Chapter 5
Variational Autoencoder

What to expect in the following sections:

- what a generative model is and what its benefits are,
- how to evaluate a generative model,
- detailed explanation of the Variational Autoencoder (VAE),
- developments that enhance the VAE's performance in different aspects,
- central problems of this class of models;
- examples and demonstrations of the discussed VAE models.

After the dense chapter on BNNs, the reader will find this one a bit simpler, as the required tools were actually already introduced. By the end of this chapter, one should

- be able to characterize generative models,
- understand when they are useful,
- know the challenges on assessing their quality,
- possess breadth of knowledge on VAE's core idea,
- comprehend the ideas behind its extensions and what effectively changes.

5.1 Motivations

Generative models are statistical models of data that attempt to capture the entire probability distribution from the observables, that is, to estimate $p(\mathbf{x})$ from $\mathcal{D} = \{\mathcal{X}\}_n$. They are a complete description of the probabilistic model that generates the observed data. In possession of the full model we can extrapolate to unseen examples, generate new samples, infer relations and dependencies, perform prediction, and much more.

Differently from discriminative models that use the target $\mathbf{Y}$ as supervisory signal to estimate the conditional distribution of $\mathbf{Y} \mid \mathbf{X}$, generative models do not

L. P. Cinelli et al., *Variational Methods for Machine Learning with Applications to Deep Networks*, https://doi.org/10.1007/978-3-030-70679-1_5

need to be supervised. Thus, the latter naturally fits unsupervised algorithms, which use unlabeled data, an abundant resource. Nonetheless, generative models can also perform discriminative tasks, such as classification, by modeling the joint distribution $p(\mathbf{y}, \mathbf{x})$. The models of Chap. 4 are examples of discriminative models, as they estimate the conditional $p(\mathbf{y} \mid \mathbf{X})$: given some inputs, they output a distribution of probable values for the unknown quantity. However, we cannot infer information such as the most probable input–output pairs or the expected input values for a given observed output, which would require the joint $p(\mathbf{y}, \mathbf{x})$.

Knowledge of the complete probabilistic model means we can simulate how the world evolves [20, 29], anticipate and plan for the future [14, 18], reason out and make decisions [11, 14], and understand elements and their factors of variation [10, 11], among other high-level abstract tasks. Exciting applications that have direct application in industry are image super-resolution [28], compression [55], denoising [8], and audio synthesis [57]. Other examples outside the multimedia domain and arguably even more compelling are in chemistry, for efficient exploration of new compounds [13], biology, for prediction of the effects of mutations in proteins and RNAs [46], and astronomy, for images of optical telescope [44].

A classical example of a generative model is the Naive Bayes that constructs the joint probability for classification. In this chapter, the focus is on modern methods that use approximate inference. Specifically, we discuss the family of VAE models, theirs issues, and advantages.

5.2 Evaluating Generative Networks

There is no universal metric for measuring the performance of a generative model, thus assessing its quality is not straightforward and often misleading. Models trained by optimizing the same criteria relate well across different properties and can be directly compared, i.e., the training criteria can be used as an objective metric for model selection. However, for different objective functions, model comparison becomes problematic. The higher the dimension of the space, the less correlated the metrics become [53].

There are many different properties we may wish for in a model: high quality samples, diverse samples, compact representations, useful representation, interpretable representations, and so on. Naturally, we cannot have all at the same time, so we must compromise.

A model may have a high log-likelihood but an average matching behavior, causing it to assign probability mass to low-density regions. Although the approximation may be overall good, unusual regions will end up getting too much attention, at least more than it normally should. Consequently, generated samples will be more likely to be very distinct from samples from the true distribution in spite of the approximation's good log-likelihood. Although it may seem as an undesirable property at first, it really depends on the context. For drug discovery, it can be great because it means we are exploring other regions of the space, where we can find new useful compounds. For images, it can be a disaster because the resulting image can be amorphous and weird or represent nothing at all. Still, how many abstract, odd artworks end up being highly valuable and praised?

On the other hand, a model with moment-matching behavior produces realistic samples but has poorer log-likelihood for multi-modal distributions. The approximation fits a single mode, assuming a unimodal model, and misses out on a great deal of information. Hence, the resulting sample diversity will also be low. Images are a good example of highly multi-modal distributions over space: there is dependence between neighboring pixels and groups of pixels (superpixels), and on the scene as a whole, besides a lot of structure built-in from the physical world. A model fitted to a single mode would then struggle to mix different factors of variation in image elements.

Besides log-likelihood and sample quality communicating different things, they do not inform us about sample diversity though they are connected. If samples seem to come from the true distribution and, at the same time, have great diversity, then probably the two densities match well overall, which implies good average log-likelihood for the model. Sample diversity relates to the entropy of the distribution.

Average log-likelihood has become the de facto standard measure of quality for generative models. However, the computation may not be possible, at least in a viable amount of time, depending on the model type. For such models, where a direct measure of the log-likelihood is not possible, it is common to resort to Parzen window estimates [53]. The Parzen window method is a nonparametric approach that consists in drawing samples from the original model and constructing a tractable approximate model, for which we can compute the average log-likelihood. Nevertheless, it has been experimentally demonstrated that evaluation with Parzen window estimates does not correlate well with likelihood nor sample quality, hence being strongly discouraged [53].

Visual sample inspection is the natural metric for assessing sample quality, and it is the go-to metric when dealing with image synthesis, as it allows us to find out in an intuitive manner about what is happening into the model. Still, this approach is only effective for audiovisual samples. Samples represent an interesting diagnostic tool but should not be used as a proxy for other tasks [53].

Perceptual quality metrics do not take into account generalization capacity. A model that simply outputs training examples without ever producing a new one would score high but would be of no use. Trivial algorithms such as nearest neighbors for detecting whether generated samples are similar to training ones are not effective: perceptually similar images can have large distances, e.g., a one-pixel shift of a texture-rich image. Additionally, overfit models do not necessarily reproduce the images from the data set [53].

A different approach for assessing the model's performance is to measure it directly on a surrogate task, what is especially useful if the aim is to learn good feature representations. Although indirect, measuring the effect of the model in the intended downstream application provides exactly what one searches for. For example, one could use a small capacity linear classifier on the learned representations to test if they form well-defined clusters, which would be indicated by a high classification accuracy.

The community has been devoting recent efforts to propose principled scores that embrace these aspects and allow to objectively compare between competing algorithms. For images, the Inception Score [48] and the Frechet Inception Distance [16] are two popular metrics that use pretrained classification models to compare the generated samples with a hold-out test set. The former measures sharpness and diversity, whereas the latter measures similarities in the feature representation space. Both empirically correlate well with the perceived quality of samples. However, the Inception Score is only valid when the classifier used for the evaluation was pretrained on the same data set as the generative model under evaluation [47].

In summary, there is no universal metric with sample quality, classification accuracy, and log-likelihood being largely independent properties in high-dimensional spaces. Thus, proper assessment of the model's performance depends on the application: different applications require different metrics, e.g., sample quality for content generation, downstream task for representation learning, and log-likelihood for compression and density estimation.

5.3 Variational Autoencoders

We begin this section by posing a modeling problem and steadily progress toward constructing the KL. We start from the set of observations $\mathcal{D} = \{\mathcal{X}\}_{n=1}^{N}$ and latent variable models, whose value we already discussed in Sect. 2.1.3. Thus, we have $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z})d\mathbf{z}$, where $\mathbf{Z}$ is the unknown latent variables which we assume to be the hidden causes for the observed $\mathbf{x}$, and $p(\mathbf{x}, \mathbf{z})$ is the generative model of Fig. 5.1a.

Although KLs can in principle work with any kind of data, we use it on images to illustrate our points and keep the discussion attractive. Hence, $\mathbf{x}$ corresponds to an image sample. This type of data has great appeal, is intuitive, and has broad support on modern programming frameworks, i.e., Pytorch [41].

We can rewrite the integral over the joint distribution according to the chain rule for conditional probability, which gives us $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} \,|\, \mathbf{Z})p(\mathbf{z})$, where $p(\mathbf{z})$ is the prior distribution over the latent space and $p(\mathbf{x} \,|\, \mathbf{Z})$ the likelihood function. For all but very simple models, the integral $\int p(\mathbf{x} \,|\, \mathbf{Z})p(\mathbf{z})d\mathbf{z}$ is intractable and cannot be analytically calculated; thus, we estimate it by Monte Carlo (MC) sampling with T samples, such as

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z})d\mathbf{z} = \int p(\mathbf{x} \,|\, \mathbf{Z})p(\mathbf{z})d\mathbf{z} \approx \frac{1}{T}\sum_{i=1}^{T} p(\mathbf{x} \,|\, \mathbf{Z}^{(i)}), \tag{5.1}$$

with $\mathbf{z} \sim p(\mathbf{z})$. However, finding samples of $\mathbf{z}$ for which $p(\mathbf{x} \,|\, \mathbf{Z})$ is large is challenging in high-dimensional latent spaces: we need millions of draws to obtain

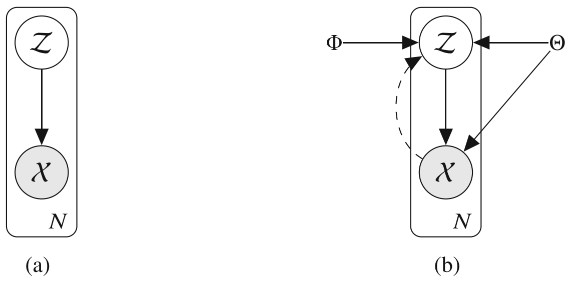


Fig. 5.1 Graphical representations of the generative model $p(\mathbf{x}, \mathbf{z})$. Figure 5.1a is the initial model we formulate: the latent variables $\mathbf{z}_i$ are the hidden cause behind the sample $\mathbf{x}_i$. Figure 5.1b depicts the parameterized model where we assume the parameters responsible for generating the data to be Θ and the posterior approximation to be determined by the parameters Φ. The dashed line represents the inference process from which we determine the posterior approximation $q(\mathbf{z} \mid \mathbf{X}; \Phi)$ from the samples $\mathbf{x}$. (**a**) The latent variable model. (**b**) The parameterized model

reasonable estimates for $p(\mathbf{x})$. Then, how to choose $p(\mathbf{z})$ such that we obtain plausible values of $\mathbf{z}$, for which $p(\mathbf{x} \mid \mathbf{Z})$ is high, with high probability?

Once more, we rewrite the problem as

$$\begin{aligned} p(\mathbf{x}) &= \int p(\mathbf{x} \mid \mathbf{Z}) p(\mathbf{z}) d\mathbf{z} \\ &= \int p(\mathbf{x} \mid \mathbf{Z}) p(\mathbf{z}) \frac{q(\mathbf{z} \mid \mathbf{X})}{q(\mathbf{z} \mid \mathbf{X})} d\mathbf{z} \\ &= \mathbb{E}_q \left[\frac{p(\mathbf{x} \mid \mathbf{Z}) p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{X})} \right] \\ &\approx \frac{1}{T} \sum_{i=1}^{T} \frac{p(\mathbf{x} \mid \mathbf{Z}^{(i)}) p(\mathbf{z}^{(i)})}{q(\mathbf{z}^{(i)} \mid \mathbf{X})}, \end{aligned} \tag{5.2}$$

with $\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{X})$, and approximate the integral with an unbiased MC estimate of T samples.

Under this new perspective, the sampling process occurs according to the proposal distribution $q(\mathbf{z} \mid \mathbf{X})$, and to obtain the same result as before, we need to properly weight the values of $p(\mathbf{x} \mid \mathbf{Z})$ by $p(\mathbf{z})/q(\mathbf{z} \mid \mathbf{X})$. The problem has now become finding suitable candidates for $p(\mathbf{z})$ and $q(\mathbf{z} \mid \mathbf{X})$.

The approach in (5.2) corresponds to Importance Sampling (IS) [35]. This technique is generally applied to reduce the variance of the estimator or when it is difficult to simulate from the original density, the latter being the present case. The optimal proposal distribution $q^*(\mathbf{z} \mid \mathbf{X})$ is

$$q^*(\mathbf{z} \mid \mathbf{X}) = \frac{p(\mathbf{x} \mid \mathbf{Z}) p(\mathbf{z})}{p(\mathbf{x})} = p(\mathbf{z} \mid \mathbf{X}), \tag{5.3}$$

for which we obtain for the single-sample estimator $\hat{p}_{T=1}(\mathbf{x})$ the true distribution we seek, that is,

$$\hat{p}_{T=1}(\mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{Z}^{(1)})p(\mathbf{z}^{(1)})}{q(\mathbf{z}^{(1)} \mid \mathbf{X})} = \frac{p(\mathbf{x} \mid \mathbf{Z}^{(1)})p(\mathbf{z}^{(1)})}{\frac{p(\mathbf{x} \mid \mathbf{Z}^{(1)})p(\mathbf{z}^{(1)})}{p(\mathbf{x})}} = p(\mathbf{x}). \tag{5.4}$$

Yet, the inability to compute $p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{Z})p(\mathbf{z})d\mathbf{z}$ was the very reason that motivated us to search for other solutions. We shall parameterize the distributions as $p(\mathbf{x} \mid \mathbf{Z}; \Theta)$ and $q(\mathbf{z} \mid \mathbf{X}; \Phi)$ and jointly optimize for Θ and Φ. The corresponding Probabilistic Graphical Model (PGM), the concept introduced in Sect. 3.1.1, is represented in Fig. 5.1b. While the posterior distribution $q(\mathbf{z} \mid \mathbf{X}; \Phi)$ allows inferring latent distributions relating to the observables, the likelihood function $p(\mathbf{x} \mid \mathbf{Z}; \Theta)$ enables the generation of new samples when paired with the prior, what effectively means sampling from the joint distribution.

We implement the likelihood model as a neural network (NN) and fit its parameters, Θ, by Maximum Likelihood Estimator (MLE), instead of variational Bayesian inference, though possible. For the variational parameters Φ, we would need to compute local estimates for each sample $\mathbf{x}_i \in \mathcal{D}$ [3]. Besides not scaling well, it implies computing new variational parameters for each test data before estimating the posterior distribution over the latent variables.

Instead, we optimize a separate model, called recognition model, to output the local variational parameters Φ that define the posterior distribution $q(\mathbf{z} \mid \mathbf{X}; \Phi)$. Hence, each new data point $\mathbf{x}'$ goes through a function $f(\mathbf{x}'; \Psi) \mapsto \Phi$. The problem becomes solving for the global variational parameters Ψ that define the mapping $f(\cdot; \Psi)$. We shall also use an NN for the recognition model. The approach of sharing the variational parameters across data points is called *amortized inference* and is common in settings with large data sets because it effectively amortizes the inference cost, allowing for faster training and testing.

We immediately note that the optimal value for Φ is such that $q(\mathbf{z} \mid \mathbf{X}; \Phi) = p(\mathbf{z} \mid \mathbf{X}; \Theta)$ or at least as close to it as possible. We have arrived at a familiar framework in which we wish to maximize the evidence $p(\mathbf{x}; \Theta)$, and for this, we do

$$\max_{\Theta} p(\mathbf{x}; \Theta) = \max_{\Theta} \log p(\mathbf{x}; \Theta) = \max_{\Theta, \Phi} \log \mathbb{E}_q \left[\frac{p(\mathbf{x} \mid \mathbf{Z}; \Theta)p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{X}; \Phi)} \right]. \tag{5.5}$$

Applying Jensen's inequality exactly as we did in Sect. 3.2.1.1 leads us once again to the ELBO objective (3.7, 3.8) as we verify in

$$\log \mathbb{E}_q \left[\frac{p(\mathbf{x} \mid \mathbf{Z}; \Theta)p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{X}; \Phi)} \right] \geqslant \mathbb{E}_q \left[\log \frac{p(\mathbf{x} \mid \mathbf{Z}; \Theta)p(\mathbf{z})}{q(\mathbf{z} \mid \mathbf{X}; \Phi)} \right] = \text{ELBO}(\Theta, \Phi). \tag{5.6}$$

Even though the final utility function has a similar form to those seen in Chap. 4, there are some subtle but important differences. Here, we perform

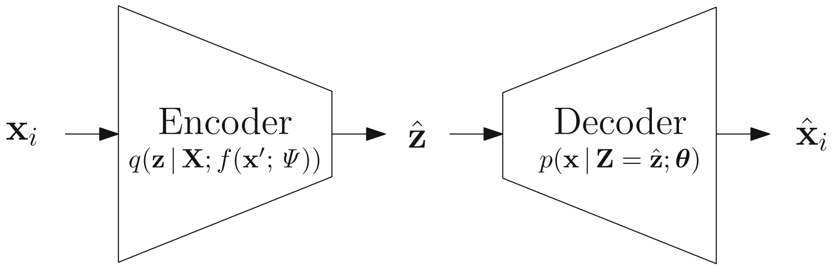

$\mathbf{x}_i$
Encoder
$q(\mathbf{z} \,|\, \mathbf{X}; f(\mathbf{x}'; \Psi))$
$\hat{\mathbf{z}}$
Decoder
$p(\mathbf{x} \,|\, \mathbf{Z} = \hat{\mathbf{z}}; \boldsymbol{\theta})$
$\hat{\mathbf{x}}_i$

Fig. 5.2 Schematic of the KL model. The image $\mathbf{x}_i$ gets mapped, thanks to the encoder NN, to a distribution over the latent variable $\mathbf{Z}$, from which we draw a sample $\hat{\mathbf{z}}$ (one-sample MC estimator). Next, we pass the sample through the likelihood model NN to obtain the distribution $p(\mathbf{x} \,|\, \mathbf{Z} = \hat{\mathbf{z}}; \theta)$, whose most probable values should be the $\mathbf{x}_i$ that generated the latent sample $\hat{\mathbf{z}}$

Variational Inference (VI), on the latent variables $\mathbf{Z}$, but point estimation on the variables Θ of the likelihood model. In Chap. 4, we performed VI on all variables, which were global, since they were the same for all data points, and there were no local latent variables. A brief treatment on the Full Variational Bayes version can be found in [23].

Note that the target density $p(\mathbf{x} \,|\, \mathbf{Z}; \Theta)$ changes over the course of training, not being static as the target densities of Chap. 4. Hence, $q(\mathbf{z} \,|\, \mathbf{X}; f(\mathbf{x}'; \Psi))$ must track this evolution so that the approximation remains "close" to the true (modeled) distribution.

From the complete model developed so far, shown in Fig. 5.2, we observe that the distribution over the latent space $\mathcal{Z}$ is in between the recognition and the likelihood models, creating an information bottleneck if $dim(\mathcal{Z}) < dim(\mathcal{X})$ [1]. Generally, this is the case since we assume that the data lives in a lower dimensional manifold than the space in which it is defined. Therefore, we may interpret the present class of models as encoding $\mathbf{x}$ to a lower dimensional space $\mathcal{Z}$, thus throwing away unnecessary information and preserving what is meaningful, which actually helps the decoder to reconstruct the original input. Hence, $q(\mathbf{z} \,|\, \mathbf{X}; \Psi)$ can be understood as a probabilistic encoder and $p(\mathbf{x} \,|\, \mathbf{Z}; \Theta)$ as a probabilistic decoder. Indeed, if we write the ELBO for the data set $\mathcal{D}$ with N samples in its most usual form, we have

$$\mathrm{ELBO}(\Theta, \Psi) = \sum_{n=1}^{N} \mathbb{E}_q \left[\log p(\mathbf{x}_n \,|\, \mathbf{Z}_n; \Theta)\right] - D_{KL}\left(q(\mathbf{z}_n \,|\, \mathbf{X}_n; \Psi) \| p(\mathbf{z})\right). \tag{5.7}$$

The first term in (5.7) aims at maximizing the likelihood of the reconstructed samples. The second term, on the other hand, works as a regularization factor imposing structure to the latent space. It must be structured so that the conditional on $\mathbf{x}_i$ is as similar as possible to the prior. Deviations from the prior should be meaningful enough, so the decoder can achieve a better reconstruction and pay off the toll imposed by the Kullback-Leibler (KL). Without the KL term, (5.7) would correspond to MLE maximization and the latent distribution would degenerate to a point estimate. This would entail a conventional autoencoder that deterministically maps a data point $\mathbf{x}_i$ to $\mathbf{z}_i$ and deterministically reconstructs it. Consequently, nearby latent points would not necessarily represent similar data points, just as in a lookup table. Thus, the latent space would not have a significant structure. The autoencoder with the KL regularization term in the latent space receives the name of *Variational Autoencoder* [23].

From the information bottleneck perspective [54], we can see the reconstruction error as a measure of distortion and the posterior misalignment as the communication rate between the prior and posterior distributions [1, 2]. Indeed, in information theory, the $D_{KL}\,(q \| p)$ can be interpreted as the extra number of bits required to send a message under q with a coding scheme that was optimally designed for the p. When the KL is zero, we have $q(\mathbf{z} \,|\, \mathbf{D}; \Psi) = p(\mathbf{z})$, and there is no information about the input $\mathbf{x}$ flowing through the model, meaning that the latent channels have zero capacity. A larger overlap between the distributions corresponds to a less informative posterior, with respect to the input x_i, and a higher reconstruction error (distortion). This constraint forces similar data points to have similar posterior distributions, imposing smoothness and locality to the latent space.

The latent distribution in between the encoder and decoder raises difficulty when trying to use gradient descent to optimize the model. We cannot numerically compute the gradient of an expectation w.r.t. its distribution; see Appendix A.1 for an in-depth discussion. However, for continuous latent distributions and differentiable likelihood models, we resort to the pathwise gradient estimator (Appendix A.1), more commonly known as the reparameterization trick [23], which after T MC samples leads to

$$\widehat{\mathrm{ELBO}}_1(\Theta, \Psi) = \sum_{n=1}^{N} \frac{1}{T} \sum_{i=1}^{T} \Big[\log p(\mathbf{x}_n, \mathbf{z}_n^{(i)}; \Theta) - \log q(\mathbf{z}_n^{(i)} \,|\, \mathbf{X}_n; \Psi) \Big], \tag{5.8}$$

where $\mathbf{z}_n^{(i)} = g(\epsilon^{(i)}, \mathbf{x}_n; \Psi)$ is a deterministic transformation and $\epsilon^{(i)}$ the i-th sample from the base distribution $p(\epsilon)$. The above estimator is equivalent to (4.8), put forth in [4] for BNNs.

By choosing families of distributions for $p(\mathbf{z})$ and for $q(\mathbf{z} \,|\, \mathbf{X}; \Psi)$ such that the KL term in (5.7) has closed-form analytical formula, we rewrite the estimator in (5.8) as

$$\widehat{\mathrm{ELBO}}_2(\Theta, \Psi) = \sum_{n=1}^{N} \left[\frac{1}{T} \left[\sum_{i=1}^{T} \log p(\mathbf{x}_n \,|\, \mathbf{Z}_n^{(i)}; \Theta) \right] - D_{KL}\left(q(\mathbf{z}_n \,|\, \mathbf{X}_n; \Psi) \| p(\mathbf{z}) \right) \right], \tag{5.9}$$

which is the form used throughout Chap. 4. Figure 5.3 illustrates the reparameterization trick applied to a random node $\mathbf{z}$ in the computational graph with the KL divergence being analytically calculated.

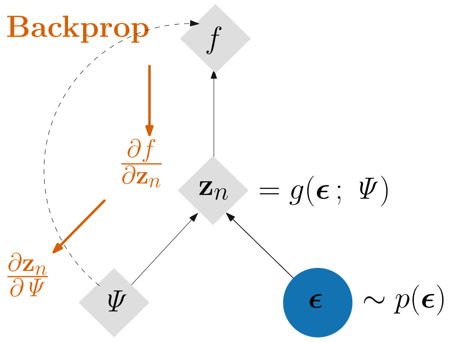

Backprop
f
$\frac{\partial f}{\partial \mathbf{z}_n}$
$\mathbf{z}_n$
$= g(\boldsymbol{\epsilon}\,;\ \Psi)$
$\frac{\partial \mathbf{z}_n}{\partial \Psi}$
Ψ
$\boldsymbol{\epsilon}$
$\sim p(\boldsymbol{\epsilon})$

Fig. 5.3 Computational graph after the reparameterization trick. The blue round node is a random node, while the gray rhombus nodes are deterministic. Black arrows represent the forward pass of the model and the red ones the backpropagation path. The black dashed line indicates the path for the computation of the KL divergence, which takes the distribution parameters Ψ as input. Note that, thanks to the reparameterization trick, the node $\mathbf{z}_n$ is no longer random, and so we can directly compute its gradient

The main technical contribution of [23] was introducing for the first time, in 2013, to the Deep Learning (DL) community the reparameterization trick to obtain a low-variance gradient estimator. The widely used KL model is simply a use case example of this estimator the authors give midway through the paper [23]. The name for the generic formulation of Fig. 5.1a optimized with (5.8) is Autoencoding Variational Bayes (AEVB). Here we consider the most common instantiation of VAE: $q(\mathbf{z} \mid \mathbf{X};\, \Psi)$ and $p(\mathbf{x} \mid \mathbf{Z};\, \Theta)$ both implemented with feedforward NNs. Still, we could implement the same general model with other blocks, such as autoregressive models.

Later, in Sect. 5.6, we will work with binary images. Consequently, we describe the pixels as realizations of independent Bernoulli distributions and interpret each output pixel of the generative NN as an estimate of Bernoulli's parameter p in the original sample.

In what follows, we suppose the input to be real-valued images, using $\mathcal{N}(\mu_i, \sigma_i^2)$ as the likelihood function $p(\mathbf{x}_i \mid \mathbf{Z};\, \Theta)$, and a centered diagonal $\mathcal{N}(\mathbf{0}, \mathbf{I})$ for the a priori $p(\mathbf{z})$. Thus the KL term in (5.7) simplifies to

$$D_{KL}\left(q(\mathbf{z}_n \mid \mathbf{X}_n;\, \Psi) \| p(\mathbf{z})\right) = \sum_{i}^{|\mathbf{z}|} \log \frac{1}{\sigma_i} + \frac{1}{2}\left(\mu_i^2 + \sigma_i^2 - 1\right). \tag{5.10}$$

In addition, approximate the posterior $q(\mathbf{z}_i \mid \mathbf{X}_i;\, \Psi)$ with a Gaussian distribution with diagonal covariance matrix. These choices for the distributions are not at all due to restrictions in the algorithm, but rather motivated by their simplicity. The deterministic transformation $g(\boldsymbol{\epsilon};\, \mathbf{x}_n, \Psi)$ is then

$$g(\boldsymbol{\epsilon};\, \mathbf{x}, \Psi) = \mu(f(\mathbf{x};\, \Psi)) + \sigma(f(\mathbf{x};\, \Psi)) \odot \boldsymbol{\epsilon}, \tag{5.11}$$

with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $f(\cdot; \Psi)$ is the recognition model and $\odot$ the operator for elementwise multiplication.

Although we employ the same transformation all the time, it does not mean that it is the only possible one, and it just happens that location-scale transformations of the standard distribution of a given family are simple and practical. Another viable option, for example, is to specify g as the inverse Cumulative Density Function (CDF) of the desired distribution and $\boldsymbol{\epsilon} \sim \mathcal{U}(\mathbf{0}, \mathbf{1})$. While we can use full covariance Gaussian posterior, the optimization problem becomes considerably harder with $O(K^2)$ parameters instead of $O(K)$, where K is the number of dimensions in the latent space, besides needing to ensure that the covariance matrix is positive semidefinite.

Experimentally, the authors [23] verified that when using mini-batch optimization with size M, one sample from the approximate posterior $\mathbf{z}^{(1)} \sim q(\mathbf{z} \mid \mathbf{X}; \Theta)$ is enough as long as M is large enough, e.g., 100. Nevertheless, it has become common for practitioners to use one sample even when the mini-batch size is not that large because of the computational gains.

We summarize the (vanilla) VAE algorithm at high level with an arbitrary base distribution $p(\epsilon)$ in Algorithm 6.

Algorithm 6: VAE (or more generally, AEVB algorithm)

```
1: while not converged do
2:     Randomly sample a data example $\mathbf{x}_i$
3:     Randomly sample $\boldsymbol{\epsilon}$ from the base distribution $p(\epsilon)$
4:     Compute the gradients of the ELBO estimator w.r.t. $\Theta$ and $\Psi$
5:     Update the parameters $\Theta$ and $\Psi$ using the gradients
6: end while
```

5.3.1 Conditional VAE

The vanilla KL model does not allow us to constrain the generated sample to have a particular characteristic: one should relentlessly draw samples until obtaining the desired feature, which restricts its usefulness in practical applications. For example, an ordinary task would be automatically coloring a person's hair in a photograph prior to actually dyeing it. A question then arises on how to endow the model to create targeted samples rather than completely random ones.

What we really wish is to *condition* the model output on some kind of information $\mathbf{Y}$, hence the name Conditional VAE (CVAE) [50]. The aim becomes to maximize $\log p(\mathbf{x}_i \mid \mathbf{Y}_i)$ for each observed variable $\mathbf{x}_i$. Following the same derivation as in (5.6) and (5.7), we arrive at

$$\begin{aligned}\log p(\mathbf{x}_i \mid \mathbf{Y}_i; \Theta) &\geqslant \mathbb{E}_{q(\mathbf{z} \mid \mathbf{X}_i, \mathbf{Y}_i; \Psi)} \left[p(\mathbf{x}_i, \mathbf{z} \mid \mathbf{Y}_i; \Theta) - q(\mathbf{z} \mid \mathbf{X}_i, \mathbf{Y}_i; \Psi)\right] \\ &= \mathbb{E}_{q(\mathbf{z} \mid \mathbf{X}_i, \mathbf{Y}_i; \Psi)} \left[\log p(\mathbf{x}_i \mid \mathbf{Z}, \mathbf{Y}_i; \Theta)\right] \\ &\quad - D_{KL}\left(q(\mathbf{z} \mid \mathbf{X}_i, \mathbf{Y}_i; \Psi) \| p(\mathbf{z} \mid \mathbf{X}_i, \mathbf{Y}_i; \Theta)\right). \end{aligned} \tag{5.12}$$

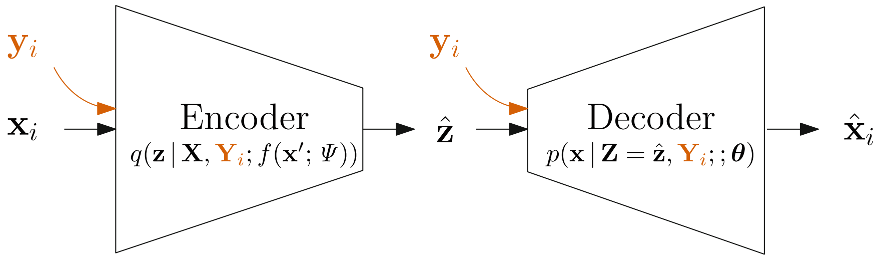

$\mathbf{y}_i$
$\mathbf{x}_i$
Encoder
$q(\mathbf{z} \mid \mathbf{X}, \mathbf{Y}_i; f(\mathbf{x}'; \Psi))$
$\hat{\mathbf{z}}$
$\mathbf{y}_i$
Decoder
$p(\mathbf{x} \mid \mathbf{Z} = \hat{\mathbf{z}}, \mathbf{Y}_i; ; \boldsymbol{\theta})$
$\hat{\mathbf{x}}_i$

Fig. 5.4 Schematic illustration of the CVAE model. Note that this is basically the same as the KL, the sole difference being the inclusion of additional conditioning information **Y** to the input **x** and the sampled latent variable **z**. The former so that the recognition model can infer the distribution corresponding to that condition, and the latter so that the generation network also knows to which condition that distribution refers

Recalling that for the KL, the inference model $q(\mathbf{z}_i \mid \mathbf{X}_i; \Psi)$ has input $\mathbf{x}_i$ and output $\mathbf{z}_i$, and it becomes straightforward that for the conditional counterpart $q(\mathbf{z}_i \mid \mathbf{X}_i, \mathbf{Y}_i; \Psi)$, we must just add $\mathbf{y}_i$ as input. One possible way is to concatenate the condition $\mathbf{y}_i$ at the end of $\mathbf{x}_i$ before passing through the inference model. Similar reasoning applies to the generator model. Thus, by changing nothing more than the input to the models, we obtain a CVAE [50].

From an implementation perspective, we can encode category information as a one-hot representation, indicating to the model which class is at the input (or latent code). Intuitively, the prior gets split into different regions, each corresponding to a specific label, which gives us the ability to choose among them. In addition, by separating the samples into different classes, the data points within the same category become more similar, enhancing the modeling capacity and sample quality of CVAEs (Fig. 5.4).

5.3.2 *β-VAE*

As seen from (5.7), optimizing the ELBO is a compromise between the reconstruction quality and posterior alignment with the prior. Depending on the application, we might want to prioritize either realistic high-quality samples or rich latent representations or even something in between. However, the formulation (5.7) offers no control over the individual objectives. Hence, higher ELBOs do not imply better learned representations. Thus, the ELBO is not a suitable objective function for representation learning [2].

The dilemma of representation size and fidelity is well established in rate-distortion theory, already discussed in Sect. . Ideally, we want the model to find a solution along the Pareto front of the distortion-rate plane, which corresponds to the set of minimal solutions. In the β-KL context, we cast fidelity as the log-likelihood of the model output and connect the concept of rate to the divergence between the posterior and prior distributions because the misalignment creates the need for sending extra bits through the channel to correct the posterior samples inaccurately coded by the prior.

As we have shown, writing the Lagrangian of the equivalent maximization problem leads us to

$$\mathcal{F}(\theta, \phi, \beta) = \mathbb{E}_q \left[\log p(\mathbf{x} \,|\, \mathbf{Z}; \Theta)\right] - \beta D_{KL} \left(q(\mathbf{z} \,|\, \mathbf{X}; \Psi) \| p(\mathbf{z})\right), \tag{5.13}$$

where β is the Lagrangian multiplier.

Equation (5.13) is the objective function of the β-KL algorithm [17], and it is equal to (3.15) after taking the whole data distribution into account. Large values of β, i.e., $\beta > 1$, give higher weight to the posterior misalignment, forcing a lower corresponding δ for the rate R, thus limiting the representation capacity of the latent space. Hence, the data locality property is further encouraged and, consequently, so does the alignment of independent generative factors of variation along separate latent dimensions [7]. The latent representations become better disentangled. The modeled diagonal structure of the covariance matrix of the posterior distribution also contributes to aligning the factors of variation with the axes.

Due to the fidelity and compression trade-off, log-likelihood is not suited for assessing the quality of the representations learned. However, we can use a linear classifier with low capacity to predict labels from the latent space, $p(\mathbf{y} \,|\, \mathbf{Z})$. If the classifier achieves high accuracy values, it means that the latent space is linearly separable and easily decoded, hence disentangled [17].

5.4 Importance Weighted Autoencoder

As we discussed so far, the VAE objective heavily penalizes approximate posterior samples that fail to explain the data. The log-likelihood term in (5.7) must be high enough to be worth the misalignment penalty imposed by the KL regularization. The VAE criterion may be too strict and limit the model flexibility. If we relax this constraint and become more lenient with samples that are unlikely to explain the observations, we obtain more flexibility on the generative model.

We replace the expectation over the likelihood ratio $p(\mathbf{x}, \mathbf{z}; \Theta)/q(\mathbf{z} \,|\, \mathbf{X}; \Psi)$ in (5.5) with the expectation over its estimator. Thus, we have the expectation of the importance sampler of (5.2), like

$$\log p(\mathbf{x}) = \log \mathbb{E}_q \left[\frac{1}{T} \sum_{i=1}^{T} \frac{p(\mathbf{x}, \mathbf{z}; \Theta)}{q(\mathbf{z} \,|\, \mathbf{X}; \Psi)} \right] \geq \mathbb{E}_q \left[\log \frac{1}{T} \sum_{i=1}^{T} \frac{p(\mathbf{x}, \mathbf{z}; \Theta)}{q(\mathbf{z} \,|\, \mathbf{X}; \Psi)} \right] = \mathrm{ELBO}_{\mathrm{IS}}\,, \tag{5.14}$$

where once again we have applied Jensen's inequality, and T is the number of drawn samples.

We immediately note that (5.14) is equal to (5.7) when $T = 1$, so it can be understood as a generalization of the latter under the point of view of importance sampling. By taking multiple samples, we get progressively tighter bounds and lower variance [6]. Actually, both bias and variance of the Importance Weighted Autoencoder (IWAE) estimator are reduced at a rate $O(1/T)$, leading to a consistent but biased estimate of the true likelihood $p(\mathbf{x})$ [39]. Hence, the gradient of $\mathrm{ELBO}_{\mathrm{IS}}$ points toward better directions as T increases.

The update rule for IWAE is the average over the samples with weights proportional to the importance weights $\mathbf{w}_i = p(\mathbf{x}, \mathbf{z}; \Theta)/q(\mathbf{z} \,|\, \mathbf{X}; \Psi)$, as shown by

$$\begin{aligned}\nabla_{\Theta}\mathrm{ELBO}_{\mathrm{IS}} &= \nabla_{\Psi,\Theta}\mathbb{E}_q\left[\log\frac{1}{T}\sum_{i=1}^{T}\frac{p(\mathbf{x},\mathbf{z};\Theta)}{q(\mathbf{z}\,|\,\mathbf{X};\Psi)}\right]\\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_{\Psi,\Theta}\log\frac{1}{T}\sum_{i=1}^{T}\frac{p(\mathbf{x},g(\boldsymbol{\epsilon},\mathbf{x}_n;\Psi);\Theta)}{g(\boldsymbol{\epsilon},\mathbf{x}_n;\Psi)\,|\,\mathbf{X};\Psi)}\right]\\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_{\Psi,\Theta}\log\frac{1}{T}\sum_{i=1}^{T}w_i\right]\\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_{\Psi,\Theta}\sum_{i=1}^{T}\tilde{w}_i\log w_i\right],\end{aligned} \tag{5.15}$$

where we have used the reparameterization trick, $\mathbf{z}_n = g(\boldsymbol{\epsilon}, \mathbf{x}_n; \Psi)$ with $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, to move the gradient inside the expectation, and $\tilde{\mathbf{w}}_i$ are the normalized importance weights $\tilde{\mathbf{w}}_i = \mathbf{w}_i / \sum_{i=1}^{T}\mathbf{w}_i$.

Respective to the log-importance weights in (5.15), the normalized weights $\tilde{\mathbf{w}}_i$ can be seen as their softmax version,

$$l_i = \log\mathbf{w}_i \rightarrow \tilde{\mathbf{w}}_i = \frac{e^{l_i}}{\sum_{j=1}^{T}e^{l_j}} = \mathrm{softmax}(\mathbf{l})_i. \tag{5.16}$$

The importance weights then prioritize the sample with the highest log-likelihood ratio, the one that best explains the data. Since the samples with low likelihood are given less importance, the penalty they impute is attenuated, and the approximate posterior is given more liberty with respect to the VAE constraints, allowing it to spread over the modes of the true posterior and become a better approximation.

However, it is important to keep in mind that the model was optimized to have better performance when drawing multiple samples from the posterior. Thus, one cannot expect IWAEs to have good performance for tasks in which single samples will be used. Training using IWAE will result in models where each individual sample from the model will often have a low (standard) ELBO that is why IWAEs might not be a good fit if we wish to use single samples from the approximate posterior for downstream tasks. Besides, importance weighted estimates have notoriously bad scaling properties to high-dimensional latent spaces [24].

Although ELBO_{IS} gets tighter and the gradient variance smaller with more samples T, the magnitude gets even smaller. More precisely, the signal-to-noise ratio of the inference model actually converges at $O(1/\sqrt{T})$ [42]. So, while $T >> 1$ is great for the generative model, it hampers the training of the inference model. This issue can be somewhat alleviated by averaging over M gradient samples, changing the convergence rate to $O(\sqrt{M}/\sqrt{T})$ [42]. However, there are now two sample sizes to tune: while M regulates the variance of the gradient, T regulates the tightness of the bound. In practice, increasing mini-batch size has a similar effect. One can also use different objective functions for each model, generative and inference [42].

5.5 VAE Issues

VAE and its variants are a great tool for generative modeling, but they are not without shortcomings. In what follows we succinctly present their main known problems so far.

5.5.1 Inexpressive Posterior

The ELBO simultaneously tries to fit the data distribution through optimization of the generative model and to perform amortized inference through the optimization of the inference model. However, due to limited capacity, it is not possible to adequately perform both and failures can emerge, imposing trade-offs onto the ELBO.

The independent Gaussian assumption for the posterior limits the expressiveness of the model. One way we can circumvent this weakness is by using more flexible families for the posterior approximation. However, the proposal distribution must still be at the same time efficient to sample from, compute, and differentiate if we wish to operate with large amounts of data.

5.5.1.1 Full Covariance Gaussian

One straightforward possibility is replacing the isotonic Gaussian with diagonal covariance matrix by one with correlation between the dimensions. The arbitrary multivariate Gaussian with full-rank covariance $\boldsymbol{\Sigma}$ allows per axis scaling and rotation, establishing preferential directions in the latent space.

Directly learning the covariance is troublesome because the number of parameters grows at $O(d^2)$ with the number of dimensions d of the latent space, and we need to assure that $\boldsymbol{\Sigma}$ is semipositive definite. Alternatively, we can again write the latent random variable as the deterministic transformation of a base random variable. We use $\mathbf{z} = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{L}$ a lower triangular matrix with non-zero entries on the diagonal.

5.5.1.2 Auxiliary Latent Variables

We can use auxiliary latent variables $\mathbf{A}$ to augment the inference and generative models [30, 43], as for example,

$$q(\mathbf{z} \,|\, \mathbf{X}) = \int q(\mathbf{a}, \mathbf{z} \,|\, \mathbf{X}) d\mathbf{a} = \int q(\mathbf{a} \,|\, \mathbf{X}) q(\mathbf{z} \,|\, \mathbf{A}, \mathbf{X}) d\mathbf{a}. \tag{5.17}$$

This hierarchical specification allows the latent variables to be correlated through $\mathbf{A}$, defining a general non-Gaussian implicit marginal distribution while maintaining the computational efficiency of fully factorized models.

Similarly, for the generative model, we have

$$p(\mathbf{z}, \mathbf{x}) = \int p(\mathbf{z}, \mathbf{x}, \mathbf{a}) d\mathbf{a} = \int p(\mathbf{a} \,|\, \mathbf{Z}, \mathbf{X}) p(\mathbf{z}, \mathbf{x}) d\mathbf{a}. \tag{5.18}$$

The ELBO objective for the auxiliary VAE follows derivations of (3.7) in Sect. 3.2.1.1 straightforwardly, giving

$$\begin{aligned} \log p(\mathbf{x}) &= \log \int p(\mathbf{a}, \mathbf{z}, \mathbf{x}) d\mathbf{a} d\mathbf{z} \\ &\geq \mathbb{E}_q \left[\log \frac{p(\mathbf{a} \,|\, \mathbf{Z}, \mathbf{X}; \boldsymbol{\theta}) p(\mathbf{x} \,|\, \mathbf{Z}; \boldsymbol{\theta}) p(\mathbf{z})}{q(\mathbf{a} \,|\, \mathbf{X}; \boldsymbol{\psi}) q(\mathbf{z} \,|\, \mathbf{A}, \mathbf{X}; \boldsymbol{\psi})} \right] && (5.19) \\ &= \text{ELBO}_{\text{aux}}. && (5.20) \end{aligned}$$

5.5.1.3 Normalizing Flow

A normalizing flow is a sequence of invertible mappings that transforms an initial base distribution into a complex one [45]. At each step k, it transforms the distribution $q_{K-1}(\mathbf{z}_{k-1})$ into $q_K(\mathbf{z}_k)$ through the change of variable specified by the mapping $\mathbf{z}_k = f_k(\mathbf{z}_{k-1})$, specifically

$$q_K(\mathbf{z}_k) = q_{K-1}(\mathbf{z}_{k-1}) \left| \det \left(\frac{\partial f_k^{-1}}{\partial \mathbf{z}_{k-1}} \right) \right| = q_{K-1}(\mathbf{z}_{k-1}) \left| \det \left(\frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right) \right|^{-1}, \tag{5.21}$$

where $\frac{\partial f}{\partial \mathbf{z}}$ is the Jacobian and det the determinant of the matrix.

If the Jacobian determinant can be computed, it is possible to sample from the estimated density by sampling from the base distribution and applying the chain of mappings $f_1 \circ \cdots \circ f_K$. Moreover, because the mappings are invertible, we can also perform inference by applying the inverse mappings in the reversed order $f_K^{-1} \circ \cdots \circ f_1^{-1}$. The log-likelihood of the final distribution $q_K(\mathbf{z}_k)$ can be written as

$$\begin{aligned}\log q_K(\mathbf{z}_k) &= \log\left[q_0(\mathbf{z}_0)\prod_{k=1}^{K}\left|\det\left(\frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right)\right|^{-1}\right]\\ &= \log q_0(\mathbf{z}_0) - \sum_{k=1}^{K}\log\left|\det\left(\frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right)\right|. \end{aligned} \tag{5.22}$$

We can then use normalizing flows parameterized by the output of the inference network to approximate the posterior distribution $q(\mathbf{z} \,|\, \mathbf{X})$ from a simple base distribution $p(\boldsymbol{\epsilon})$, i.e., standard multivariate Gaussian, and obtain more complex distributions [45]. For example, we can use a single hidden-layer feedforward network with scalar output to define a non-linear transformation [45], like

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + \mathbf{b}), \tag{5.23}$$

where the weight $\mathbf{w}$, bias $\mathbf{b}$, and scale $\mathbf{u}$ are learned parameter vectors and $h(\cdot)$ a smooth elementwise non-linearity.

The above mapping is not very expressive, and we need a large stack of chained transformations to capture high-dimensional dependencies [26]. Nonetheless, we can introduce such dependencies by using autoregressive functions that preserve the Jacobian computation simple enough [9, 26, 40]. Autoregressive flows with affine transformations are very popular, but their expressiveness is limited due to their affine nature.Still, chaining multiple layers of autoregressive transformations leads to complex and multivariate distributions.

5.5.2 The Posterior Collapse

The posterior collapse is the common effect of achieving the undesirable local optimum $q(\mathbf{z} \,|\, \mathbf{X}; \boldsymbol{\psi}) = p(\mathbf{z} \,|\, \mathbf{X}; \boldsymbol{\theta}) = p(\mathbf{z})$ during the VAE optimization. In the state where the variational posterior and true model posterior collapse to the prior, the posterior encodes no information about the input $\mathbf{x}$, and no useful latent representation was learned. The KL divergence is zero and no extra bits are communicated. This corresponds to the $(D, 0)$ of the distortion-rate plane discussed in Sects. 5.3.2 and 5.3.2.

The true model posterior is a moving target for the inference network, so it naturally lags behind, especially at the beginning of training where the variables $\mathbf{X}$ and $\mathbf{Z}$ are nearly independent as a consequence of the random initialization of the models' parameters. When $q(\mathbf{z} \,|\, \mathbf{X}; \boldsymbol{\psi})$ and $p(\mathbf{z} \,|\, \mathbf{X}; \boldsymbol{\theta})$ start to diverge, but $\mathbf{Z}$ still is independent of $\mathbf{X}$ such that $q(\mathbf{z} \,|\, \mathbf{X}; \boldsymbol{\psi}) \approx q(\mathbf{z}; \boldsymbol{\psi})$, and no information is passed through the model, the regularization signal from the KL divergence term in the ELBO objective may be too strong compared to the weak signal coming from the data likelihood [15]. As a result, the model is encouraged to ignore the latent encoding and converges to the local optimum $q(\mathbf{z} \,|\, \mathbf{X}; \boldsymbol{\psi}) = p(\mathbf{z} \,|\, \mathbf{X}; \boldsymbol{\theta}) = p(\mathbf{z})$. Then, it is the generative model that is responsible for reconstructing $\mathbf{X}$ and effectively maximizing the bound.

The information about the data distribution can be encoded both on the latent space representation and on the weights of the generative model. An excessively powerful generator facilitates the posterior collapse because it has enough capacity to store sufficient information in its weights. Indeed, it is frequently reported in the literature that autoregressive generators are prone to this effect [2, 15].

Many modifications have been proposed to solve posterior collapse since it was first detected, such as using an annealing schedule on weighting factor on the KL term [51]; modifying the ELBO objective to ensure that the KL has on average a minimum value, guaranteeing a minimum amount of information transmission [26]; using the β-VAE framework with $\beta < 1$ [2]; and aggressively updating the inference network by optimizing it in an inner loop [15]. The latter is currently the most promising one, obtaining better performance for both inference and generative models, but at a higher computational cost, i.e., 2–3 times slower [15].

5.5.3 Latent Distributions

VAEs rely on the pathwise gradient estimator to enable the computation of gradients through random nodes in computational graphs and the usage of automatic differentiation tools. However, this estimator assumes the distribution under expectation to be continuous and cannot be used for the discrete case. This characteristic restraints our modeling capacity, and we cannot, for example, use categorical distributions for the latent space.

There are works that use the score function estimator instead to estimate the gradient, such as [36, 37]. The alternative estimator on which these algorithms rely makes no assumption about the underlying distribution and can work with both discrete and continuous data, eliminating the issue whatsoever. Although more general, as shown in Appendix A.1, the score function estimator has large variance, so it requires usage of variance reduction techniques to work properly.

There are works on how to reparameterize discrete random variables by relaxing them into continuous distributions [19, 33] and use the pathwise estimator to obtain low-variance biased gradient estimates of the objective function. Still, the cost of these methods is the introduction of a new temperature parameter that should be annealed during training.

5.5.3.1 Continuous Relaxation

Similarly to the reparameterization trick where we sampled from an arbitrary Gaussian distribution through transformation of a standard Gaussian random variable, we can use the Gumbel-max trick to allow us to sample a discrete random variable X from the unnormalized K-Categorical distribution $\widetilde{\Pi_K}$, whose PDF is $\pi_K(x)$, using a continuous distribution [32], following

$$x = \underset{k}{\operatorname{argmax}} \log \pi_k(\cdot) + G_k, \tag{5.24}$$

where G_k is one element from the K-sample independent and identically distributed (iid) sequence of standard Gumbel distributed random variables.

The Gumbel is useful to model the distribution of extreme values of samples from the exponential family and its CDF defined by $F(x) = \exp(-\exp(-x))$. Thus, we can use the inverse formula to obtain

$$G = -\log(-\log(U)), \text{ with } U \sim \mathcal{U}[0, 1], \tag{5.25}$$

and combine it with (5.24) to efficiently get discrete samples by drawing from a standard uniform distribution [32].

However, the argmax function is not differentiable and cannot be used within a gradient learning setting. Thus, we replace it with the softmax function to obtain a continuous relaxation with a temperature parameter τ over the probability simplex [19, 33], as follows:

$$x = \operatorname{softmax}(\log \alpha + G). \tag{5.26}$$

The softened version of the reparameterization is known as the Gumbel-softmax trick and the resulting distribution as Concrete. The temperature τ regulates the discreteness of the representation such that $\lim_{\tau \to 0} \text{Concrete}_K(x) = \Pi_K(x)$. Higher temperatures result in a smoother distribution but lower variance of the gradients, while lower temperatures give more accurate samples but higher variance. The additional hyper-parameter τ is robust and generally follows an annealing schedule from high to low temperature during the optimization [19].

5.5.3.2 Vector Quantization

A popular method for learning discrete latent representations is the Vector Quantized VAE, where vector quantization maps the output of the recognition model to the nearest of M reference elements in the codebook, which is then passed to the generative model [58]. Since the nearest-neighbor match is not differentiable, for the algorithm to work the gradients must be copied from the generator input to the decoder output and the codebook updated using nearest-neighbor lookup to match.

Although the algorithm achieves great generation results, the original and prevalent formulation is not probabilistic: all operations are deterministic [58]. Nonetheless, we can replace the nearest-neighbor lookup by sampling over a K-Categorical distribution Π_K, defined as

$$\Pi_K = \prod_{i=1}^{k} p_i^{[x=i]}, \tag{5.27}$$

where $[x = i]$ evaluates to 1 if $x = i$, 0 otherwise.

The probabilities p_i should be the distance between the recognition model output $h(\mathcal{X})$ and the codebook elements $\{\mathbf{c}\}_M$ [52], such that

$$q(\mathbf{z} \,|\, \mathbf{X}) = \Pi_K \left(\mathbf{z} \,|\, \text{softmax}\left(\|\{\mathbf{c}\}_M - h(\mathcal{X})\|_2\right)\right) \tag{5.28}$$

5.6 Experiments

We train and analyze the VAE and CVAE methods with different latent dimensions on two well-known image toy data sets. In addition, we study the effect on the distortion-rate plane of varying the weight β of the KL term in (5.13) as well as the effect of normalizing flows on the posterior distribution.

5.6.1 Data Sets

5.6.1.1 MNIST

The MNIST data set is composed of 60,000 training and 10,000 testing 28×28 grayscale images of handwritten digits [27]. Each sample depicts a single digit out the 10 possibilities. Figure 5.5 presents one example of each digit class.

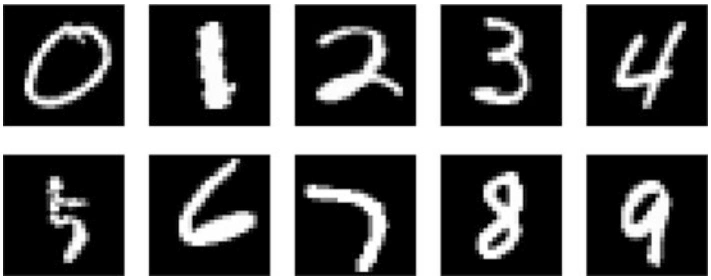


Fig. 5.5 Mosaic of the 10 different digit classes of the MNIST data set

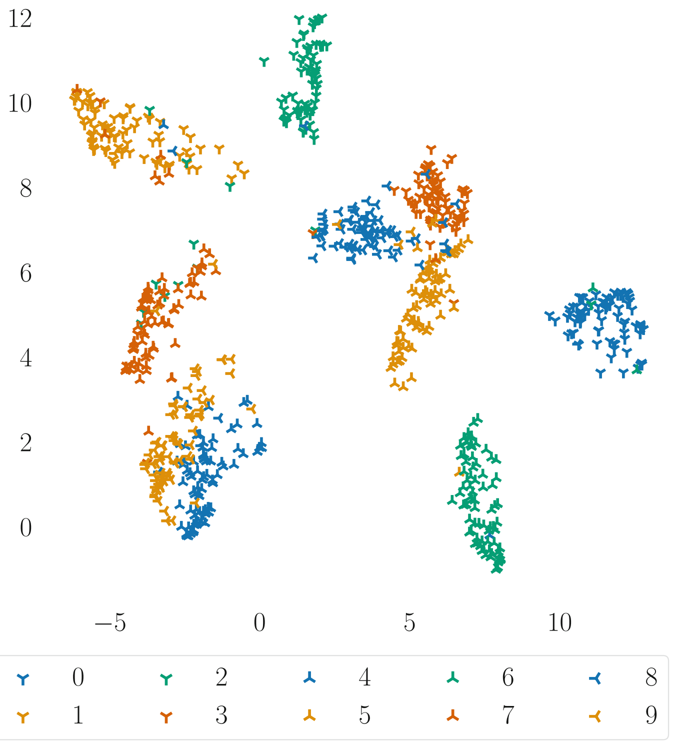


Fig. 5.6 UMAP 2D projection of the raw pixel space of MNIST

Uniform Manifold Approximation and Projection (UMAP) [34] can be used as an out-of-the-box visualization tool similar to t-distributed Stochastic Neighbor Embedding (t-SNE) [31], while being faster, better scaling to high dimensions and better preserving aspects of global data structure. Using this dimension reduction technique, we observe in Fig. 5.6 the structure of MNIST data set. It has well-defined clusters for all of its classes.

Standard ML algorithms obtain over 97% classification accuracy on MNIST, i.e., random forest [5] and support vector machine with Gaussian kernel [49], while DL models over 99.5% [59]. There is almost no space left for researchers to evaluate if observed performance improvements are statistically relevant. Hence, MNIST has little use for benchmarking and is no longer representative of modern computer vision tasks. Still, it has been employed in recent years mainly as a toy data set to do sanity checks and algorithm prototyping.

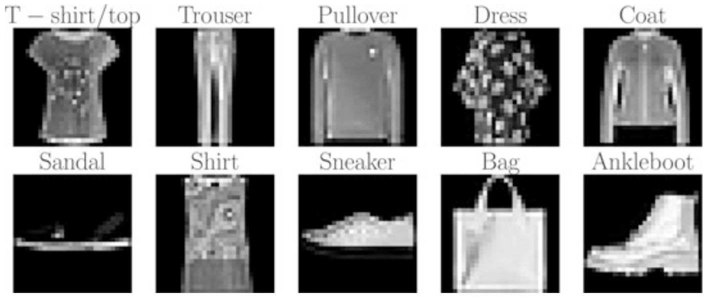


Fig. 5.7 Mosaic of the 10 different classes of the Fashion-MNIST data set

5.6.1.2 Fashion-MNIST

Fashion-MNIST presents the same general layout: 60,000 training and 10,000 test samples, 28×28 grayscale images, and 10 possible exclusive classes [60]. It is constructed to be a drop-in replacement for MNIST with each class associated with a different piece of clothing, as shown in Fig. 5.7. It is noticeable the higher level of details in the images.

Comparing the raw pixel structure of Fashion-MNIST data set, shown in Fig. 5.8, with that from MNIST in Fig. 5.6, we note that the former has clusters corresponding to garments for the same body region partially overlapping. Although simple, Fashion-MNIST is not as easy as MNIST and still has margin for improvements [60].

5.6.2 Experimental Setup

We implement the encoder and generator of all models as fully connected networks with ReLU activations and Gaussian distributions for the latent spaces. In all experiments, we binarize the input images $\mathbf{x}_i$ in the range [0, 1] to {0, 1} with a threshold of 0.5, considering each element j as a realization of a independent Bernoulli distribution. Similarly, we model the output as the parameters of independent Bernoulli distributions $\hat{X}_{ij}$. Hence, the log-likelihood function $\log p(\mathbf{x}_i \mid \mathbf{Z}_i; \Theta)$ becomes the binary cross entropy, like

$$\mathcal{H}[p_i, \hat{p}_i] = \sum_{j \in |X_i|} -x_{ij} \log x_{ij} - (1 - \hat{x}_{ij}) \log(1 - \hat{x}_{ij}), \tag{5.29}$$

where p_i is the binomial distribution induced by the binarized sample $\mathbf{x}_i$, and the $\hat{\cdot}$ symbol denotes quantities related to the model output.

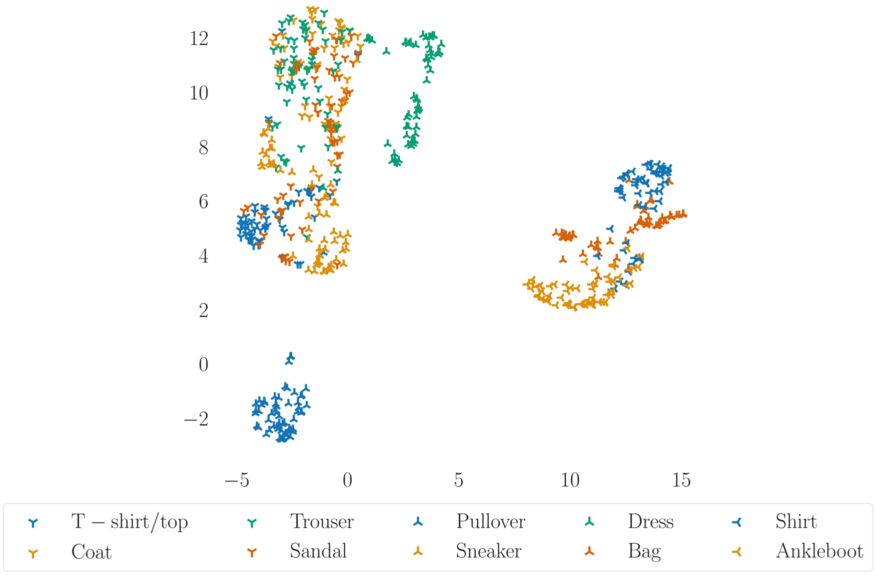


Fig. 5.8 UMAP 2D projection of the raw pixel space of Fashion-MNIST

Additionally, we use mini-batches of size $bs = 128$, draw $T = 1$ MC sample of the latent space for each input example, and train for $epc = 100$ epochs with Adam [21] using a learning rate of $lr = 0.001$.

We train both the VAE and CVAE with varying latent space sizes $d = \{2, 8, 32, 128\}$. Since the MNIST data set is simpler, we use $hl = 1$ hidden layer for both the encoder and the generator, whereas for the Fashion-MNIST we use $hl = 2$. For simplicity, we design the decoders as mirrored versions of the encoders. Thus, the constructed models have the structure:

- MNIST: $784 \rightarrow 200 \rightarrow d \rightarrow 200 \rightarrow 784$;
- Fashion-MNIST: $784 \rightarrow 400 \rightarrow 200 \rightarrow d \rightarrow 200 \rightarrow 400 \rightarrow 784$.

5.6.3 Results

Overall, the behavior of VAE and CVAE models is similar across both data sets as Fig. 5.9 shows. When there are few latent dimensions, the models lose a lot of information in compression and cannot properly represent the different features of the data set, what explains the low ELBO curves for $d = 2$ in Fig. 5.9. Increasing the dimension d of the latent space allows the models to encode relevant information that was previously ignored, bringing expressive gains to the ELBO. Although this boost on performance saturates with larger d, it has almost no negative effect on performance for excessively large d, i.e., the ELBO for the 128-dimensional

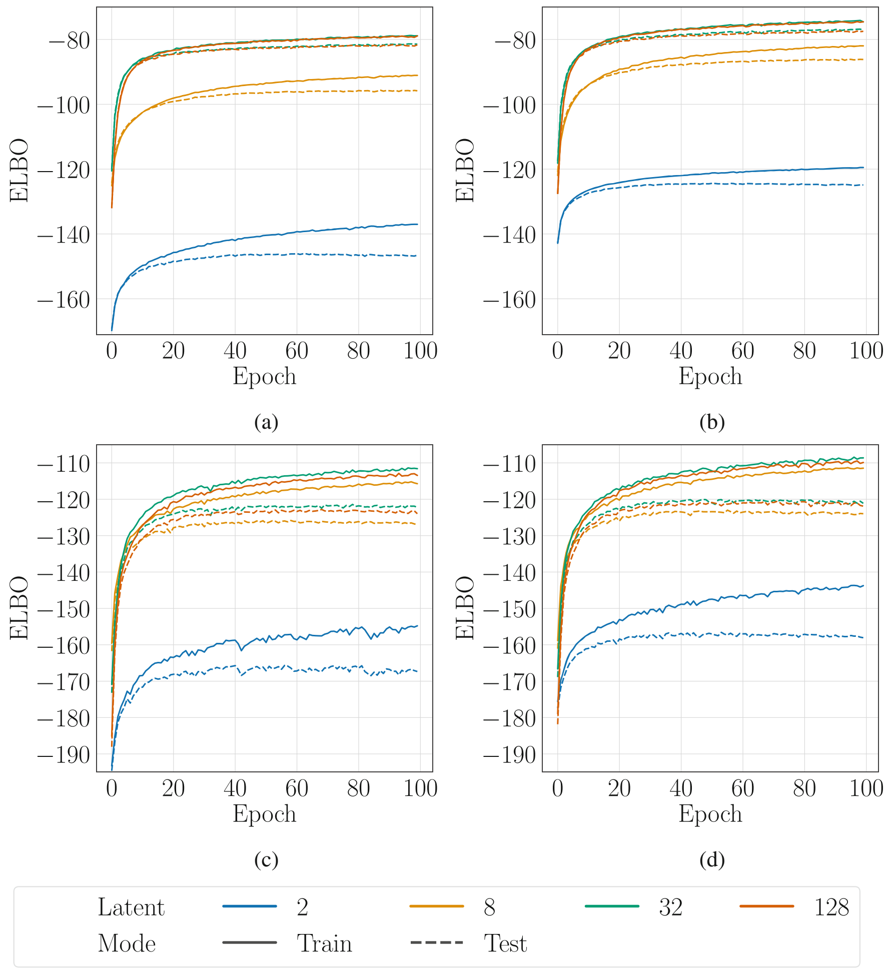


Fig. 5.9 Training and evaluation ELBO for the MNIST and Fashion-MNIST data sets for different sizes of the latent dimension space. Performance on Fashion-MNIST is lower than on MNIST, and the CVAE models achieve performances similar to the VAEs. (**a**) VAE model on MNIST. (**b**) CVAE model on MNIST. (**c**) VAE model on Fashion-MNIST. (**d**) CVAE model on Fashion-MNIST

models is similar to that for the 32D ones in Figs. 5.9a,b, and only slightly lower in Figs. 5.9c,d. VAEs are robust to overfitting, at least with respect to the size of the latent space.

As expected, the Fashion-MNIST models have remarkably worse results. Indeed, we can confirm it visually by observing the generated samples from Fig. 5.10. While the original MNIST samples are not very rich in details, this is not true for the Fashion-MNIST objects, and the VAE struggles to recover the finer details and more complex shapes. The main reason why the trained CVAEs do not increase

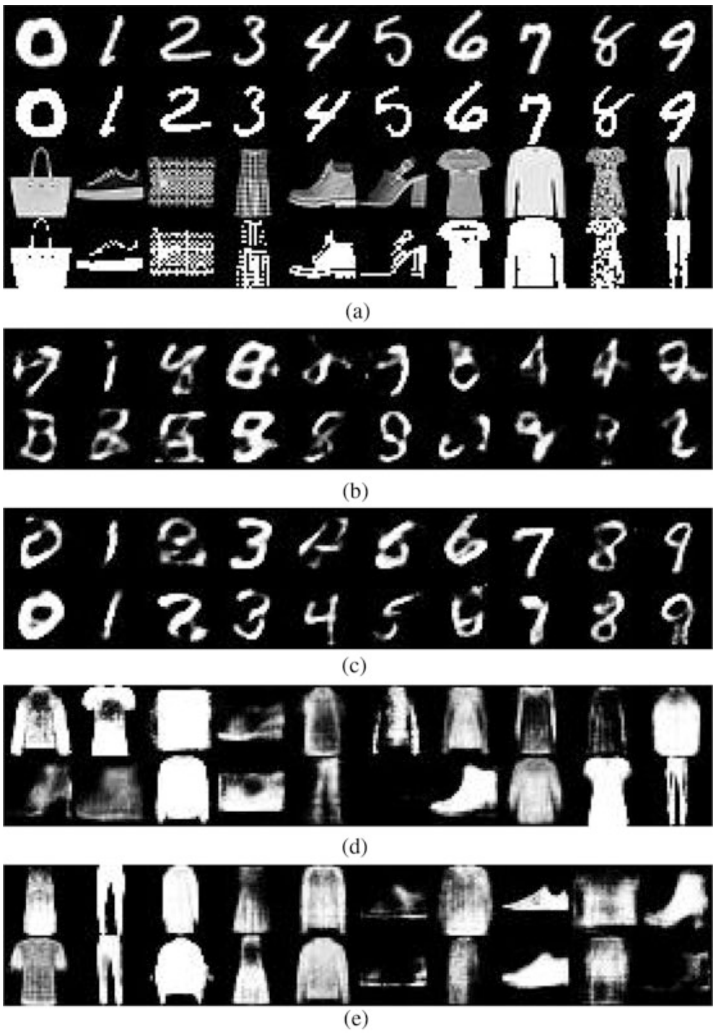


Fig. 5.10 Samples generated by the VAE and CVAE models with $d = 32$. (**a**) Original and corresponding binarized samples from MNIST and FashionMNIST data sets. (**b**) VAE model—MNIST. (**c**) CVAE model—MNIST. (**d**) VAE model—Fashion-MNIST. (**e**) CVAE model—Fashion-MNIST

performance in our experiments is because, in most cases, i.e., $d = \{8, 32, 128\}$, further augmenting the latent dimension does not translate on better ELBOs. Thus, appending 10 extra dimensions for the conditioning does not make the models any more expressible.

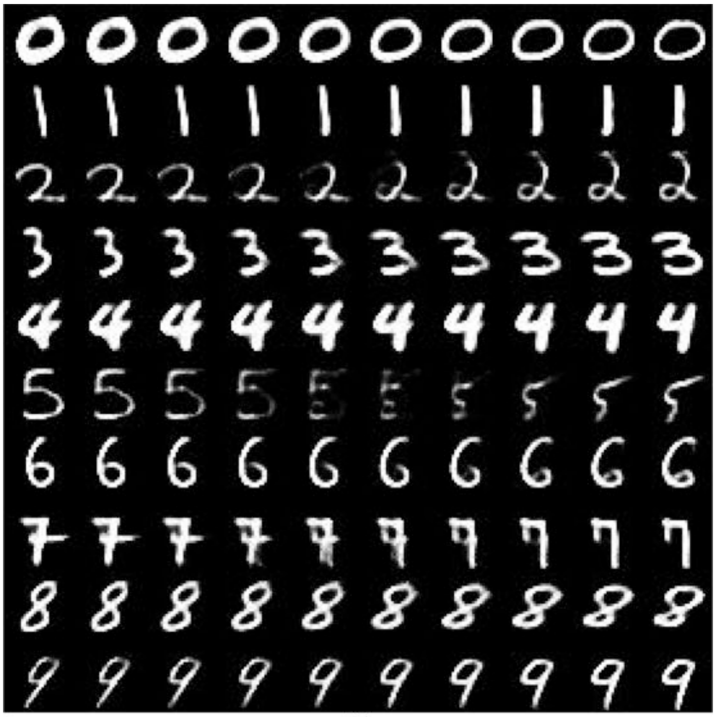
Even though the CVAE does a slightly better job at generating new MNIST images, what we can observe by comparing Figs. 5.10b,c, for Fashion-MNIST data the sample quality is indistinguishable for both CVAE and VAE, because images in Fig. 5.10e are only marginally better than those of the VAE model, displayed in Fig. 5.10d. The quality of both models is overall poor if compared to the real samples of Fig. 5.7.

One central argument during the construction of the VAE was the latent space structure. This property allows us to smoothly interpolate between different latent representations of arbitrary dimension to create new image samples. In Fig. 5.11a, we interpolate between pairs of randomly drawn high-dimensional latent space samples conditioned on the same MNIST digits. We can see the samples gradually morphing, i.e., the 0 gets thinner and the 1 gets simultaneously bolder and straighter. When $d = 2$, we can span the whole latent space and plot its reconstructions. Figure 5.11b shows samples generated from evenly spaced percentiles of the latent Gaussian prior, note the smoothness in the transition between concepts. The models effectively encode factors of variation of the data into the latent space.

We note from Fig. 5.12 that the latent representations of samples from digits 4 and 9, as well as 5 and 3, are generally overlapped, which indicates that the model cannot properly tell them apart. A classifier built from the latent feature space would have a poor accuracy for samples from these classes. Similarly, as already observed in Fig. 5.8, the pullover, coat, and shirt classes are mostly distributed on the same region of the latent space, which intuitively makes sense since they are designed for the same body part and, thus, have similar shapes. From this, we can conclude that the inference network was not capable of identifying the distinctive features of those classes, in accordance with the previous discussion of our models not being powerful enough. More modern types of flow, such as Sylvester flows [56], a generalization of the planar flow, use more powerful transformations and is better suited to real applications.

In Sect. 5.5.1, we discussed different approaches for obtaining more expressive posteriors. Normalizing flows are one of the most prominent approaches nowadays, with several works relying exclusively on it to efficiently perform density estimation and sample generation [22, 26, 40]. In our experiments we use it to enhance VI and obtain models with better posterior approximations and, consequently, higher likelihoods. Table 5.1 shows the estimated marginal log-likelihood on both MNIST and Fashion-MNIST of the VAE with increasing number of steps K in the planar flow, whose transformation was defined in (5.23). Although not statistically significant for $K = \{2, 4, 8\}$, the gain in performance is clear when comparing with the plain VAE. Planar transformations are an elementary case and affect only a small volume of the space at each step, thus calling for a large number of steps to effectively obtain the desired effect, especially on high-dimensional spaces.

Unfortunately, all reconstructed and generated samples were considerably blurry. Blurriness is a general characteristic of VAEs, originating from the objective that seeks to minimize the *average* log-likelihood of data. On average, they may be good, but individually they are not sharp. This effect is more pronounced on Fashion-MNIST as clothes are more diverse and have more details than digits and precisely what we observe in Figs. 5.10 and 5.13. Still, much of the fine-grained details are lost in the binarization procedure applied on the input samples. We could use


(a)

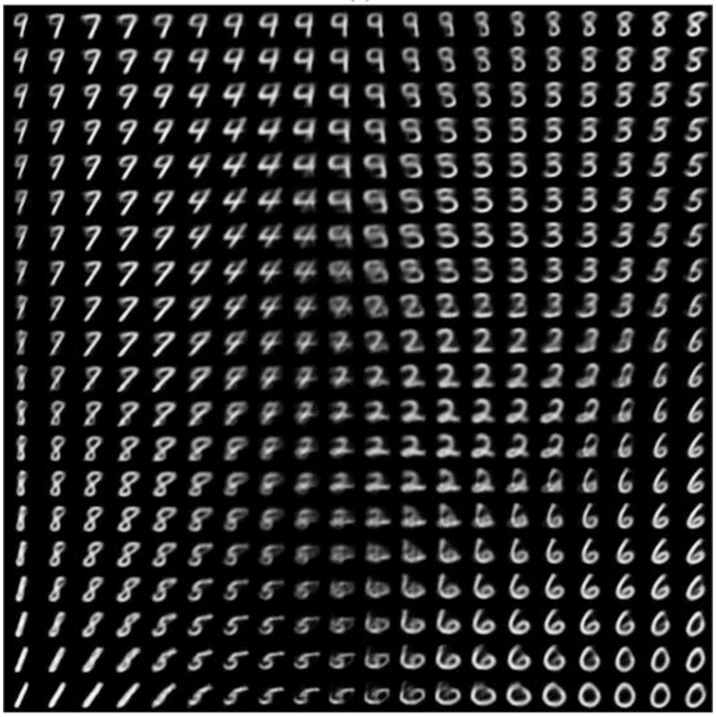

(b)

Fig. 5.11 Interpolation of the latent space of VAE and CVAE models trained on MNIST. Human concepts as thickness, orientation, and digit-specific traits vary smoothly between samples, signaling the latent space effectively captures factors of variation in the data. (**a**) CVAE model with $d = 32$ on MNIST. (**b**) VAE model with $d = 2$ on MNIST

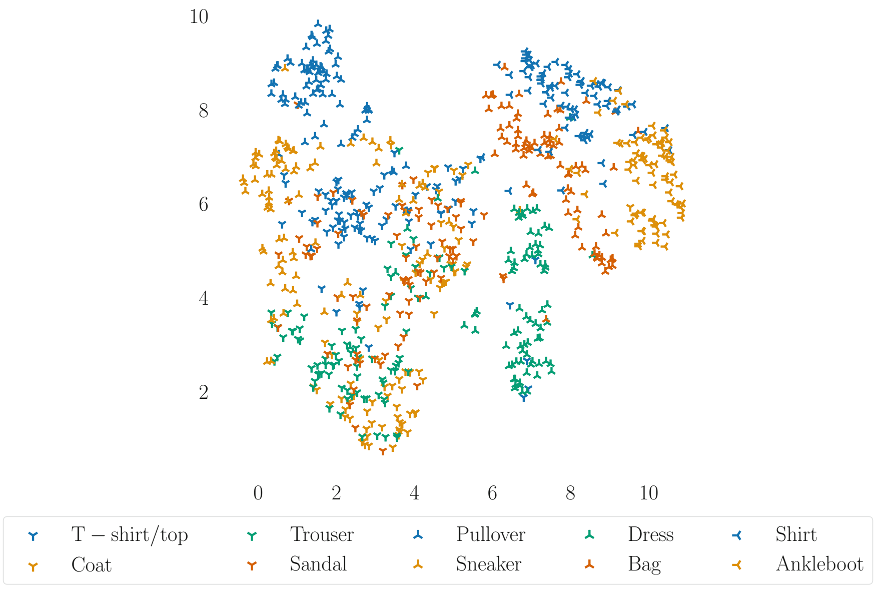


Fig. 5.12 Visualization of the 2D projection of the VAE with $d = 32$ trained on Fashion-MNIST

Table 5.1 Estimated marginal log-likelihood $p(\mathcal{X})$ of the VAE model with planar normalizing flows for varying length K. Estimations are computed by importance sampling with 1024 samples for each instance of the test set

Steps K	Marginal log-likelihood $p(\mathcal{X})$	
	MNIST	FashionMNIST
0	-73.4 ± 0.2	-116.6 ± 0.7
1	-71.4 ± 0.1	-112.3 ± 0.6
2	-71.7 ± 0.6	-112.6 ± 0.7
4	-71.8 ± 0.2	-112.5 ± 0.7
8	-71.8 ± 0.3	-112.5 ± 0.5
16	-71.0 ± 0.3	-112.9 ± 0.6
32	-70.0 ± 0.2	-113.1 ± 0.7

the original grayscale values, but the binary cross entropy that stems from the Bernoulli log-likelihood would cease to be adequate, and it would be necessary to employ one from a suitable continuous real-valued distribution, such as the logit-Normal. In general, to achieve better log-likelihood and sample quality, we need to employ better models. Indeed, plain fully connected networks have pretty much been replaced nowadays by convolutional architectures, especially in the image domain.

The KL divergence remains stable throughout the whole training, varying very little, as shown in Fig. 5.14. Although we only exhibit the case of the CVAE trained on Fashion-MNIST, this is a general behavior observed in all experiments. This is a consequence of the powerful regularizing effect the KL term has on the model, seen in Sect. 5.5.2. The learned posterior distribution moves away from the prior within the first epoch and, even though not much has been learned yet as confirmed by

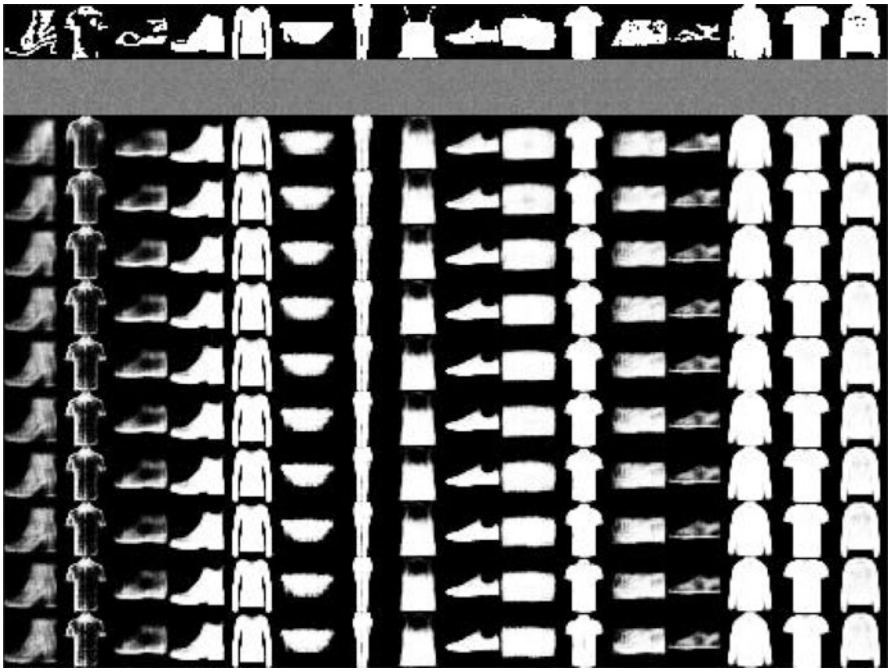


Fig. 5.13 Reconstruction of Fashion-MNIST samples throughout training of the VAE model with $d = 32$ latent dimensions. The original samples are in the first row, while the others are snapshots at every 10 epochs. Most of the information is learned within the first 10 epochs. Still, it is perceptible the higher level of details in the images in the last row

Fig. 5.13, the posterior stays at approximately the same "distance" during the rest of the optimization procedure. Had we used a powerful generator with greater ability to reconstruct the input, e.g., autoregressive model, the KL term strength would have succeeded in keeping the posterior aligned with the prior, causing the undesired posterior collapse. The most straightforward way to sidestep this issue is by directly decreasing the value of β in the β-VAE model. Adjusting the value of β allows us to weight the relative importance of the regularization effect of the KL term on the ELBO. The hyper-parameter balances the compromise between distortion, the reconstruction error, communication rate, and the posterior misalignment, given the model's limited capacity. This characteristic can be observed from Fig. 5.15, where model performance is plotted on the distortion-rate plane.

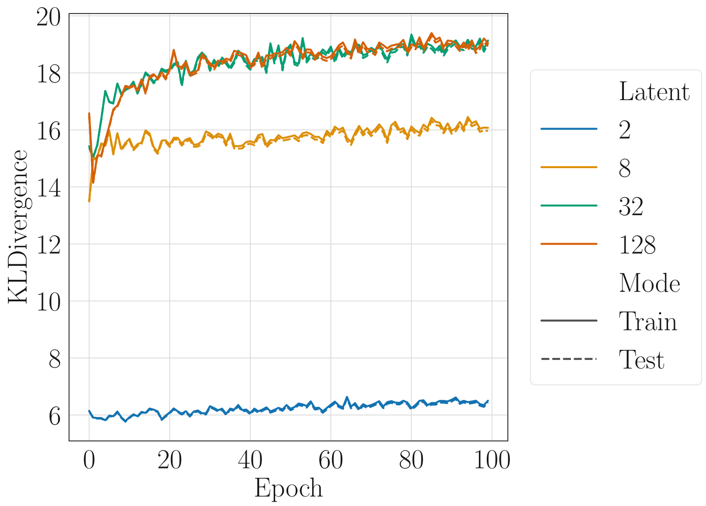


Fig. 5.14 The KL divergence curve during the training and evaluation of the CVAE models with different latent dimension sizes in the Fashion-MNIST data set

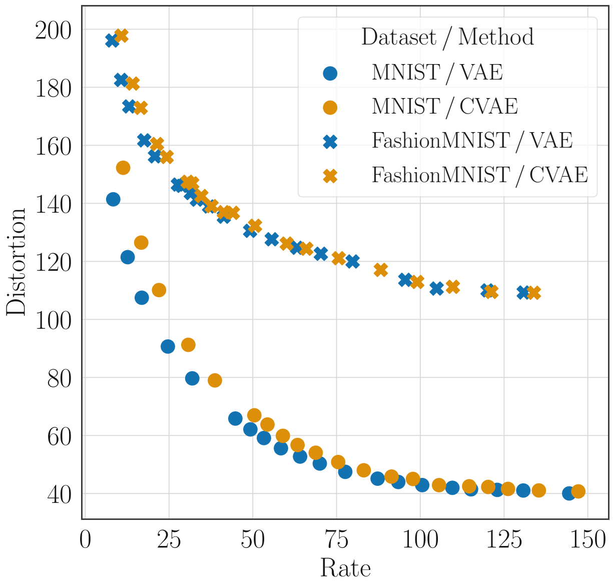


Fig. 5.15 Representation of the ELBO on the distortion-rate plane (in nats). The ELBO can be decomposed into the fidelity term, measured by the log-likelihood of the data set and a rate term, quantifying the average number of extra bits needed to correct the samples inaccurately represented with the prior distribution. In the plane graphic we use negative fidelity, the distortion. Given a model with finite capacity, not capable of achieving the data entropy lower bound, we must set the hyper-parameter β and make other design decisions with this behavior trade-off in mind

5.7 Application: Generative Models on Semi-supervised Learning

Popular modern approaches to ML rely on models with millions of parameters and require large amounts of annotated data, currently the most expensive and desirable asset in AI. When annotation is not extensively available, supervised models overfit to the presented data and achieve poor generalization. Generative models allow us to leverage performance from unlabeled samples, effectively reducing the reliance on annotations, under an approach we call semi-supervised learning.

We can optimize a discriminative classifier together with a VAE, sharing their parameters, and use them for semi-supervised learning of the target variable **Y** [25, 30]. For the unlabeled samples, we treat **Y** as a discrete latent stochastic variable distributed according to a categorical distribution, what enables us to infer the target label **Y**. Figure 5.16 shows the PGMs for the generative and the inference models. Note that we use **Y** to condition the latent variable **Y**, segmenting the latent space in different regions according to the class, similarly to the CVAE in Sect. 5.3.1. Figure 5.17 illustrates the high-level computational graph of such model.

More complex versions of the model in Fig. 5.17 were able to achieve an average classification error below 1% on MNIST while using only 10 labeled images per class, a total of 100 out of the 60,000 available in the training set [30]. The auxiliary deep generative model extends the above VAE with an auxiliary latent variable (see Sect. 5.5.1.2), making it a two-layered stochastic model [30]. This increases the flexibility of the variational approximation, allowing it to better fit complex latent distributions, hence improving the variational lower bound. The underlying PGM for the generative model is shown in Fig. 5.18a, whereas Fig. 5.18b depicts the inference model. They are basically the same as those of Fig. 5.18a, except for the inclusion of the auxiliary node **a**. Figure 5.19 illustrates the high-level computational graph of the auxiliary deep generative model used for

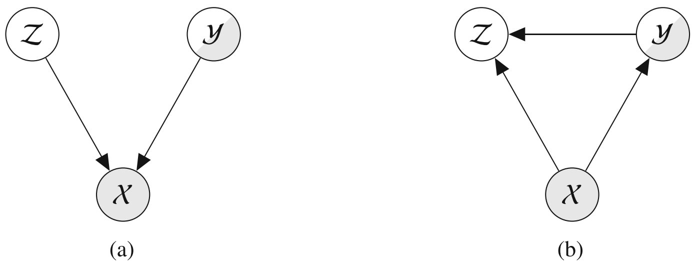


Fig. 5.16 Graphical model of the semi-supervised VAE. The partially colored node **y** denotes the partially observed target labels. We assume that **y** and **z** are conditionally independent in the generative process, so while **y** captures digits' semantics, **z** captures styles and position. Since **z** is never observed and different digits possess different styles, **y** is used during inference to estimate **z**, such relation is depicted in (**b**) by the arrow **y** $\rightarrow$ **z**. (**a**) Graphical model for the generative network. (**b**) Graphical model for the inference network

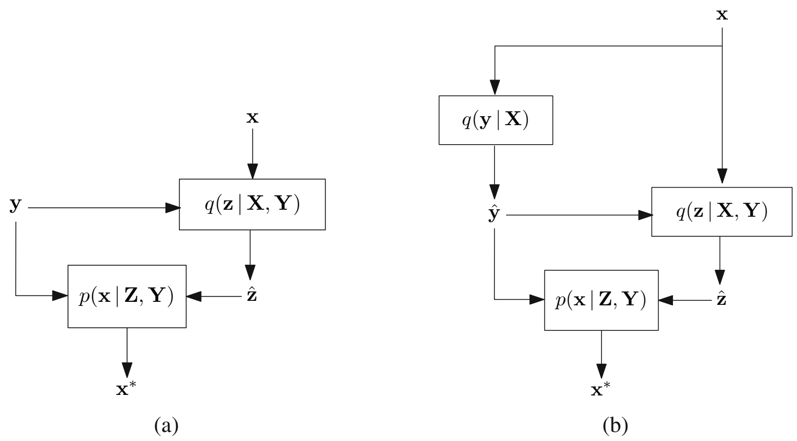

x
y
$q(\mathbf{z} \mid \mathbf{X}, \mathbf{Y})$
$p(\mathbf{x} \mid \mathbf{Z}, \mathbf{Y})$
$\hat{\mathbf{z}}$
$\mathbf{x}^*$
(a)
$q(\mathbf{y} \mid \mathbf{X})$
$\hat{\mathbf{y}}$
(b)

Fig. 5.17 Overview of the computational diagram of the semi-supervised VAE model in the example. $\mathbf{x}^*$ is the reconstruction of the original sample $\mathbf{x}$. For labeled samples, we have once again the CVAE seen in Sect. 5.3.1, where $\hat{\mathbf{z}}$ represents the estimated value of $\mathbf{z}$. For unlabeled data, however, $\mathbf{y}$ is unknown, and its value must first be inferred from the categorical distribution $q(\mathbf{y} \mid \mathbf{X})$, which gives the estimate $\hat{\mathbf{y}}$. Although each box is implemented by a separate fully connected NN, the complete model is optimized simultaneously. (**a**) Diagram for labeled samples. (**b**) Diagram for unlabeled samples

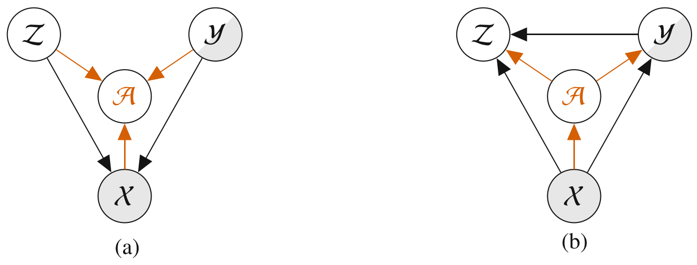


Fig. 5.18 Graphical representations of both inference and generative parts of the auxiliary deep generative model. The partially colored node $\mathbf{y}$ denotes the partially observed target labels. The novelty here w.r.t. Fig. 5.16 is the inclusion of the stochastic node $\mathbf{a}$, which gives more flexibility to the variational posterior. (**a**) Graphical model for the generative network. (**b**) Graphical model for the inference network

semi-supervised learning. Although only one variable was added to the model, two distributions were included: $q(\mathbf{a} \mid \mathbf{X})$ for the inference network and $p(\mathbf{a} \mid \mathbf{X}, \mathbf{Z}, \mathbf{Y})$ for its generative counterpart.

In what follows, we use the model in Figs. 5.17 and 5.16, i.e., without the auxiliary variable $\mathcal{A}$, to construct our toy example.

We model the generative process of the $\mathbf{x}_i$ as also being dependent on the partially observed latent class variable $\mathbf{Y}_i$ that specifies the digit. Both latent variables $\mathbf{Y}$ and $\mathbf{Z}$ are conditionally independent so that the first captures digits' semantics and the second digits' styles, independently. Hence, we can write $p(\mathbf{y}_i, \mathbf{z}_i \mid \mathbf{X}_i) =$

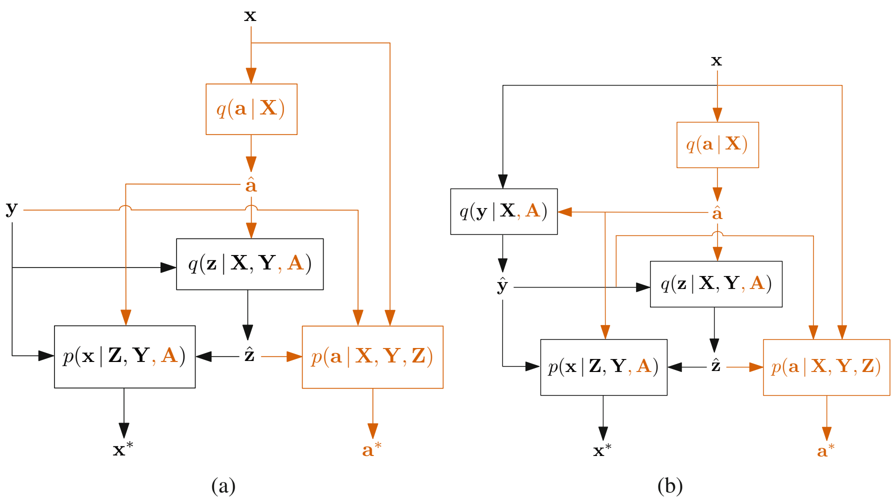

x
$q(\mathbf{a}\,|\,\mathbf{X})$
$\hat{\mathbf{a}}$
y
$q(\mathbf{z}\,|\,\mathbf{X},\mathbf{Y},\mathbf{A})$
$p(\mathbf{x}\,|\,\mathbf{Z},\mathbf{Y},\mathbf{A})$
$\hat{\mathbf{z}}$
$p(\mathbf{a}\,|\,\mathbf{X},\mathbf{Y},\mathbf{Z})$
$\mathbf{x}^*$
$\mathbf{a}^*$
(a)
x
$q(\mathbf{a}\,|\,\mathbf{X})$
$q(\mathbf{y}\,|\,\mathbf{X},\mathbf{A})$
$\hat{\mathbf{a}}$
$\hat{\mathbf{y}}$
$q(\mathbf{z}\,|\,\mathbf{X},\mathbf{Y},\mathbf{A})$
$p(\mathbf{x}\,|\,\mathbf{Z},\mathbf{Y},\mathbf{A})$
$\hat{\mathbf{z}}$
$p(\mathbf{a}\,|\,\mathbf{X},\mathbf{Y},\mathbf{Z})$
$\mathbf{x}^*$
$\mathbf{a}^*$
(b)

Fig. 5.19 Computational diagram overview of the auxiliary deep generative model. All differences w.r.t. Fig. 5.17, seen above in color, stem from the inclusion of the auxiliary variable $\mathbf{A}$, which represents an intermediate step in the inference process, illustrated in Fig. 5.18b. Then, $\mathbf{a}$ feeds $\mathbf{Y}$ and $\mathbf{Z}$ encoders (the $q(\cdot)$ blocks), as well as the $\mathbf{X}$ decoder (the $q(\mathbf{x}\,|\,\cdot)$ block). Similarly to Fig. 5.17, $\mathbf{x}^*$ and $\mathbf{a}^*$ both represent the reconstruction of the samples $\mathbf{x}$ and $\mathbf{a}$, respectively. (**a**) Diagram for labeled samples. (**b**) Diagram for unlabeled samples

$p(\mathbf{y}_i\,|\,\mathbf{X}_i)p(\mathbf{z}_i\,|\,\mathbf{X}_i)$. We define the prior $p(\mathbf{y}_i)$ to be a K-Categorical distribution Π_K over the class variable and the prior $p(\mathbf{z})$ a multivariate standard Gaussian, similarly to Sect. 5.3. Then, the complete generative process for one sample is defined by

$$p(\mathbf{x}_i,\mathbf{z}_i,\mathbf{y}_i) = p(\mathbf{x}_i\,|\,\mathbf{Z}_i,\mathbf{Y}_i)p(\mathbf{z})p(\mathbf{y}) \tag{5.30}$$

$$p(\mathbf{y}) = \Pi_K(\mathbf{y}\,|\,\pi) \tag{5.31}$$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}\,|\,\mathbf{0},\mathbf{I}) \tag{5.32}$$

$$p(\mathbf{x}_i\,|\,\mathbf{Z}_i,\mathbf{Y}_i) = \prod_{j=1}^{|\mathcal{X}|} p_j(\mathbf{x}_i\,|\,\mathbf{Z}_i,\mathbf{Y}_i), \tag{5.33}$$

where π is a probability vector and the elements of $\mathcal{X}$, i.e., the dimensions, iid. Specifically, we use Bernoulli variables to model the binary black-and-white pixel value.

Optimizing this model involves the marginal likelihood of observed and unobserved class variables $\mathbf{Y}$, $p(\mathbf{x},\mathbf{y})$ and $p(\mathbf{x})$, respectively. As in Sect. 5.3, we cannot directly compute those marginals and resort to the variational framework introduced in Sect. 3.2.1, like

$$\begin{aligned}\log p(\mathbf{x}_i,\mathbf{y}_i) &= \log \int p(\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i)d\mathbf{z}_i \\ &\geq \mathbb{E}_{q(\mathbf{z}_i\,|\,\mathbf{X}_i,\mathbf{Y}_i;\boldsymbol{\psi})}\left[\log\frac{p(\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i;\boldsymbol{\theta})}{q(\mathbf{z}_i\,|\,\mathbf{X}_i,\mathbf{Y}_i;\boldsymbol{\psi})}\right] = \mathcal{L}(\mathbf{x}_i,\mathbf{y}_i)\end{aligned} \tag{5.34}$$

$$\begin{aligned}\log p(\mathbf{x}_i) &= \log \int p(\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i)d\mathbf{z}_i d\mathbf{y}_i \\ &\geq \mathbb{E}_{q(\mathbf{z}_i,\mathbf{y}_i\,|\,\mathbf{X}_i;\boldsymbol{\psi})}\left[\log\frac{p(\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i;\boldsymbol{\theta})}{q(\mathbf{z}_i,\mathbf{y}_i\,|\,\mathbf{X}_i;\boldsymbol{\psi})}\right] \\ &= \mathbb{E}_{q(\mathbf{y}_i\,|\,\mathbf{X}_i;\boldsymbol{\psi})}\left[\mathbb{E}_{q(\mathbf{z}_i\,|\,\mathbf{Y}_i,\mathbf{X}_i;\boldsymbol{\psi})}\left[\log\frac{p(\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i;\boldsymbol{\theta})}{q(\mathbf{z}_i,\mathbf{y}_i\,|\,\mathbf{X}_i;\boldsymbol{\psi})}\right]\right] = \mathcal{U}(\mathbf{x}_i),\end{aligned} \tag{5.35}$$

where $q(\cdot)$ is again the proposal distribution learned by the inference model. For detailed explanation on how to obtain the inequalities in (5.35) and (5.34), read Sect. 3.2.1.1.

We train one recognition model for each latent variable and assume that their distribution follows

$$q(\mathbf{y}_i\,|\,\mathbf{X}_i) = \Pi_K(\mathbf{y}_i\,|\,\pi(\mathbf{x}_i;\boldsymbol{\psi})) \tag{5.36}$$

$$q(\mathbf{z}_i\,|\,\mathbf{Y}_i,\mathbf{X}_i) = \mathcal{N}\left(\mathbf{z}_i\,|\,\mu(\mathbf{x}_i,\mathbf{y}_i;\boldsymbol{\phi}),\operatorname{diag}\left(\sigma^2\left(\mathbf{y}_i,\mathbf{x}_i;\boldsymbol{\phi}\right)\right)\right). \tag{5.37}$$

Thus, we are able to perform inference in the latent space using (5.37) conditioned on the additional attributes $\mathbf{y}_i$, similarly to the CVAE in Sect. 5.3.1. Besides, we can infer the unknown label of $\mathbf{x}_i$ through the distribution defined by (5.36).

Unfortunately, the reparameterization trick does not apply to $q(\mathbf{y}_i\,|\,\mathbf{X}_i)$ because the distribution is discrete. We can use the score function estimator of the gradient instead, and however it entails high-variance estimates (see Appendix A.1). Alternatively, we can marginalize over $\mathbf{Y}_i$ in (5.37) and perform inference on $q(\mathbf{z}_i\,|\,\mathbf{Y}_i,\mathbf{X}_i;\boldsymbol{\psi})$ for each value of $\mathbf{y}$ [25]. However, marginalizing over all classes rapidly becomes costly since it is necessary to repeat the same operation K times, where K is the number of classes in a K-Categorical distribution, defined in (5.27). Another option is to relax the discrete distributions $p(\mathbf{y})$ and $q(\mathbf{z}_i\,|\,\mathbf{Y}_i,\mathbf{X}_i;\boldsymbol{\psi})$ onto continuous approximations using the Gumbel-softmax trick [19, 33] (see Sect. 5.5.3.1), making it possible again to apply the pathwise gradient estimator. The continuous relaxation allows us to take MC samples instead of marginalizing. When using 1 MC sample, it was experimentally verified to increase the overall training speed by 2× for 10 classes and 10× for 100 classes compared to marginalization [19].

There is still a practical issue in the proposed model: direct optimization of the label predictive distribution $q(\mathbf{y}_i\,|\,\mathbf{X}_i)$ is restricted to the unlabeled portion of the data via (5.35). Let S be the set of indexes of all labeled samples in the data set.

Table 5.2 Results of both semi-supervised VAE and supervised NN classifier on the Fashion-MNIST and MNIST for different amounts of training labels. The values inside parentheses represent the percentage of the original set size used

Model accuracy on test set				
	Fashion-MNIST		MNIST	
Label Count	Semi-supervised	Supervised	Semi-supervised	Supervised
6000 (10%)	79.2	77.1	93.6	91.4
3000 (5%)	77.9	71.6	91.4	88.0
600 (1%)	72.0	56.4	86.1	44.6
300 (0.5%)	70.9	46.7	81.7	30.1
100 (0.17%)	63.5	21.3	68.0	20.4

For $i \in S$, the model does not directly learn to infer classes. Thus, we augment the objective by adding an auxiliary cross-entropy term that constrains $q(\mathbf{y}_i \mid \mathbf{X}_i)$ to distributions that correctly classify the sample $\mathbf{x}_i$ according to the observed class label $\mathbf{y}_i$, which leads to

$$\mathcal{J} = \left[\sum_{i \in S} \mathcal{U}(\mathbf{x}_i) + \sum_{i \notin S} \mathcal{L}(\mathbf{x}_i, \mathbf{y}_i) \right] + \alpha \sum_{i \notin S} \log q(\mathbf{y}_i \mid \mathbf{X}_i), \tag{5.38}$$

where the weight α is a hyper-parameter that balances the regularization strength of the cross-entropy term.

Although we arbitrarily appended the cross-entropy term to (5.38) to enhance learning, we could have achieved the same result directly from the variational framework by also inferring the parameters π in (5.36) coupling it with a symmetric Dirichlet prior for $p(\pi)$, instead of defining a categorical prior of $\mathbf{Y}$ [25].

Table 5.2 presents the classification accuracy obtained from the semi-supervised VAE described in this section for the MNIST and Fashion-MNIST data sets at different levels of annotation.

We construct the model by following the diagram in Fig. 5.17, using 2 hidden-layer fully connected networks with ReLU activations [38] for each module. The Gumbel-softmax approximation to the categorical distribution $\Pi_K(\mathbf{y}_i \mid \pi(\mathbf{x}_i; \boldsymbol{\psi}))$ introduces a temperature hyper-parameter T_k, which we set to $T_k = 0.6$. Additionally, we set the latent space dimension $d = 32$, draw $T = 1$ MC samples from the approximating distributions $q(\mathbf{y}_i \mid \mathbf{X}_i)$ and $q(\mathbf{z}_i \mid \mathbf{Y}_i, \mathbf{X}_i)$, and use the regularization weight $\alpha = 25$. Regarding the optimization procedure, we employ the Adam [21] optimizer, training for $epc = 40$ epochs with fixed learning rate $lr = 0.003$ and batch size $bs = 128$.

The generative approach allows to leverage the latent information from the unlabeled data and obtain reasonable performances at an annotation effective standpoint. This translates to cheaper and faster development cycles, since cleaning and labeling are by far the toughest and more expensive parts of any ML project in business and industrial applications. Together with active learning, the approach in which the system presents to the user which samples should be annotated next for optimal performance gain, semi-supervision is a promising and refreshing frontier to the field, with several real-world use cases.

5.8 Closing Remarks

In this chapter we discussed the family of VAE models, how the models arise by applying VI to the latent variable modeling, and how they are related to one another. Moreover, we presented their major drawbacks, i.e., inexpressive posterior, posterior collapse, and discrete latent variable, as well as different manners to mitigate them. Many of these issues are current research topics.

In our experiments, we illustrated the generative and inference capabilities of VAEs and their main variations as well as properties of the ELBO and the latent space. Finally, we demonstrated the incredible potential of generative modeling for semi-supervised learning, a learning approach that uses very few labeled examples and takes advantage of the unlabeled samples.

There exist several other types of generative algorithms, the most popular being Generative Adversarial Networks [12], whose training dynamic can also be interpreted as through probabilistic lenses. Recently, pure flow-based models [9, 22] have gained a lot of attention of the research community due to their ability to perform inference and/or generation impressively fast and straightforward training.

As final words, we would link to point out that throughout the book we have seen the value of the Bayesian approach to probabilistic modeling and how it seamlessly allows us to reason under uncertainty, make predictions, and simulate new data, all achieved through the marginalization and conditioning operations. Besides, model fitting and comparison naturally arise from within the framework, which also has the additional advantage of being better equipped to handle data-poor regimes.

Bayesian deep learning is proving to be a central topic in current ML conferences with venues for applications crystallizing by the day: out-of-domain detection, adversarial robustness, compound exploration, audio synthesis, and image super-resolution, among many others. From the start of the writing of this book to the moment of its publication, many new interesting applications and methods came to light, rendering impossible the mission of keeping up with advancements made by the scientific community. However, we hope that our presentation was able to motivate the reader to confidently go further in this field. This is an exciting time to statisticians and ML practitioners alike, as there is a new wave of innovation ahead of us.

5.9 Final Words

Throughout the book we have seen the value of the Bayesian approach to probabilistic modeling and how it seamlessly allows us to reason under uncertainty, make predictions, and simulate new data, all achieved through the marginalization and conditioning operations. Besides, model fitting and comparison naturally arise from within the framework, which also has the additional advantage of being better equipped to handle data-poor regimes.

Bayesian DL is proving to be a central topic in current ML conferences with venues for applications crystallizing by the day: out-of-domain detection, adversarial robustness, compound exploration, audio synthesis, and image super-resolution, among many others. From the start of the writing of this book to the moment of its publication, many new interesting applications and methods came to light, rendering impossible the mission of keeping up with advancements made by the scientific community. However, we hope that our presentation was able to motivate the reader to confidently go further in this field. This is an exciting time to statisticians and ML practitioners alike, as there is a new wave of innovation ahead of us.

References

1. Alemi A, Fischer I, Dillon J, Murphy K (2017) Deep variational information bottleneck. In: Proceedings of the international conference on learning representations, Toulon, France
2. Alemi A, Poole B, Fischer I, Dillon J, Saurous RA, Murphy K (2018) Fixing a broken ELBO. In: Dy J, Krause A (eds) Proceedings of the international conference on machine learning, Stockholm, Sweden, vol 80, pp 159–168
3. Beal MJ, Ghahramani Z (2003) The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. In: Bayesian Statistics 7: the Seventh Valencia International Meeting, Tenerife, Spain pp. 453–464
4. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural networks. In: Proceedings of the international conference on machine learning, Lille, France, vol 37, pp 1613–1622
5. Breiman L (2001) Random forests. Machine Learning 45(1):5–32. https://doi.org/10.1023/A:1010933404324
6. Burda Y, Grosse R, Salakhutdinov R (2016) Importance weighted autoencoders. In: Proceedings of the international conference on learning representations, San Juan, Puerto Rico
7. Burgess CP, Higgins I, Pal A, Matthey L, Watters N, Desjardins G, Lerchner A (2018) Understanding disentangling in β-VAE. arXiv e-prints 1804.03599
8. Chen J, Chen J, Chao H, Yang M (2018) Image blind denoising with generative adversarial network based noise modeling. In: Proceedings of the conference on computer vision and pattern recognition, Salt Lake City, USA
9. Dinh L, Sohl-Dickstein J, Bengio S (2017) Density estimation using real NVP. In: Proceedings of the international conference on learning representations, Toulon, France
10. Eslami SMA, Heess N, Weber T, Tassa Y, Szepesvari D, kavukcuoglu k, Hinton GE (2016) Attend, infer, repeat: Fast scene understanding with generative models. In: Advances in neural information processing systems, pp 3225–3233

11. Eslami SMA, Jimenez Rezende D, Besse F, Viola F, Morcos AS, Garnelo M, Ruderman A, Rusu AA, Danihelka I, Gregor K, Reichert DP, Buesing L, Weber T, Vinyals O, Rosenbaum D, Rabinowitz N, King H, Hillier C, Botvinick M, Wierstra D, Kavukcuoglu K, Hassabis D (2018) Neural scene representation and rendering. Science 360(6394):1204–1210
12. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, Montreal, Canada, pp 2672–2680
13. Gómez-Bombarelli R, Wei JN, Duvenaud D, Hernández-Lobato JM, Sánchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel TD, Adams RP, Aspuru-Guzik A (2018) Automatic chemical design using a data-driven continuous representation of molecules. ACS Central Sci 4(2):268–276
14. Ha D, Schmidhuber J (2018) Recurrent world models facilitate policy evolution. In: Advances in neural information processing systems, Montreal, Canada, pp 2450–2462
15. He J, Spokoyny D, Neubig G, Berg-Kirkpatrick T (2019) Lagging inference networks and posterior collapse in variational autoencoders. In: Proceedings of the international conference on learning representations, New Orleans, USA
16. Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S (2017) GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Advances in neural information processing systems, Long Beach, USA, pp 6626–6637
17. Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, Mohamed S, Lerchner A (2017) β-VAE: Learning basic visual concepts with a constrained variational framework. In: Proceedings of the international conference on learning representations, Toulon, France
18. Houthooft R, Chen X, Chen X, Duan Y, Schulman J, De Turck F, Abbeel P (2016) VIME: Variational information maximizing exploration. In: Advances in neural information processing systems, Barcelona, Spain, pp 1109–1117
19. Jang E, Gu S, Poole B (2017) Categorical reparameterization with Gumbel-softmax. In: Proceedings of the international conference on learning representations, Toulon, France
20. Kalchbrenner N, van den Oord A, Simonyan K, Danihelka I, Vinyals O, Graves A, Kavukcuoglu K (2017) Video pixel networks. In: Proceedings of the international conference on machine learning, Sydney, NSW, Australia, vol 70, pp 1771–1779
21. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Proceedings of the international conference on learning representations, San Diego, USA
22. Kingma DP, Dhariwal P (2018) Glow: Generative flow with invertible 1x1 convolutions. In: Advances in neural information processing systems, Montreal, Canada, pp 10215–10224
23. Kingma DP, Welling M (2014) Auto-encoding variational Bayes. In: Proceedings of the international conference on learning representations, Banff, Canada
24. Kingma DP, Welling M (2019) An introduction to variational autoencoders. Found Trends Mach Learn 12(4):307–392. https://doi.org/10.1561/2200000056
25. Kingma DP, Mohamed S, Jimenez Rezende D, Welling M (2014) Semi-supervised learning with deep generative models. In: Advances in neural information processing systems, Montreal, Canada, pp 3581–3589
26. Kingma DP, Salimans T, Jozefowicz R, Chen X, Sutskever I, Welling M (2016) Improved variational inference with inverse autoregressive flow. In: Advances in neural information processing systems, Barcelona, Spain, pp 4743–4751
27. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
28. Ledig C, Theis L, Huszar F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z, Shi W (2017) Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the conference on computer vision and pattern recognition, Honolulu, USA
29. Lee AX, Zhang R, Ebert F, Abbeel P, Finn C, Levine S (2018) Stochastic adversarial video prediction. arXiv e-prints `1804.01523`

30. Maaløe L, Sønderby CK, Sønderby SK, Winther O (2016) Auxiliary deep generative models. In: Proceedings of the international conference on machine learning, New York, USA, vol 48, pp 1445–1453
31. Maaten Lvd, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
32. Maddison CJ, Tarlow D, Minka T (2014) A* sampling. In: Advances in neural information processing systems, Montreal, Canada, pp 3086–3094
33. Maddison C, Mnih A, Teh YW (2017) The concrete distribution: a continuous relaxation of discrete random variables. In: Proceedings of the international conference on learning representations, Toulon, France
34. McInnes L, Healy J, Melville J (2018) UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv e-prints `1802.03426`
35. Migon HS, Gamerman D, Louzada F (2014) Statistical inference: An integrated approach. CRC press, Boca Raton, USA
36. Mnih A, Gregor K (2014) Neural variational inference and learning in belief networks. In: Proceedings of the international conference on machine learning, Bejing, China, vol 32, pp 1791–1799
37. Mnih A, Rezende D (2016) Variational inference for Monte Carlo objectives. In: Proceedings of the international conference on machine learning, New York, USA, vol 48, pp 2188–2196
38. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the international conference on machine learning, Haifa, Israel, pp 807–814
39. Nowozin S (2018) Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In: Proceedings of the international conference on learning representations, Vancouver, Canada
40. Papamakarios G, Pavlakou T, Murray I (2017) Masked autoregressive flow for density estimation. In: Advances in neural information processing systems, Long Beach, USA, pp 2338–2347
41. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) PyTorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems, Vancouver, Canada, pp 8024–8035
42. Rainforth T, Kosiorek A, Le TA, Maddison C, Igl M, Wood F, Teh YW (2018) Tighter variational bounds are not necessarily better. In: Proceedings of the international conference on machine learning, Stockholm, Sweden, vol 80, pp 4277–4285
43. Ranganath R, Tran D, Blei D (2016) Hierarchical variational models. In: Proceedings of the international conference on machine learning, New York, USA, vol 48, pp 324–333
44. Regier J, Miller A, McAuliffe J, Adams R, Hoffman M, Lang D, Schlegel D, Prabhat M (2015) Celeste: Variational inference for a generative model of astronomical images. In: Proceedings of the international conference on machine learning, Lille, France, vol 37, pp 2095–2103
45. Rezende D, Mohamed S (2015) Variational inference with normalizing flows. In: Proceedings of the international conference on machine learning, Lille, France, vol 37, pp 1530–1538
46. Riesselman AJ, Ingraham JB, Marks DS (2018) Deep generative models of genetic variation capture the effects of mutations. Nature Methods 15:816–822
47. Rosca M, Lakshminarayanan B, Warde-Farley D, Mohamed S (2017) Variational approaches for auto-encoding generative adversarial networks. arXiv e-prints `1706.04987`
48. Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X, Chen X (2016) Improved techniques for training GANs. In: Advances in neural information processing systems, Barcelona, Spain, pp 2234–2242
49. Scholkopf B, Sung KK, Burges CJ, Girosi F, Niyogi P, Poggio T, Vapnik V (1997) Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Trans Signal Process 45(11):2758–2765
50. Sohn K, Lee H, Yan X (2015) Learning structured output representation using deep conditional generative models. In: Advances in neural information processing systems, Montreal, Canada, pp 3483–3491

51. Sønderby CK, Raiko T, Maaløe L, Sønderby SK, Winther O (2016) Ladder variational autoencoders. In: Advances in neural information processing systems, Barcelona, Spain, pp 3738–3746
52. Sønderby CK, Poole B, Mnih A (2017) Continuous relaxation training of discrete latent variable image models. In: Neural information processing systems - workshop on bayesian deep learning, Long Beach, USA
53. Theis L, Oord Avd, Bethge M (2016) A note on the evaluation of generative models. In: Proceedings of the international conference on learning representations, San Juan, Puerto Rico
54. Tishby N, Pereira FC, Bialek W (2000) The information bottleneck method. arXiv e-prints `physics/0004057`
55. Tschannen M, Agustsson E, Lucic M (2018) Deep generative models for distribution-preserving lossy compression. In: Advances in neural information processing systems, Montreal, Canada, pp 5929–5940
56. van den Berg R, Hasenclever L, Tomczak J, Welling M (2018) Sylvester normalizing flow for variational inference. In: Proceedings of the international conference on learning representations, Monterey, USA
57. van den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K (2016) WaveNet: A generative model for raw audio. In: ISCA speech synthesis workshop, Sunnyvale, USA, pp 125–125
58. van den Oord A, Vinyals O, kavukcuoglu k (2017) Neural discrete representation learning. In: Advances in neural information processing systems, Long Beach, USA, pp 6306–6315
59. Wan L, Zeiler M, Zhang S, Cun YL, Fergus R (2013) Regularization of neural networks using dropconnect. In: Proceedings of the international conference on machine learning, Atlanta, USA, vol 28, pp 1058–1066
60. Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv e-prints `1708.07747`