_1(z_1,\tilde{z}_1) \overset{\text{def}}{=} \int_{\mathcal{Z}_2} p_{\theta_2}(\tilde{z}_1|z_2)\, q_{\phi_2}(z_2|z_1)\, q_1^{\text{agg}}(Z_1)\, dz_2 \tag{13}$$

And as before, $Q_{\phi_2}(Z_2|Z_1)$ in the inf does not need to provide a distribution over the $z_{>2}$ without loss of generality.

The divergence $\mathcal{D}_1$ that arose in Eq. (8) between two distributions over $Z_1$ is thus mapped onto the latent at the next level in the latent hierarchy, $Z_2$, via Eq. (11). This process can be repeated by using $\widehat{W}_{c_2}$ again for $\mathcal{D}_2$ in Eq. (11) to get an expression that maps to $Z_3$, requiring the introduction of another variational distribution $Q_{\phi_3}(Z_3|Z_2)$. Repeating this process until we get to the final layer of the hierarchical-latent-variable model gives the StackedWAE objective:

$$W_{\text{StackedWAE}}(P_D, P_\Theta) = \inf_{Q_\Phi(Z_1,\ldots,Z_N|X=x)} \Bigg[\Bigg[\prod_{i=1}^{N}\lambda_i\Bigg]\mathcal{D}_N\Big(Q_N^{\text{agg}}(Z_N),\, P(Z_N)\Big) + \sum_{n=0}^{N-1}\Bigg[\prod_{i=1}^{n}\lambda_i\Bigg]\int_{\mathcal{Z}_n\times\mathcal{Z}_n} c_n(z_n,\tilde{z}_n)\,\gamma_n(z_n,\tilde{z}_n)\,dz_n\, d\tilde{z}_n\Bigg] \tag{14}$$

where we denote $(\mathcal{Z}_0, Z_0, z_0) = (\mathcal{X}, X, x)$ and we define the empty product $\prod_{i=1}^{0}\lambda_i \overset{\text{def}}{=} 1$. Similarly to Eq. (11), we defined the joints $\gamma_n$ as

$$\gamma_n(z_n,\tilde{z}_n) \overset{\text{def}}{=} \int_{\mathcal{Z}_{n+1}} p_{\theta_{n+1}}(\tilde{z}_n|z_{n+1})\, q_{\phi_{n+1}}(z_{n+1}|z_n)\, q_n^{\text{agg}}(z_n)\, dz_{n+1} \tag{15}$$

Each $p_{\theta_n}$ is the nth layer of the generative model given in Eq. (1). The $q_{\phi_n}$'s are the inference models introduced each time a WAE is "stacked", which combine to make the overall StackedWAE Markovian inference model given in Eq. (2), and the aggregated posterior distributions are defined as

$$Q_n^{\text{agg}}(Z_n) \overset{\text{def}}{=} \int_{\mathcal{Z}_{n-1}} Q_{\phi_n}(Z_n|z_{n-1})\, q_{n-1}^{\text{agg}}(z_{n-1})\, dz_{n-1} \tag{16}$$

with $Q_0^{\text{agg}} \overset{\text{def}}{=} P_D$.

Note that the StackedWAE objective function in Eq. (14) still requires the specification of a divergence function at the highest latent layer $\mathcal{D}_N$, which we simply take to be the MMD as originally proposed by Tolstikhin et al. (2018). Other choices can be made, as in Patrini et al. (2018), who choose a Wasserstein distance computed using the Sinkhorn algorithm (Cuturi, 2013). While Patrini et al. (2018) provide a theoretical justification for the minimisation of a Wasserstein distance in the prior space, we found that it did not result in significant improvement and comes at an extra efficiency cost. Similarly, one could choose different cost functions $c_n$ at each layer; for simplicity we take all cost functions to be the squared Euclidean distance in their respective spaces.

## 3. Experiments

We now show how the StackedWAE approach of Section 2 can be used to train deep-latent hierarchies without customizing the generative or inference models (e.g. no skip connections, no parameter sharing). We also show through explicit comparison that the StackedWAE approach performs significantly better than the standard WAE when training deep-hierarchical-latent models.

## 3.1. MNIST

### Experimental setup

We trained a deep-hierarchical-latent variable model with $N = 5$ latent layers on raw (non-binarised) MNIST (LeCun & Cortes, 2010). The latent layers have dimensions: $d_{\mathcal{Z}_1} = 32$, $d_{\mathcal{Z}_2} = 16$, $d_{\mathcal{Z}_3} = 8$, $d_{\mathcal{Z}_4} = 4$ and $d_{\mathcal{Z}_5} = 2$. We chose Gaussian distributions for both the generative and inference models, except for the bottom layer of the generative model, which we choose to be deterministic, as in Tolstikhin et al. (2018). The mean and covariance matrices used are parametrised by fully connected neural networks similar to that of Sønderby et al. (2016). Full details of the experimental setup is given in Appendix B.1.

### Learning a deep-latent hierarchy

Our results are shown in Figure 2, with samples from the generative model in Figure 2a, and latent space interpolations in Figure 2b where digits are reconstructed from points in $\mathcal{Z}_5$ taken evenly in a grid varying between $\pm 2$ standard deviations from the origin. StackedWAE generates compelling samples and learns a smooth manifold (see Figure 7, top-left, and Figure 8b of Appendix B.1 for additional samples and latent interpolations respectively). Note that the choice $d_{\mathcal{Z}_5} = 2$ allows for easy visualisation of the learned manifold, rather than being the optimal dimension for capturing all of the variance in MNIST.

Figure 2b shows that StackedWAE manages to use all of its latent layers, capturing most of the covariance structure of the data in the deepest-latent layer, which is something that VAE methods struggle to accomplish (Sønderby et al., 2016; Zhao et al., 2017). Figure 3a shows the encoded input images through the latent layers, with corresponding digit labels coloured. We see through each layer that StackedWAE leverages the full hierarchy in its latents, with structured manifolds learnt at each stochastic layer. Note that for the shallower layers with higher dimensions (*e.g.* $d_{\mathcal{Z}_1} = 32$ or $d_{\mathcal{Z}_2} = 16$), the PCA algorithm results in a poor visualisation. For such high dimensional spaces, other visualisation techniques can be used such as UMAP (McInnes & Healy, 2018) as done in Figure 8a of Appendix B.1.

An advantage of deep-latent hierarchies is their capacity to capture information at different levels, augmenting a single-layer latent space. In Figure 3b, input images, shown in the bottom-row, are encoded and reconstructed for each latent layer. More specifically, the inputs are encoded up to the latent layer $i$ and reconstructed from the encoded $z_i$ using the generative model $p(x|z_1)p(z_1|z_2)\dots p(z_{i-1}|z_i)$. Row $i$ in Figure 3b shows the reconstruction obtained from encoding up to layer $i$. We can see that each additional encoding layer moves slightly farther away from copying the input image as it moves towards fitting the encoding into the

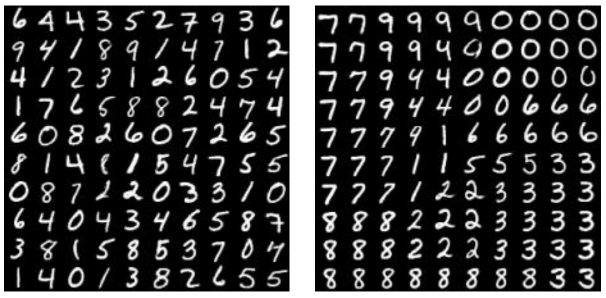
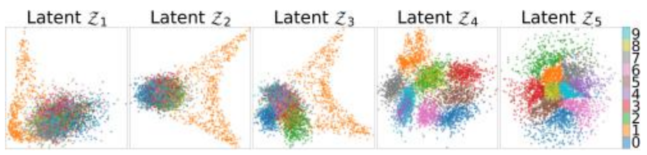

(a) (b)

Figure 2: 5-layer StackedWAE. (a) Model samples. (b) $\mathcal{Z}_5$ latent-space interpolations in a 2-dimensional grid varying between $\pm 2$ standard deviations from the origin.



(a)

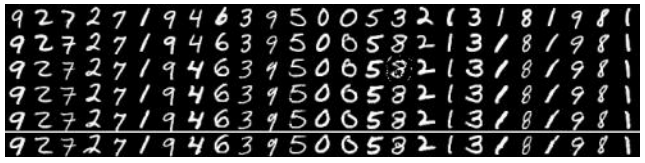

(b)

Figure 3: 5-layer StackedWAE. (a) Visualisations of latent spaces $\mathcal{Z}_i$. Each colour corresponds to a digit label. $d_{\mathcal{Z}_5} = 2$ can be directly plotted; for higher dimensions we use PCA. (b) Reconstructions for different encoding layers. The bottom row is data; the $i^{\text{th}}$ row from the bottom is generated using the latent codes $z_i$ which are from the $i^{\text{th}}$ encoding layer.

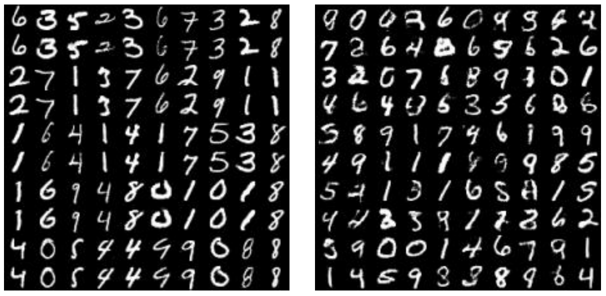

(a) (b)

Figure 4: 1-layer implicit-prior WAE. (a) Reconstructions (within pairs of rows, data is above with the corresponding reconstructions below). (b) $\mathcal{Z}_5$ latent-space interpolations.

2-dimensional, unit-normal prior distribution. The dimensionality of the deeper-latent layers is a modelling choice which determines how much information is preserved; this can be seen through the loss of information from deeper reconstructions in Figure 3b (for a better visualisation of the *real* model reconstructions, corresponding to the top-row in Figure 3b, see Figure 7, bottom-left, of Appendix B.1). Indeed, in each layer, the encoder is asked to map the incoming encodings into a lower-dimensional latent space, filtering the amount of information to keep and pass up to the deeper layer. Thus, if there is a mismatch in the dimensions between the true underlying generative process of the data and the chosen model, the encoder will have to project the encodings into lower-dimensional space, losing information along the way.

### Ablation study: StackedWAE versus WAE

In this section, we compare StackedWAE with the original WAE framework for training generative models with deep-hierarchical latents. In particular, we train a WAE using the objective defined in Eq. (8) and an inference distribution as in Eq. (2); the corresponding graphical model is shown in Figure 1c. We use the same the experimental setup as outlined earlier in this section, and the same parametrised networks. This experiment can be related to the work of Tomczak & Welling (2018); Klushyn et al. (2019) in the VAE framework case, as we can rephrase it as training a plain-vanilla 1-layer WAE whose prior over the latent variable is defined and parametrised as in Eq. (7), and learned alongside the inference network and the generative model. However, we do not use any specific optimisation scheme nor do we constraint the structure of the prior beside the modelling choices made in Section 2.1, as opposed to what is done in Tomczak & Welling (2018); Klushyn et al. (2019).

The results are shown in Figure 4, with reconstructions in Figure 4a, and deepest-latent interpolations in Figure 4b. The latter two should be compared directly with bottom and top-row of Figure 3b and Figure 2b, respectively, for the StackedWAE (see also Figure 7 of Appendix B.1 for samples and reconstructions comparisons with StackedWAE). The samples generated with the standard WAE when training deep-hierarchical-latent variable models (Figure 7, top-right, of Appendix B.1) are poor in comparison with those of the StackedWAE (Figure 7, top-left, of Appendix B.1).

The lack of smooth interpolations in Figure 4b shows that almost no structure has been captured in the deepest-latent layer. This is likely due to the fact that the standard WAE, with objective given in Eq. (8), is independent of the deeper-latent inference distributions, thus weakening the smoothness constraint in the deeper layers. The relatively accurate reconstructions in Figure 4a indicate that the model only needs the first latent layer to capture most of the structure of the data. This behaviour is similar to that of the Markov HVAE as described in Zhao et al. (2017). They show that, for Markov HVAEs to learn interpretable latents, one needs additional constraints beyond simply maximising the likelihood, or in our case, the WAE objective of Tolstikhin et al. (2018) with MMD divergence.

## 3.2. Real world Datasets

We now turn to more realistic datasets to show that StackedWAE is still able to leverage a deep-latent hierarchy. In particular, we trained a 6-layer and a 10-layer StackedWAE on Street View House Numbers (SVHN) (Netzer et al., 2011) and CelebA (Liu et al., 2015), respectively. Our goal in this section, is to train very deep-latent variable models such as the ones in Maaløe et al. (2019), and show how StackedWAE still manages to use the deepest-latent layers of the generative model. It is worth stating that we are not aiming to achieve state-of-the-art performance on these datasets, but rather to show that it is easy to learn a deep-hierarchical latent representation of the data.

Our implementation choices resulted in relatively high-dimensional latent spaces (see below and Appendix B.2 for more details). Rubenstein et al. (2018) observed that when training WAEs with high-dimensional latent spaces, the variance of the inference network tends to collapse to $0$. The authors argue that this might come either from an optimisation issue or from the failing of the divergence used to regularise the latents. Either way, the collapse to deterministic encoders results in poor sample quality as the deterministic encoder is being asked to map the input manifold into a higher-dimensional space than its intrinsic dimension. They proposed to explicitly include a regulariser that maintains a non-zero variance in Eq. (14). As in Rubenstein et al. (2018), we included a log-variance penalty term given by Eq. (17):

$$\mathcal{L}_{\text{pen}} = \sum_{i=1}^{N} \lambda_i^{\Sigma} \sum_{m=1}^{d_{\mathcal{Z}_i}} |\log \Sigma_i^q[m]| \qquad (17)$$

Where $\Sigma_i^q$ is the covariance matrix of the $n^{\text{th}}$ inference network. We find that an exponentially decreasing penalty term $\lambda_i^{\Sigma}$ (see following sections) works well for the dataset at hand with our experimental setup. This choice is motivated by the fact that the bigger the latent dimension (shallower latent layers in the hierarchy), the more likely it is that the latent dimension is larger than the data's intrinsic dimension.

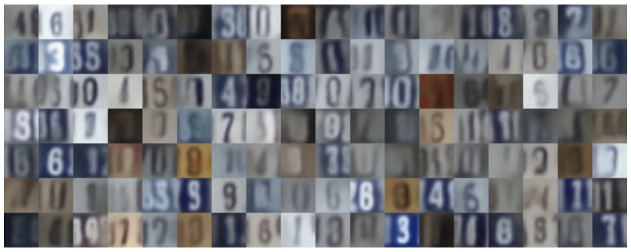
### Street View House Numbers

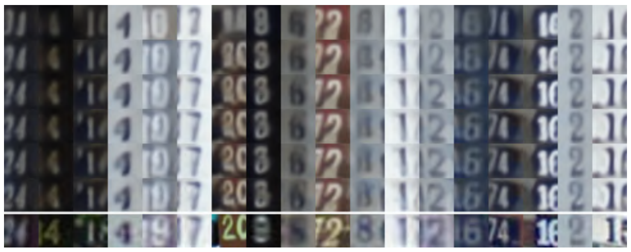
We trained a 6-layer StackedWAE with Gaussian inference networks and generative models at each latent layer, with mean and covariance functions parametrised by 3-layer ResNet-style neural networks (Kaiming et al., 2015). The details are given in Appendix B.2. These architectures dictate the dimensionality of the latent spaces, with the latent dimension given by the size of the output feature map at that layer times the number of these feature maps. In this experiment, networks $i = 1, 3, 5$ have stride 2, effectively performing downsampling and upsampling operations in the bottom-up path and the top-down path respectively, resulting in latent dimensions (width $\times$ height $\times$ feature map) of: 16×16×2, 16×16×1, 8×8×2, 8×8×1, 4×4×2 and 4×4×1. For the regularisation hyperparameters, we use $\prod_{i=1}^{n} \lambda_i = \lambda_{\text{rec}}^{(n-1)+1}$ for $n = 1, \dots, 5$ (each reconstruction term in the objective), and $\prod_{i=1}^{6} \lambda_i = \lambda_{\text{match}}$ (the final divergence term). The choice for the reconstruction weights is motivated by the fact that the effective regularisation hyperparameters scale exponentially. Thus, to avoid the collapse (or blowing up for $\lambda_{\text{rec}} > 1$) of the corresponding reconstruction terms, we choose the weights to scale like $\mathcal{O}\big(\lambda_{\text{rec}}^{n}\big)$. We found that $\big(\lambda_{\text{rec}}, \lambda_{\text{match}}\big) = \big(0.5, 10\big)$ worked well with our experimental setup. As mentioned above, in order to avoid the collapse to deterministic encoder networks, we penalised the log-variance of the inference networks (as done in Rubenstein et al. (2018)). We found that $\lambda_i^{\Sigma} = \lambda^{\Sigma} \cdot \mathrm{e}^{-(i-1)}$, for $i = 1, \dots, 6$, with $\lambda^{\Sigma} = 2.5$ worked well in our setting. More details of the experimental setup can be found in Appendix B.2.

Our results on SVHN are given in Figure 5. Samples from the generative model are shown in Figure 5a. Figure 5b shows the reconstructions of the data points (along the bottom row) at each latent layer in the hierarchy. Similarly to the results for MNIST, we can see that the deepest latent layer may not be large enough to enable high-fidelity reconstructions. Our intention is to show that the hierarchy of latents can be learnt, which is clearly the case, rather than to model SVHN perfectly, so we do not attempt to tune to the optimal latent dimensionality. Figure 5c represents point interpolation, with the actual anchor data points shown in the first and last column, their reconstructions in the second, respectively second-to-last, columns, and the latent interpolation in-between.

## CelebA

We trained a 10-layers StackedWAE. Similarly to SVHN, the inference and generative networks are fully-convolutional ResNet. The inference networks and generative models, $i = 1, 4, 7, 10$ have stride 2. These downsampling, respectively upsampling, operations resulted in 4 groups of latent spaces sharing features maps of the same spatial dimensions: latent layers $i = 1, 2, 3$ with output features maps of size 32×32, layers $i = 4, 5, 6$ with features maps of size 16×16, layers $i = 7, 8, 9$ with size 8×8 and layer $i = 10$ with size 4×4. Within each group, the number of feature maps is decreased by two as we go up the hierarchy, with the biggest layer having


(a)


(b)

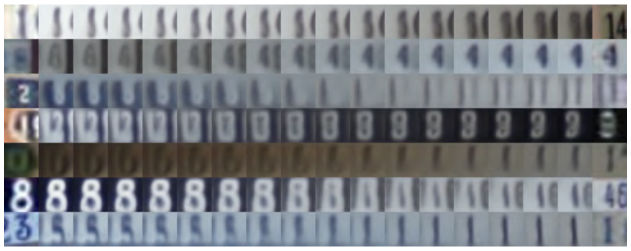

(c)

Figure 5: 6-layer StackedWAE. (a) Model samples. (b) Reconstructions for different encoding layers, as in Figure 3b. (c) Points interpolations; the first and last columns are actual data points, with corresponding reconstructions shown in the second, respectively second-to-last, columns.

8 features maps and the smallest 4. As with SVHN, we used an exponentially decreasing regularisation parameters. More precisely, we chose $\prod_{i=1}^{p} \lambda_i = \lambda_{\text{rec}}^{(p-1)/3+1}$ for $p = 1, \ldots, 9$ and $\prod_{i=1}^{10} \lambda_i = \lambda_{\text{match}}$. For this dataset, choosing $\left(\lambda_{\text{rec}}, \lambda_{\text{match}}\right) = \left(10^{-1}, 10^{0}\right)$ worked well. We also added a $L_1$ penalty on the log-covariance matrices of the encoders to avoid any variance collapse. We used the same penalisation scheme than in the previous experiment, that is $\lambda_i^{\Sigma} = \lambda^{\Sigma} \cdot \mathrm{e}^{-(i-1)}$, for $i = 1, \ldots, 10$, with $\lambda^{\Sigma} = 2.5$. See Appendix B.2 for more details about the experimental setup.

Results are shown Figure 6, with model samples given in Figure 6a and layer-wise reconstructions in Figure 6b. As with the 6-layer StackedWAE trained on SVHN, Figure 6b shows that StackedWAE manages to use all its latent layers, up to the deepest ones. While the full reconstructions, shown in the top-row of Figure 6b, are relatively close to the original data points, shown in the bottom-row, we can notice, as in the MNIST and SVHN experiments, a loss of information as we go deeper in the hierarchy. In other words, the deeper the encoding, the smoother or blurrier the reconstructions. Here again, one possible explanation is the excessive filtering of information by the encoder when going up in the hierarchy due to the miss-match between the intrinsic dimension and the latent dimension.

## 4. Conclusion

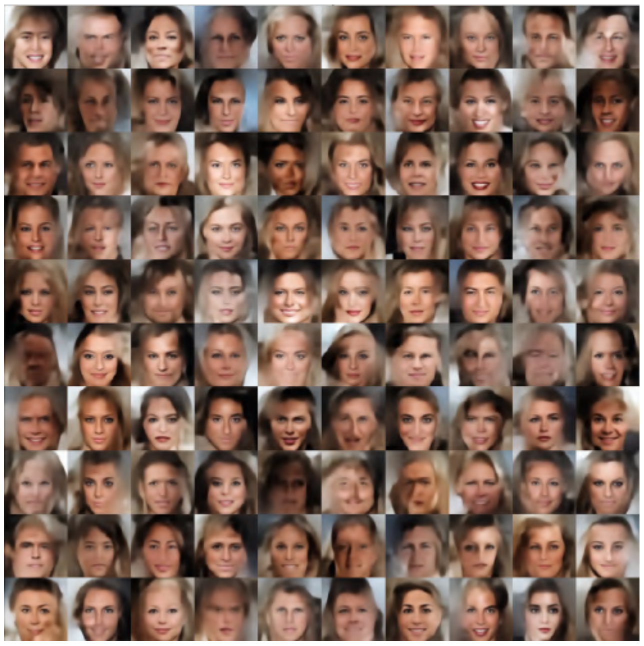
In this work we introduced a novel objective function for training generative models with deep hierarchies of latent variables using Optimal Transport. Our approach recursively applies the Wasserstein distance as the regularisation divergence, allowing the stacking of WAEs for arbitrarily deep-latent hierarchies. We showed that this approach enables the learning of smooth latent distributions even in deep latent hierarchies, which otherwise requires extensive model design and tweaking of the optimisation procedure to train. We also showed that our approach is significantly more effective at learning smooth hierarchical latents than the standard WAE.

## References

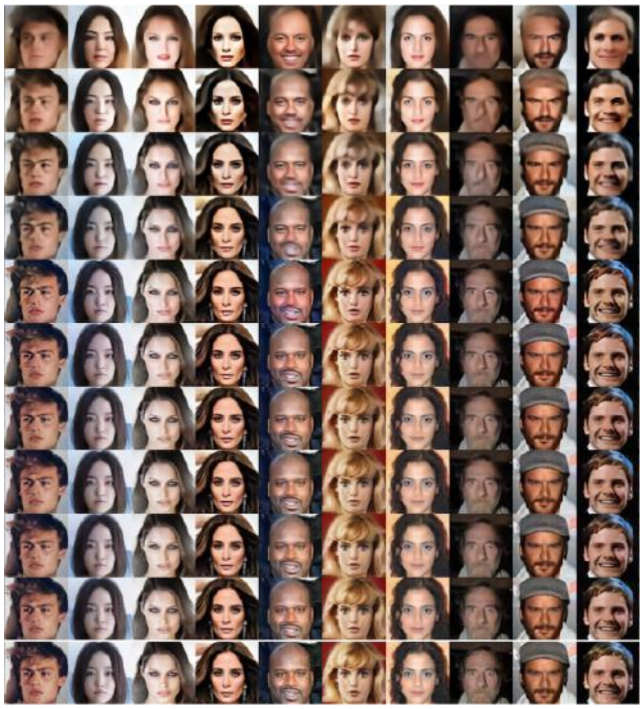
Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, 2017.

Bachman, P. An architecture for deep, hierarchical generative models. In *Advances in Neural Information Processing Systems*, 2016.

Bousquet, O., Gelly, S., Tolstikhin, I., Simon-Gabriel, C. J., and Schölkopf, B. From optimal transport to generative modeling: the VEGAN cookbook. *arXiv:1705.07642*, 2017.



(a)



(b)

Figure 6: 10-layer StackedWAE. (a) Model samples. (b) Reconstructions for different encoding layers, as in Figure 3b.

Burda, Y., R., G., and Salakhutdinov, R. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2015.

Cuturi, M. Sinkhorn distances: lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, 2013.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *International Conference on Learning Representations*, 2016.

Gaujac, B., Feige, I., and Barber, D. Gaussian mixture models with Wasserstein distance. *arXiv:1806.04465*, 2018.

Genevay, A., Peyre, G., and Cuturi, M. Learning generative models with Sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, 2018.

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2011.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. In *Journal of Machine Learning Research*, 2012.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Kaiming, H., Xiangyu, Z., Shaoqing, R., and Jian, S. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Kingma, D. P. and Ba, J. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 2016.

Klushyn, A., Chen, N., Kurle, R., Cseke, B., and van ver Smagt, P. Learning hierarchical priors in VAEs. In *Advances in Neural Information Processing Systems*, 2019.

Larochelle, H. and Murray, I. The neural autoregressive distribution estimator. In *International Conference on Artificial Intelligence and Statistics*, 2011.

Larsen, A. B. L., Sønderby S., K., Larochelle, H., and Winther, O. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning*, 2016.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision*, 2015.

Maaløe, L., Fraccaro, M., Liévin, V., and Winther, O. BIVA: a very deep hierarchy of latent variables for generative modeling. In *Advances in neural information processing systems*, 2019.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. Auxiliary deep generative models. In *International Conference on Machine Learning*, 2016.

McInnes, L. and Healy, J. UMAP: uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426*, 2018.

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

Patrini, G., Carioni, M., Forré, P., Bhargav, S., Welling, M., Van Den Berg, R., Genewein, T., and Nielsen, F. Sinkhorn autoencoders. *arXiv:1810.01118*, 2018.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Rubenstein, P. K., Schölkopf, B., and Tolstikhin, I. On the latent space of Wasserstein auto-encoders. In *Workshop track - International Conference on Learning Representations*, 2018.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. In *Advances in neural information processing systems*, 2016.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

Tomczak, J. M. and Welling, W. VAE with VampPrior. In *International Conference on Artificial Intelligence and Statistics*, 2018.

Van Den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *International Conference on Machine Learning*, 2016a.

Van Den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, 2016b.

Villani, C. *Optimal Transport: Old and New*. Springer Berlin Heidelberg, 2008.

Zha, S., Song, J., and Ermon, S. Towards deeper understanding of variational autoencoding models. *arxiv:1702.08658*, 2017.

Zhao, S., Song, J., and Ermon, S. Learning hierarchical features from deep generative models. In *International Conference on Machine Learning*, 2017.

## A. StackedWAE derivation details

### A.1. Marginal constraint

The space of couplings $\Gamma \in \mathcal{P}(P_D, P_\Theta)$ that defines the OT distance in Eq. (3) is constrained according to Eq. (4). WAE assumes a joint density of the form given in Eq. (5), which automatically satisfies the $p_D$ marginal constraint, but requires the further sufficient condition given in Eq. (6) in order to satisfy the $p_\Theta$ marginal constraint. To see that Eq. (6) is indeed a sufficient condition for the $p_\Theta$ marginal constraint, note that from the Markovian assumption of the generative model (see Eq. (1)), we can write $\gamma$ as
$\forall (x, \tilde{x}) \in \mathcal{X} \times \mathcal{X}$,

$$\begin{aligned}\gamma(x, \tilde{x}) &= \int_{\mathcal{Z}_{1:N}} p_{\theta_1}(\tilde{x}|z_1)\, q_{\phi_1}(z_{1:N}|x)\, p_D(x)\, dz_{1:N} \\ &= \int_{\mathcal{Z}_1} p_{\theta_1}(\tilde{x}|z_1)\, q_{\phi_1}(z_1|x)\, p_D(x)\, dz_1\end{aligned} \tag{18}$$

The constraint in Eq. (4) on the $p_D$ marginal is trivially true as the integral over the second variable can be brought inside the integral over $\mathcal{Z}_1$, after which all the integrals simply integrate to unity leaving $p_D$.

The constraint on the second marginal is obtained by integrating Eq. (18) over the first variable,
$\forall x \in \mathcal{X}$,

$$\begin{aligned}\int_{\mathcal{X}} \gamma(x, \tilde{x})\, dx &= \int_{\mathcal{X}} \int_{\mathcal{Z}_1} p_{\theta_1}(\tilde{x}|z_1) q_{\phi_1}(z_1|x) p_D(x) dz_1\, dx \\ &= \int_{\mathcal{Z}_1} p_{\theta_1}(\tilde{x}|z_1) \int_{\mathcal{X}} q_{\phi_1}(z_1|x) p_D(x) dx\, dz_1\end{aligned} \tag{19}$$

Thus, to satisfy Eq. (4), we need:
$\forall \tilde{x} \in \mathcal{X}$,

$$\begin{aligned}\int_{\mathcal{Z}_1} p_\Theta(\tilde{x}|z_1) \overbrace{\int_{\mathcal{X}} q_{\phi_1}(z_1|x) p_D(x) dx}^{\overset{\text{def}}{=} q_1^{\text{agg}}(z_1)}\, dz_1 \overset{\text{need}}{=} \\ \int_{\mathcal{Z}_1} p_{\theta_1}(\tilde{x}|z_1)\, p_{\Theta_{2:N}}(z_1)\, dz_1\end{aligned} \tag{20}$$

where we use the generative model defined in Eq. (1) and we introduced:
$\forall z_1 \in \mathcal{Z}_1$,

$$p_{\Theta_{2:N}}(z_1) \overset{\text{def}}{=} \int_{\mathcal{Z}_{2:N}} \prod_{n=2}^{N} p_{\theta_n}(z_{n-1}|z_n)\, p(z_N)\, dz_{2:N} \tag{21}$$

To satisfy Eq. (20), one obvious sufficient condition on the aggregated posterior distribution $Q_1^{\text{agg}}(Z_1)$ defined in Eq. (9) is Eq. (6), namely that

$$\forall z_1 \in \mathcal{Z}_1, \quad q_1^{\text{agg}}(z_1) = p_{\Theta_{2:N}}(z_1) \tag{22}$$

which is what we sought out to show. However, Eq. (22) is a sufficient condition, not a necessary one: indeed Eq. (22) must only hold under $\int_{\mathcal{Z}_1} P_\Theta(\tilde{X}|z_1) dz_1$. So for example, if $P_{\theta_1}(\tilde{X}|z_1) = P_{\theta_1}(\tilde{X})$, then Eq. (20) would boil down to a constraint only on the expectations of $Q_1^{\text{agg}}(Z_1)$ and $P_{\Theta_{2:N}}(Z_1)$.

### A.2. WAE objective

Starting from the definition of the OT distance given in Eq. (3), and using the WAE approach with density $\gamma$ written as Eq. (18), we find:

$$\begin{aligned}\text{OT}_c(P_D, P_\Theta) &= \inf_{\Gamma \in \mathcal{P}(P_D, P_\Theta)} \int_{\mathcal{X}\times\mathcal{X}} c(x, \tilde{x})\, d\Gamma(x, \tilde{x}) \\ &\leq \inf_{\substack{Q_{\phi_1}(Z_1, Z_2, \dots Z_n|X), \\ \text{Eq.(20) satisfied}}} \int_{\mathcal{X}\times\mathcal{X}} c(x, \tilde{x})\, \gamma(x, \tilde{x})\, dx\, d\tilde{x}\end{aligned}$$

where $\gamma$ is defined in Eq. (18). Given that the above does not depend on $z_{>1}$, the inf can be written over $Q_{\phi_1}(Z_1|X)$ rather than the full $Q_{\phi_1}(Z_1, Z_2, \dots Z_n|X)$. Replacing the constraint in the inf with the sufficient condition according to Eq. (6), which amounts to replacing Eq. (20) with Eq. (22), we obtain:

$$\text{OT}_c(P_D, P_\Theta) \leq \inf_{\substack{Q_{\phi_1}(Z_1|X), \\ Q_1^{\text{agg}} = P_{\Theta_{2:N}}}} \int_{\mathcal{X}\times\mathcal{X}} c(x, \tilde{x})\, \gamma(x, \tilde{x})\, dx\, d\tilde{x} \tag{23}$$

with $\gamma$ still as in Eq. (18). Eq. (8) is then obtained by relaxing constraint in Eq. (23); replacing the hard constraint by a soft constraint via a penalty term added to the objective, weighted by a $\lambda_1$:

$$\begin{aligned}\widehat{W}_c(P_D, P_\Theta) = \inf_{Q_{\phi_1}(z_1|x)} \Bigg[ \int_{\mathcal{X}\times\mathcal{X}} c(x, \tilde{x})\, \gamma(x, \tilde{x})\, dx\, d\tilde{x} \\ + \lambda_1\, \mathcal{D}_1\Big( Q_1^{\text{agg}}(Z_1),\, P_{\Theta_{2:N}}(Z_1) \Big) \Bigg]\end{aligned} \tag{24}$$

where $\mathcal{D}_1$ is any divergence function between distributions on $\mathcal{Z}_1$.

## B. Experiments

### B.1. MNIST experiments

#### Experimental setup

We train a deep-hierarchical latent-variable model with $N = 5$ latent layers whose dimensions are $d_{\mathcal{Z}_1} = 32$, $d_{\mathcal{Z}_2} =$

16, $d_{\mathcal{Z}_3} = 8$, $d_{\mathcal{Z}_4} = 4$ and $d_{\mathcal{Z}_5} = 2$, respectively. We parametrise the generative and inference models as:

$$\begin{aligned} q_{\phi_i}(z_i|z_{i-1}) &= \mathcal{N}\big(z_i; \mu_i^q(z_{i-1}), \Sigma_i^q(z_{i-1})\big), \quad i = 1, \dots, 5 \\ p_{\theta_i}(z_{i-1}|z_i) &= \mathcal{N}\big(z_{i-1}; \mu_i^p(z_i), \Sigma_i^p(z_i)\big), \quad i = 2, \dots, 5 \\ p_{\theta_1}(x|z_1) &= \delta\big(x - f_{\theta_1}(z_1)\big) \end{aligned} \tag{25}$$

For both the encoder and decoder, the mean and diagonal covariance functions $\mu_i, \Sigma_i$ are fully-connected networks with 2 same-size hidden layers (consider $f_{\theta_1}$ as $\mu_1^p$). For $i = 1, 2, 3, 4, 5$, the number of units is 2048, 1024, 512, 256, 128 respectively.

For the regularisation hyperparameters, we use $\prod_{i=1}^n \lambda_i = \lambda_{\text{rec}}^n / d_{\mathcal{Z}_n}$ for $n = 1, \dots, 4$ (for each reconstruction term in the objective), and $\prod_{i=1}^5 \lambda_i = \lambda_{\text{match}}$ (for the final divergence term). We then perform a grid search over the 25 pairs $(\lambda_{\text{rec}}, \lambda_{\text{match}}) \in \{0.005, 0.01, 0.05, 0.1, 0.5\} \otimes \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and find the best result (smallest Eq. (14)) is obtained with $(\lambda_{\text{rec}}, \lambda_{\text{match}}) = (0.01, 10^{-4})$.

Finally, we choose the squared Euclidean distance as the cost function: $c_n(z_n, \tilde{z}_n) = \|z_n - \tilde{z}_n\|_{L^2}^2$. The expectations in Eq. (14) are computed analytically whenever possible, and with Monte Carlo sampling otherwise. We use batch normalisation (Ioffe & Szegedy, 2015) after each hidden fully-connected layer, followed by a ReLU activation (Glorot et al., 2011). We train the models over 5,000 epochs using Adam optimiser (Kingma & Ba, 2015) with default parameters and batch size of 128.

## Additional results

### B.2. Real word dataset

In the following experiments, the inference networks and generative models at each latent layer are taken to be Gaussian distributions with mean and covariance functions parametrised by 3-layer ResNet (similar to those of Eq. 25). A $M$-layer residual network is composed of $M-1$ convolutional blocks followed by a resampling convolution, and a residual connection. The outputs of the two are added and a last operation (either fully connected or convolution layer) is applied on the result. A convolutional block is composed of a convolution layer followed by batch normalisation (Ioffe & Szegedy, 2015) and a ReLU non-linearity (Glorot et al., 2011). We also use batch normalisation and ReLU after the sum of the convolutional blocks output and the residual connection. See Figure 9 for an example of a 3-layers residual network with a last convolution operation. When resampling, we use a convolution layer with stride 2 in both the skip connection and the resampling covolution for the inference networks and a deconvolution layer with stride 2 in both the skip connection and the resampling convolution in the generative models. If no resampling is performed,

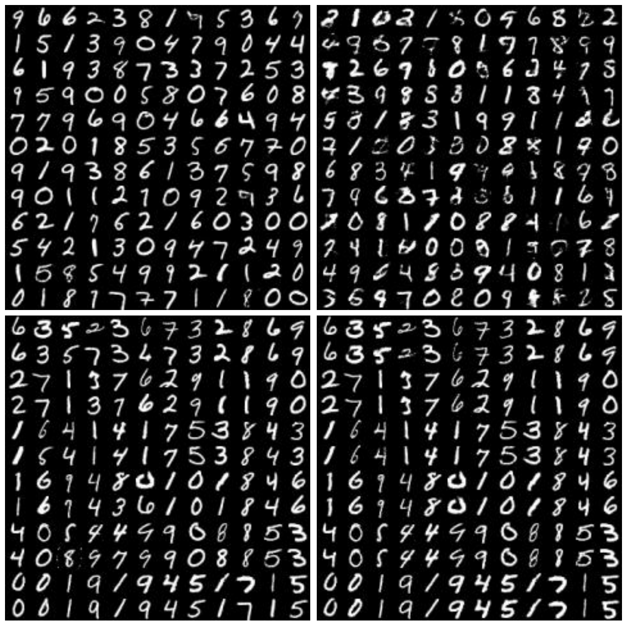
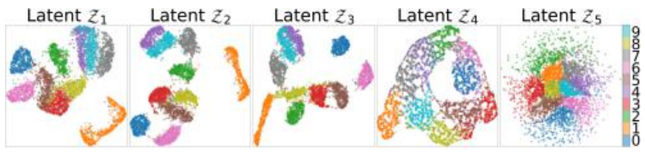
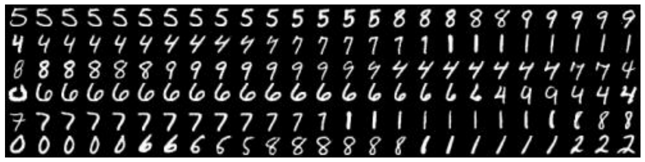


Figure 7: 5-layer STACKEDWAE (left column) *versus* 1-layer implicit-prior WAE (right column). First row: Model samples. Second row: Reconstructions (within pairs of rows, data is above with the corresponding reconstructions below).



(a)



(b)

Figure 8: 5-layer STACKEDWAE. (a) UMAP (McInnes & Healy, 2018) visualisations of latent spaces $\mathcal{Z}_i$. Each colour corresponds to a digit label. $d_{\mathcal{Z}_5} = 2$ can be directly plotted; for higher dimensions we use UMAP. (b) Points interpolations; the first and last columns are actual data points with corresponding reconstructions shown in the second, respectively second-to-last, columns.

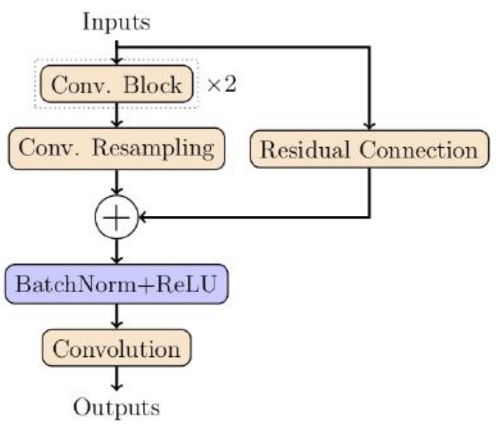


Figure 9: Residual network with 3 hidden convolutions.

then the resampling convolution is a simple convolution layer with stride 1 and the skip connection performs the identity operation. The latent dimensions are then given by the dimensions and the number of features in the last convolutional operation.

The ground cost is chosen to be the $L_2$-squared norm given by $c_n(z_n, \tilde{z}_n) = \|z_n - \tilde{z}_n\|_{L^2}^2$, and the expectations in Eq. (14) are computed analytically whenever possible, otherwise using Monte Carlo sampling.

## SVHN

We train a 6-layers StackedWAE on the SVHN dataset using both the training dataset (73,257 digits) and the additional training dataset (531,131 digits). We train the models over 1000 epochs using Adam optimiser (Kingma & Ba, 2015) with default parameters and batch size of 100.

As previously mentioned, we use 3-layer ResNet for the mean and covariance functions of the inference networks and generative models. More specifically, for each ResNet, we use $M = 2$ convolutional blocks with the dimensions of the filters specified in the top-table of Figure 10. Networks in layers $i = 1, 2$ have 64 convolution filters, layers $i = 3, 4$ 64 and layers $5, 6$ 128, each filters having the same size within each residual network, and doubling (in the inference networks) or divide by 2 (in the generative models) their number in the resampling convolution if any resampling is performed. The latent layers $i = 1, 3, 5$ have a stride of 2 and we choose the number of features to be $2, 1, 2, 1, 2$ and 1 for the latent layers $i = 1 \dots 6$ (top-table of Figure 10 for the full details).

SVHN

| Layer i | Filters dim. | Resampling | Output dim. |
|---|---|---|---|
| Layer 1 | 5×5×64 | down / up | 16×16×2 |
| Layer 2 | 3×3×64 | None | 16×16×1 |
| Layer 3 | 3×3×96 | down / up | 8×8×2 |
| Layer 4 | 3×3×96 | None | 8×8×1 |
| Layer 5 | 3×3×128 | down / up | 4×4×2 |
| Layer 6 | 3×3×128 | None | 4×4×1 |

CelebA

| Layer i | Filters dim. | Resampling | Output dim. |
|---|---|---|---|
| Layer 1 | 7×7×64 | down / up | 32×32×8 |
| Layer 2 | 5×5×64 | None | 32×32×6 |
| Layer 3 | 3×3×64 | None | 32×32×4 |
| Layer 4 | 3×3×64 | down / up | 16×16×8 |
| Layer 5 | 3×3×96 | None | 16×16×6 |
| Layer 6 | 3×3×96 | None | 16×16×4 |
| Layer 7 | 3×3×96 | down / up | 8×8×8 |
| Layer 8 | 3×3×128 | None | 8×8×6 |
| Layer 9 | 3×3×128 | None | 8×8×4 |
| Layer 10 | 3×3×128 | down / up | 4×4×8 |

Figure 10: Details of the architectures used in Section 3.2. Top-table: architecture of the 6-layer StackedWAE trained on SVHN. Bottom-table: architecture of the 10-layer StackedWAE trained on CelebA.

CelebA

We train a 10-layers StackedWAE on the CelebA dataset over $100$ epochs using Adam optimiser (Kingma & Ba, 2015) with default parameters and batch size of $64$.

We use the same ResNet building blocks than previously, mainly, $M = 2$ convolutional block, each filter within a ResNet block having the same size, and doubling their number in the last convolutional operation. Networks in layers $i = 1, 2, 3, 4$ have $64$ filters, $96$ in layers $i = 5, 6, 7$ and $128$ in the remaining layers. The latent layers $i = 1, 4, 7, 10$ have stride $2$ and we choose the number of features to be $8, 4, 2, 8, 4, 2, 8$ (See bottom-table of Figure 10 for the full details).