ON INCORPORATING INDUCTIVE BIASES INTO VAES

Ning Miao[1*] **Emile Mathieu**[1] **N. Siddharth**[2] **Yee Whye Teh**[1] **Tom Rainforth**[1*]

ABSTRACT

We explain why directly changing the prior can be a surprisingly ineffective mechanism for incorporating inductive biases into variational auto-encoders (VAEs), and introduce a simple and effective alternative approach: *Intermediary Latent Space VAEs* (InteL-VAEs). InteL-VAEs use an intermediary set of latent variables to control the stochasticity of the encoding process, before mapping these in turn to the latent representation using a parametric function that encapsulates our desired inductive bias(es). This allows us to impose properties like sparsity or clustering on learned representations, and incorporate human knowledge into the generative model. Whereas changing the prior only indirectly encourages behavior through regularizing the encoder, InteL-VAEs are able to directly enforce desired characteristics. Moreover, they bypass the computation and encoder design issues caused by non-Gaussian priors, while allowing for additional flexibility through training of the parametric mapping function. We show that these advantages, in turn, lead to both better generative models and better representations being learned.

1 INTRODUCTION

VAEs provide a rich class of deep generative models (DGMs) with many variants (Kingma & Welling, 2014; Rezende & Mohamed, 2015; Burda et al., 2016; Gulrajani et al., 2016; Vahdat & Kautz, 2020). Based on an encoder-decoder structure, VAEs encode datapoints into latent embeddings before decoding them back to data space. By parameterizing the encoder and decoder using expressive neural networks, VAEs provide a powerful basis for learning both generative models and representations.

The standard VAE framework assumes an isotropic Gaussian prior. However, this can cause issues, such as when one desires the learned representations to exhibit some properties of interest, for example sparsity (Tonolini et al., 2020) or clustering (Dilokthanakul et al., 2016), or when the data distribution has very different topological properties from a Gaussian, for example multi-modality (Shi et al., 2020) or group structure (Falorsi et al., 2018). Therefore, a variety of recent works have looked to use non-Gaussian priors (van den Oord et al., 2017; Tomczak & Welling, 2018; Casale et al., 2018; Razavi et al., 2019; Bauer & Mnih, 2019), often with the motivation of adding inductive biases into the model (Davidson et al., 2018b; Mathieu et al., 2019b; Nagano et al., 2019; Skopek et al., 2019).

In this work, we argue that this approach of using non-Gaussian priors can be a problematic, and even ineffective, mechanism for adding *inductive biases* into VAEs. Firstly, non-Gaussian priors will often necessitate complex encoder models to maintain consistency with the prior's shape and dependency structure (Webb et al., 2018), which typically no longer permit simple parameterization. Secondly, the latent encodings are still not guaranteed to follow the desired structure because the 'prior' only appears in the training objective as a regularizer on the encoder. Indeed, Mathieu et al. (2019b) find that changing the prior is typically insufficient in practice to learn the desired representations at a *population level*, with mismatches occurring between the data distribution and learned model.

To provide an alternative, more effective, approach that does not suffer from these pathologies, we introduce *Intermediary Latent Space VAEs* (InteL-VAEs), an extension to the standard VAE framework that allows a wide range of powerful inductive biases to be incorporated while maintaining an isotropic Gaussian prior. This is achieved by introducing an *intermediary* set of latent variables that deal with the stochasticity of the encoding process *before* incorporating the desired inductive biases via a parametric function that maps these intermediary latents to the latent representation itself, with the decoder taking this final representation as input. See Fig. 1 for an example.

[1]Department of Statistics, University of Oxford, [2]University of Edinburgh
[*]Correspondence to: Ning Miao <ning.miao@stats.ox.ac.uk>, Tom Rainforth <rainforth@stats.ox.ac.uk>

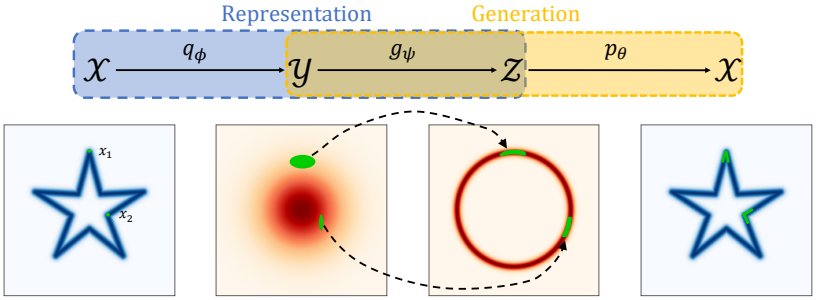


Figure 1: Example InteL-VAE with star-like data. We consider the auto-encoding for two example datapoints (x_1 and x_2, shown in green), which are first stochastically mapped to Y using a Gaussian encoder. This embedding is then pushed forward to Z using the *non-stochastic* mapping g , which is a radial mapping to enforce a spherical distribution. Decoding is then done in the standard way from Z, with the complexity of the decoder mapping simplified by the induced structural properties of Z.

The InteL-VAE framework provides a variety of advantages over directly replacing the prior. Firstly, it directly enforces our inductive biases on the representations, rather than relying on the regularizing effect of the prior to encourage this implicitly. Secondly, it provides a natural congruence between the generative and representational models via sharing of the mapping function, side-stepping the issues that non-Gaussian priors can cause for the inference model. Finally, it allows for more general and more flexible inductive biases to be incorporated, by removing the need to express them with an explicit density function and allowing for parts of the mapping to be learned during training.

To further introduce a number of novel specific realizations of the InteL-VAE framework, showing how they can be used to incorporate various inductive biases, enforcing latent representations that are, for example, multiply connected, multi-modal, sparse, or hierarchical. Experimental results show their superiority compared with baseline methods in both generation and feature quality, most notably providing state-of-the-art performance for learning sparse representations in the VAE framework.

To summarize, we a) highlight the need for inductive biases in VAEs and explain why directly changing the prior is a suboptimal means for incorporating them; b) propose InteL-VAEs as a simple but effective general framework to introduce inductive biases; and c) introduce specific InteL-VAE variants which can learn improved generative models and representations over existing baselines on a number of tasks. Accompanying code is provided at https://github.com/NingMiao/InteL-VAE.

2 The Need for Inductive Biases in VAEs

Variational auto-encoders (VAEs) are deep stochastic auto-encoders that can be used for learning both deep generative models and low-dimensional representations of complex data. Their key components are an encoder, $q(z|x)$, which probabilistically maps from data $x \in \mathcal{X}$ to latents $z \in \mathcal{Z}$; a decoder, $p(x|z)$, which probabilistically maps from latents to data; and a prior, $p(z)$, that completes the generative model, $p(z)p(x|z)$, and regularizes the encoder during training. The encoder and decoder are parameterized by deep neural networks and are simultaneously trained using a dataset $\{x_1, x_2, \ldots, x_N\}$ and a variational lower bound on the log-likelihood, most commonly,

$$L(x; \theta, \phi) := \mathbb{E}_{z \sim q(z|x)}[\log p(x|z)] - D_{\mathrm{KL}}(q(z|x) \| p(z)). \tag{1}$$

Namely, we optimize $L(\theta, \phi) := \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[L(x; \theta, \phi)]$, where $p_{\mathrm{data}}(x)$ represents the empirical data distribution. Here the prior is typically fixed to a standard Gaussian, i.e. $p(z) = \mathcal{N}(z; 0, I)$.

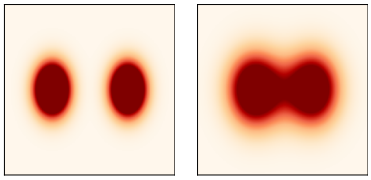
While it is well documented that this standard VAE setup with a 'Gaussian' latent space can be suboptimal (Davidson et al., 2018a; Mathieu et al., 2019b; Tomczak & Welling, 2018; Bauer & Mnih, 2019; Tonolini et al., 2020), there is perhaps less of a unified high-level view on exactly when, why, and how one should change it to incorporate inductive biases. Note here that the prior does not play the same role as in a Bayesian model: because the latents themselves are somewhat arbitrary and the model is learned from data, it does not encapsulate our initial beliefs in the way one might expect.

We argue that there are two core reasons why inductive biases can be important for VAEs: (a) standard VAEs can fail to encourage, and even prohibit, desired structure in the *representations* we learn; and (b) standard VAEs do not allow one to impart prior information or desired topological characteristic into the *generative model*.

Considering the former, one often has some a priori desired characteristics, or constraints, on the representations learned (Bengio et al., 2013). For example, sparse features can be desirable because they can improve data efficiency (Yip & Sussman, 1997), and provide robustness to noise (Wright et al., 2009; Ahmad & Scheinkman, 2019) and attacks (Gopalakrishnan et al., 2018). In other settings one might desire clustered (Jiang et al., 2017), disentangled (Ansari & Soh, 2019; Kim & Mnih, 2018; Higgins et al., 2018) or hierarchical representations (Song & Li, 2013; Sønderby et al., 2016; Zhao et al., 2017). The KL-divergence term in Eq. (1) regularizes the encoding distribution towards the prior and, as a standard Gaussian distribution typically does not exhibit our desired characteristics, this regularization can significantly hinder our ability to learn representations with the desired properties.

Not only can this be problematic at an individual sample level, it can cause even more pronounced issues at the *population level*: desired structural characteristics of our representations often relate to the pushforward distribution of the data in the latent space, $q(z) := \mathbb{E}_{p_{\text{data}}(x)}[q(z|x)]$, which is both difficult to control and only implicitly regularized to the prior (Hoffman & Johnson, 2016).

Inductive biases can also be essential to the generation quality of VAEs: because the generation process of standard VAEs is essentially pushing-forward the Gaussian prior on Z to data space X by a 'smooth' decoder, there is an underlying inductive bias that standard VAEs prefer sample distributions with similar topology structures to Gaussians. As a result, VAEs can perform poorly when the data manifold exhibits certain different topological properties (Caterini et al., 2020). For example, they can struggle when data is clustered into unconnected components as shown in Fig. 2, or when data is not simply-connected. This renders learning effective mappings using finite datasets and conventional architectures (potentially prohibitively) difficult. In particular, it can necessitate large Lipschitz constants in the decoder, causing knock-on issues like unstable training and brittle models (Scaman & Virmaux, 2018), as well as posterior collapse (van den Oord et al., 2017; Alemi et al., 2018). In short, the Gaussian prior of a standard VAE can induce fundamental topological differences to the true data distribution (Falorsi et al., 2018; Shi et al., 2020).



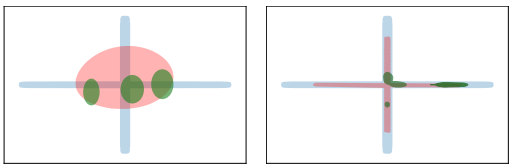
(a) Data (b) VAE

Figure 2: VAE learned generative distribution $\mathbb{E}_{p(z)}[p(x|z)]$ for mixture data.

3 Shortfalls of VAEs with non-Gaussian Priors

Though directly replacing the Gaussian prior with a different prior sounds like a simple solution, effectively introducing inductive biases can, unfortunately, be more complicated.

Firstly, the only influence of the prior during training is as a regularizer on the encoder through the $D_{\text{KL}}(q(z|x) \,\|\, p(z))$ term. This regularization is always competing with the need for effective reconstructions and only has an indirect influence on $q(z)$. As such, simply replacing the prior can be an ineffective way of inducing desired structure at the population level (Mathieu et al., 2019b), particularly if $p(z)$ is a complex distribution that it is difficult to fit (see, e.g., Fig. 3a). Mismatches between $q(z)$ and $p(z)$ can also have further deleterious effects on the learned generative model: the former represents the distribution of the data in latent space during training, while the latter is what is used by the learned generative model, leading to unrepresentative generations if there is mismatch.



(a) Directly replacing $p(z)$ (b) InteL-VAE

Figure 3: Prior-encoder mismatch. We train (a) a VAE with a sparse prior and (b) an InteL-VAE with a sparse inductive bias on 2 dimensional sparse data. Figure shows target latent distribution $p(z)$ (blue), learned variational embeddings $q(z|x)$ of exemplar data (green), and data pushforward $q(z)$ (red shadow) for each method. Simply replacing the prior does not help the VAE match prior structure on either a per-sample or population level, whereas InteL-VAE produces an effective match.

Secondly, it can be extremely difficult to construct appropriate encoder mappings and distributions for non-Gaussian priors. While the typical choice of a mean-field Gaussian for the encoder distribution is simple, easy to train, and often effective for Gaussian priors, it is often inappropriate for other choices of prior. For example, in Fig. 3, we consider replacement with a sparse prior. A VAE with a Gaussian encoder struggles to encode points in a manner that even remotely matches the prior. One

might suggest replacing the encoder distribution as well, but this has its own issues, most notably that other distributions can be hard to effectively parameterize or train. In particular, the form of the required encoding noise might become heavily spatially variant; in our sparse example, the noise must be elongated in a particular direction depending on where the mean embedding is. If the prior has constraints or topological properties distinct from the data, it can even be difficult to learn a mean encoder mapping that respects these, due to the continuous nature of neural networks.

4 THE INTEL-VAE FRAMEWORK

To solve the issues highlighted in the previous section, and provide a principled and effective method for adding inductive biases to VAEs, we propose *Intermediary Latent Space VAEs* (InteL-VAEs). The key idea behind InteL-VAEs is to introduce an *intermediary* set of latent variables $y \in Y$, used as a stepping stone in the construction of the *representation* $z \in Z$. Data is initially encoded in Y using a conventional VAE encoder (e.g. a mean-field Gaussian) before being passed through a *non-stochastic* mapping $g_\theta : Y \mapsto Z$ that incorporates our desired inductive biases and which can be trained, if needed, through its parameters θ. The prior is defined on Y and taken to be a standard Gaussian, $p(y) = \mathcal{N}(y; 0, I)$, while our representations, $z = g_\theta(y)$, correspond to a pushforward of y. By first encoding datapoints to y, rather than z directly, we can deal with all the encoder and prior stochasticity in this first, well-behaved, latent space, while maintaining z as our representation and using it for the decoder $p_\theta(x|z)$. In principle, g_θ can be any arbitrary parametric (or fixed) mapping, including non-differentiable or even discontinuous functions. However, to allow for reparameterized gradient estimators (Kingma & Welling, 2014; Rezende & Mohamed, 2015), we will restrict ourselves to g_θ that are sub-differentiable (and thus continuous) with respect to both their inputs and parameters. Note that setting g_θ to the identity mapping recovers a conventional VAE.

As shown in Fig. 1, the auto-encoding process is now $X \xrightarrow{q} Y \xrightarrow{g_\theta} Z \xrightarrow{p} X$. This three-step process no longer unambiguously fits into the encoder-decoder terminology of the standard VAE and permits a variety of interpretations; for now we take the convention of calling $q_\phi(y|x)$ the encoder and $p_\theta(x|z)$ the decoder, but also discuss some alternative interpretations below. We emphasize here that these no longer respectively match up with our representation model—which corresponds to passing an input into the encoder and then mapping the resulting encoding using g_θ—and our generative model—which corresponds to $\mathcal{N}(y; 0, I)p_\theta(x|z = g_\theta(y))$, such that we sample a y from the prior and then pass this through through g_θ and the decoder in turn.

The mapping g_θ introduces inductive biases into *both* the generative model and our representations by imposing a particular form on z, such as the spherical structure enforced in Fig. 1 (see also Sec. 6). It can be viewed as a *shared module* between them, ensuring congruence between the two. This congruence allows us to more directly introduce inductive biases through careful construction of g_θ, without complicating the process of learning an effective inference network. In particular, because Y is treated as our latent space for the purposes of training, we sidestep the inference issues that non-Gaussian priors usually cause. Moreover, because all samples must explicitly pass through g_θ during both training and generation, we can more directly ensure the desired structure is enforced without causing a mismatch in the latent distribution between training and deployment.

Training As with standard VAEs, training of an InteL-VAE is done by maximizing a variational lower bound (ELBO) on the log evidence, which we denote $\mathcal{L}_Y$. Most simply, we have

$$\log p_{\theta,\phi}(x) := \log\left(\mathbb{E}_{p(y)}[p_\theta(x|g_\theta(y))]\right) = \log\left(\mathbb{E}_{q_\phi(y|x)}\left[\frac{p_\theta(x|g_\theta(y))\mathcal{N}(y;0,I)}{q_\phi(y|x)}\right]\right) \geq \mathbb{E}_{q_\phi(y|x)}[\log p_\theta(x|g_\theta(y))] - D_{\mathrm{KL}}(q_\phi(y|x) \,\|\, \mathcal{N}(y;0,I)) =: \mathcal{L}_Y(x;\theta;\phi;\beta). \tag{2}$$

Note that the regularization is on y, but our representation corresponds to $z = g_\theta(y)$. Training corresponds to the optimization $\arg\max_{\theta,\phi,\beta} \mathbb{E}_{x \sim p_{\text{data}}(x)}[\mathcal{L}_Y(x;\theta;\phi;\beta)]$, which can be performed using stochastic gradient ascent with reparameterized gradients in the standard manner. Although inductive biases are introduced, the calculation, and optimization, of $\mathcal{L}_Y$ is thus equivalent to the standard ELBO. In particular, parameterizing $q_\phi(y|x)$ with a Gaussian distribution still yields an analytical $D_{\mathrm{KL}}(q_\phi(y|x) \,\|\, \mathcal{N}(y;0,I))$ term.

Alternative Interpretations It is interesting to note that our representation, $g_\theta(y)$, only appears in the context of the decoder in this training objective. As such, we see that an important alternative interpretation of InteL-VAEs is to consider g_θ as being a customized first layer in the decoder, and our

test–time representations as partial decodings of the latents y. This viewpoint allows it to be applied with more general bounds and VAE variants (e.g. Burda et al. (2016); Le et al. (2018); Maddison et al. (2017); Naesseth et al. (2018); Zhao et al. (2019)), as it requires only a carefully customized decoder architecture during training and an adjusted mechanism for constructing representations at test–time.

Yet another interpretation is to think about InteL-VAEs as implicitly defining a conventional VAE with latents z, but where both the non-Gaussian prior, $p(z)$, and our encoder distribution, $q(z|x)$, are themselves defined implicitly as pushforwards along g, which acts as a shared module that instills a natural compatibility between the two. Formally we have the following theorem.

Theorem 1. *Let $p(z)$ and $q(z|x)$ represent the respective pushforward distributions of $\mathcal{N}(0; I)$ and $q(y|x)$ induced by the mapping $g : \mathcal{Y} \to \mathcal{Z}$. The following holds for all measurable g:*

$$D_{KL}(q(z|x) \,\|\, p(z)) \quad D_{KL}(q(y|x) \,\|\, \mathcal{N}(y; 0; I)). \tag{3}$$

If g is also an invertible function then the above becomes an equality and $\mathcal{L}_{\mathcal{Y}}$ equals the standard ELBO on the space of $\mathcal{Z}$ as follows

$$\mathcal{L}_{\mathcal{Y}}(x; \cdot; \cdot; \cdot) = \mathbb{E}_{q(z|x)}[\log p(x|z)] \quad D_{KL}(q(z|x) \,\|\, p(z)). \tag{4}$$

The proof is given in Appendix A. Here, (3) shows that the divergence in our representation space $\mathcal{Z}$ is never more than that in $\mathcal{Y}$, or equivalently that the implied ELBO on the space of $\mathcal{Z}$ is always at least as tight as that on $\mathcal{Y}$; (4) shows they are exactly equal if g is invertible. As the magnitude of $D_{\mathrm{KL}}(q(y|x) \,\|\, \mathcal{N}(y; 0; I))$ in an InteL-VAE will remain comparable to the KL divergence in a standard Gaussian prior VAE setup, this, in turn, ensures that $D_{\mathrm{KL}}(q(z|x) \,\|\, p(z))$ does not become overly large. This is in stark contrast to the conventional non-Gaussian prior setup, where it can be difficult to avoid $D_{\mathrm{KL}}(q(z|x) \,\|\, p(z))$ exploding without undermining reconstruction (Mathieu et al., 2019b). The intuition here is that having the stochasticity in the encoder *before* it is passed through g ensures that the form of the noise in the embedding is inherently appropriate for the space: the same mapping is used to warp this noise as to define the generative model in the first place. For example, when g is a sparse mapping, the Gaussian noise in $q(y|x)$ will be compressed to a sparse subspace by g, leading to a sparse variational posterior $q(z|x)$ as shown in Fig. 3b. In particular, $q(y|x)$ does not need to learn any complex spatial variations that result from properties of $\mathcal{Z}$. In turn, InteL-VAEs further alleviate issues of mismatch between $p(z)$ and $q(z)$.

Further Benefits A key benefit of InteL-VAEs is that the extracted features are *guaranteed* to have the desired structure. Take the spherical case for example, all extracted features $g(\cdot)$ lie within a small neighborhood of the unit sphere. By comparison, methods based on training loss modifications, e.g. Mathieu et al. (2019b), often fail to generate features with the targeted properties.

A more subtle advantage is that we do not need to explicitly specify $p(z)$. This can be extremely helpful when we want to specify complex inductive biases: designing a non-stochastic mapping is typically much easier than a density function, particularly for complex spaces. Further, this can make it much easier to parameterize and learn aspects of $p(z)$ in a data-driven manner (see e.g. Sec. 6.3).

5 Related Work

Inductive biases There is much prior work on introducing human knowledge to deep learning models by structural design, such as CNNs (LeCun et al., 1989), RNNs (Hochreiter & Schmidhuber, 1997) and transformers (Vaswani et al., 2017). However, most of these designs are on the *sample* level, utilizing low–level information such as transformation invariances or internal correlations in each sample. By contrast, InteL-VAEs provide a convenient way to incorporate *population* level knowledge—information about the global properties of data distributions can be effectively utilized.

Non-Gaussian priors There is an abundance of prior work utilizing non-Gaussian priors to improve the fit and generation capabilities of VAEs, including MoG priors (Dilokthanakul et al., 2016; Shi et al., 2020), sparse priors (Mathieu et al., 2019b; Tonolini et al., 2020; Barello et al., 2018), Gaussian-process priors (Casale et al., 2018) and autoregressive priors (Razavi et al., 2019; van den Oord et al., 2017). However, these methods often require specialized algorithms to train and are primarily applicable only to specific kinds of data. Moreover, as we have explained, changing the prior alone often provides insufficient pressure on its own to induce the desired characteristics. Others have proposed non-Gaussian priors to reduce the prior-posterior gap, such as Vamp-VAE (Tomczak & Welling, 2018) and LARS (Bauer & Mnih, 2019), but these are tangential to our inductive bias aims.

Non-Euclidean latents A related line of work has focused on non-Euclidean latent spaces. For instance Davidson et al. (2018a) leveraged a von Mises-Fisher distribution on a hyperspherical latent space, Falorsi et al. (2018) endowed the latent space with a SO(3) group structure, and Mathieu et al. (2019a); Ovinnikov (2019); Nagano et al. (2019) with hyperbolic geometry. Other spaces like product of constant curvature spaces (Skopek et al., 2019) and embedded manifolds (Rey et al., 2019) have also been considered. However, these works generally require careful design and training.

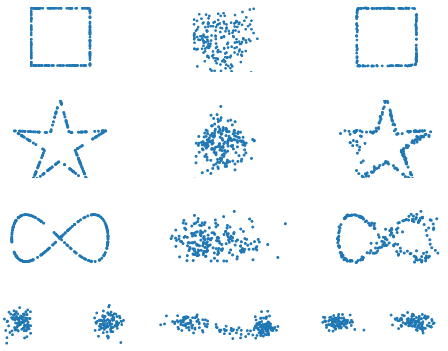
Normalizing flows Our use of a non-stochastic mapping shares some interesting links to normalizing flows (NFs) (Rezende & Mohamed, 2015; Papamakarios et al., 2019; Grathwohl et al., 2018; Dinh et al., 2017; Huang et al., 2018; Papamakarios et al., 2018). Indeed a NF would be a valid choice for g , albeit an unlikely one due to their architectural constraints. However, unlike previous use of NFs in VAEs, our g is crucially *shared* between the generative and representational models, rather than just being used in the encoder, while the KL divergence in our framework is taken before, not after, the mapping. Moreover, the underlying motivation, and type of mapping typically used, differs substantially: our mapping is used to introduce inductive biases, not purely to improve inference. Our mapping is also more general than a NF (e.g. it need not be invertible) and does not introduce additional constraints or computational issues.

6 Specific Realizations of the InteL-VAE Framework

We now present several novel example InteL-VAEs, introducing various inductive biases through different choices of g . We will start with artificial, but surprisingly challenging, examples where some precise topological properties of the target distributions are known, incorporating them directly through a fixed g . We will then move onto experiments where we impose a fixed clustering inductive bias when training on image data, allowing us to learn InteL-VAEs that account effectively for multi-modality in the data distribution. Finally, we consider the example of learning sparse representations of high–dimensional data. Here we will see that it is imperative to exploit the ability of InteL-VAEs to learn aspects of g during training, providing a flexible inductive bias framework, rather than a pre-fixed mapping. By comparing InteL-VAEs with strong baselines, we show that InteL-VAEs are effective in introducing these desired inductive biases, and consequently both improve generation quality and learn better data representations for downstream tasks. One note of particular importance is that we find that InteL-VAEs provide state-of-the-art performance for learning sparse VAE representations. A further example of using InteL-VAEs to learn hierarchical representations is presented in Appendix B, while full details on the various examples are given in Appendix C.

6.1 Multiple–Connectivity

Data is often most naturally described on non-Euclidean spaces such as circles, e.g. wind directions (Mardia & Jupp, 2000), and other multiply-connected shapes, e.g. holes in disease databases (Liu et al., 1997). For reasons previously explained in Sec. 2, standard VAEs cannot practically model such topologies, which prevents them from learning generative models which match even the simplest data distributions with non-trivial topological structures, as shown in Fig. 4b.



(a) Data (b) VAE (c) InteL-VAE

Figure 4: Training data and samples from learned generative models of vanilla-VAE and InteL-VAE for multiply-connected and clustered distributions. InteL-VAE uses [Rows 1,2] circular prior with one hole, [Row 3] multiply-connected prior with two holes, and [Row 4] clustered prior. Vamp-VAE behaves similarly to a vanilla VAE; its results are presented in Fig. 4.

Luckily, by designing g to map the Gaussian prior to a simple representative distribution in a topological class, we can easily equip InteL-VAEs with the knowledge to approximate any data distributions with similar topological properties. Specifically, by defining g as the orthogonal projection to S^1, $g(z) = z/(\|z\|_2 + \epsilon)$, we map the Gaussian prior approximately to a uniform distribution to S^1, where ϵ is a small positive constant to ensure the continuity of g near the origin. From Rows 1 and 2 of Fig. 4, we find that this inductive bias gives InteL-VAEs the ability to learn various distributions with a hole.

We can add further holes by simply 'gluing' point pairs. For example, for two holes we can use

$$g_2(y) = \text{Concat}\left(g_1(y)_{[:,1]},\ g_1(y)_{[:,2]}\sqrt{(4=3 \quad (1 \quad jg_1(y)_{[:,1]}j)^2)} \quad 1=\overline{3}\right); \tag{5}$$

which first map y to approximately S^1, and then glues $(0; 1)$ and $(0; \quad 1)$ together to create new holes (see Fig. C.1 for an illustration). Furthermore, we can continue to glue points together to achieve a higher number of holes h, and thus more complex connectivity. Row 3 of Fig. 4 gives an example of learning an infinity sign by introducing a 'two-hole' inductive bias.

Compared with vanilla-VAE and Vamp-VAE, which try to find a convex hull for real data distributions, InteL-VAEs can deal with distributions with highly non-convex and very non-smooth supports (see Fig. 4 and Appendix C.1). We emphasize here that our inductive bias does not contain the information about the precise shape of the data, only the number of holes. We thus see that InteL-VAEs can provide substantial improvements in performance by incorporating only basic prior information about the topological properties of the data, which point out a way to approximate distributions on more complex structures, such as linear groups (Gupta & Mishra, 2018).

6.2 Multi–Modality

Many real-world datasets exhibit multi-modality. For example, data with distinct classes are often naturally clustered into (nearly) disconnected components representing each class. However, vanilla VAEs generally fail to fit multi-modal data due to the topological issues explained in Sec. 2. Previous work (Johnson et al., 2017; Mathieu et al., 2019b) has thus proposed the use of a multi-modal prior, such as a mixture of Gaussian (MoG) distribution, so as to capture all components of the data. Nonetheless, VAEs with such priors often still struggle to model multi-modal data because of mismatch between $q \ (z)$ and $p(z)$ or training instability issues.

We tackle this problem by using a mapping g which contains a clustering inductive bias. The high-level idea is to design a mapping g with a localized high Lipschitz constant that 'splits' the continuous Gaussian distribution into K disconnected parts and then pushes them away from each other. In particular, we split Y it into K equally sized sectors using its first two dimensions (noting it is not needed to split on all dimensions to form clusters), as shown in Fig. 5. For any point y, we can easily get the center direction $\mathrm{r}(y)$ of the sector that y belongs to and the distance $\mathrm{dis}(y)$ between y and the sector boundary. Then we define $g \ (y)$ as:

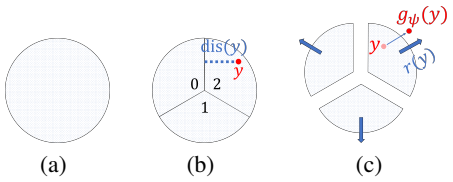


Figure 5: Illustration of clustered mapping where $K = 3$. The circle represents a density isoline of a Gaussian. Note that not all points in the sector are moved equally: points close to the boundaries between sectors are moved less, with points on the boundary themselves not moved at all.

$$g \ (y) = y + c_1 \mathrm{dis}(y)^{c_2} \mathrm{r}(y); \tag{6}$$

where c_1 and c_2 are empirical constants. We can see that although g has very different function on different sectors, it is still continuous on the whole plane with $g \ (y) = y$ on sector boundaries, which is desirable for gradient-based training. See Appendix C.2 for more details.

To assess the performance of our approach, we first consider a simple 2-component MoG synthetic dataset in the last row of Fig. 4. We see that the vanilla VAE fails to learn a clustered distribution that fits the data, while the InteL-VAE sorts this issue and fits the data well.

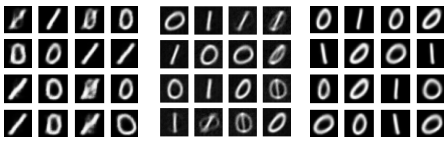
To provide a more real-world example, we train an InteL-VAE and a variety of baselines on the **MNIST** dataset, comparing the generation quality of the learned models using the FID score (Heusel et al., 2017) in Table 1. We find that the GM-VAE (Dilokthanakul et al., 2016) and MoG-VAE (VAE with a fixed MoG prior) achieve performance gains by using non-Gaussian priors. The Vamp-VAE (Tomczak & Welling, 2018) and a VAE with a Sylvester Normalizing Flow (Berg et al., 2018) encoder provide further gains by making the prior and encoder distributions more flexible respectively. However, the InteL-VAE comfortably outperforms all of them.

Method	FID Score (#)	
VAE	42:0	1:1
GM-VAE	41:0	4:7
MoG-VAE	41:2	3:3
Vamp-VAE	38:8	2:4
VAE with Sylvester NF	35:0	0:9
InteL-VAE	32:2	1:5

Table 1: Generation quality on MNIST. Shown is mean FID score (lower better) standard deviation over 10 runs.

Method	Data		VAE		GM-VAE		MoG-VAE		Vamp-VAE		Flow		InteL-VAE	
Uncertainty(%)	0.2	0.1	2.5	0.4	3.5	1.8	4.5	0.8	2.4	0.3	16.2	2.1	**0.9**	**0.8**
'1' proportion(%)	50.0	0.2	48.8	0.2	48.1	0.3	47.7	0.4	48.8	0.1	42.5	1.0	**49.5**	**0.4**

Table 2: Quantitative results on **MNIST-01**. **Uncertainty** is the proportion of images whose labels are 'indistinguishable' by the pre-trained classifier, defined as having prediction confidence $< 80\%$. **'1' proportion** is the proportion of images classified as '1'.



(a) VAE (b) MoG-VAE (c) InteL-VAE

Figure 6: Generated samples for **MNIST-01**.

To gain insight into how InteL-VAEs achieve superior generation quality, we perform analysis on a simplified setting where we select only the '0' and '1' digits from the **MNIST** dataset to form a strongly clustered dataset, **MNIST-01**. We further decrease the latent dimension to 1 to make the problem more challenging. Fig. 6 shows that here the vanilla VAE generates some samples which look like interpolations between '0' and '1', meaning that it still tries to learn a connected distribution containing '0' and '1'. Further, the general generation quality is poor, with blurred images and a lack of diversity in generated samples (e.g. all the '1's have the same slant). Despite using a clustered prior, the MoG-VAE still produces unwanted interpolations between the classes. By contrast, InteL-VAE generates digits that are unambiguous and crisper.

To quantify these results, we further train a logistic classifier on **MNIST-01** and use it to classify images generated by each method. For each method, we calculate the proportion of samples produced by the generative model that are assigned to each class by this pre-trained classifier, as well as the proportion of samples for which the classifier is uncertain. From Table 2 we see that InteL-VAE significantly outperforms its competitors in the ability to generate balanced and unambiguous digits. To extend this example further, and show the ability of InteL-VAEs to learn aspects of g during training, we further consider parameterizing and then learning the relative size of the clusters. Table 3 shows that this can be successfully learned by InteL-VAEs on **MNIST-01**.

True Prop.	Learned Prop.	
0.5	0.47	0.01
0.4	0.36	0.10
0.25	0.25	0.08
0.2	0.16	0.11
0	0.02	0.01

Table 3: Learned proportions of '0's on **MNIST-01** for different ground truths. Error bars are std. dev. from 10 runs.

6.3 Sparsity

Sparse features are often well-suited to data efficiency on downstream tasks (Huang & Aviyente, 2006), in addition to being naturally easier to visualize and manipulate than dense features (Ng et al., 2011). However, existing VAE models for sparse representations trade off generation quality to achieve this sparsity (Mathieu et al., 2019b; Tonolini et al., 2020; Barello et al., 2018). Here, we show that InteL-VAEs can instead *simultaneously* increase feature sparsity and generation quality. Moreover, they are able to achieve state-of-the-art scores on sparsity metrics.

Compared with our previous examples, the g here needs to be more flexible so that it can learn to map points in a data-specific way and induce sparsity without unduly harming reconstruction. To achieve this, we use the simple form for the mapping: $g(y) = y \odot \mathrm{DS}(y)$, where $\odot$ is pointwise multiplication, and DS is a 'dimension selector' network that selects dimensions to deactivate given y. DS outputs values between $[0; 1]$ for each dimension, with 0 being fully deactivated and 1 fully activated; the more dimensions we deactivate, the sparser the representation. By learning DS during training, this setup allows us to learn a sparse representation in a data-driven manner. To control the degree of sparsity, we add a sparsity regularizer, L_{sp}, to the ELBO with weighting parameter λ (higher λ corresponds to more sparsity). Namely, we optimize $L_Y(\cdot\,;\,\cdot\,;\,\cdot) + \lambda L_{sp}(\cdot\,;\,\cdot)$, where

$$L_{sp}(\cdot\,;\,\cdot) := \mathrm{E}\left[\frac{1}{M}\sum_{i=1}^{M}(H(DS(y_i))) \quad H\left(\frac{1}{M}\sum_{i=1}^{M} DS(y_i)\right)\right]; \tag{7}$$

$H(v) = \sum_i (v_i / \|v\|_1) \log (v_i / \|v\|_1)$ is the normalized entropy of an positive vector v, and the expectation is over drawing a minibatch of samples $x_1; \ldots; x_M$ and then sampling each corresponding $y_i \sim q(y|x = x_i)$. L_{sp} encourages DS to deactivate more dimensions, while also encouraging diversity in which dimensions are activated for different data points, improving utilization of the latent space. Please see Appendix C.3 for more details and intuitions. Initial qualitative results are shown in Fig. 8, where we see that our InteL-VAE is able to learn sparse and intuitive representations.

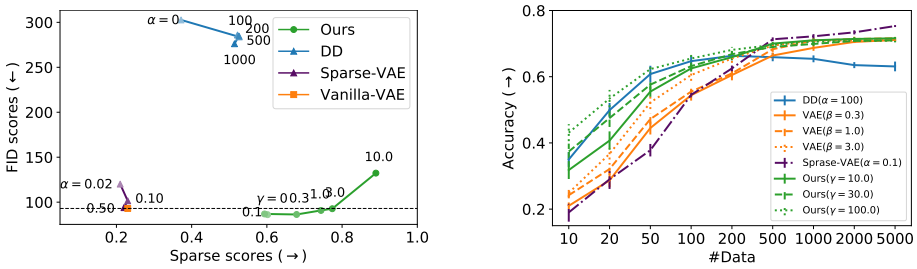


Figure 7: Results on **Fashion-MNIST**. The *left* figure shows FID and sparsity scores. Lower FID scores (#) represent better sample quality while higher sparse scores (!) indicate sparser features. The *right* figure shows the performance of sparse features from InteL-VAE on downstream classification tasks. See Appendix C.3 for details and results for **MNIST**.

To quantitatively assess the ability of our approach to yield sparse representations and good quality generations, we compare against vanilla VAEs, the specially customized sparse-VAE of Tonolini et al. (2020), and the sparse version of Mathieu et al. (2019b) (DD) on **Fashion-MNIST** (Xiao et al., 2017) and **MNIST**. As shown in Fig. 7 (*left*), we find that InteL-VAEs increase sparsity of the representations—measured by the Hoyer metric (Hurley & Rickard, 2009)—while increasing generative sample quality at the same time. Indeed, the FID score obtained by InteL-VAE outperforms the vanilla VAE when < 3:0, while the sparsity score substantially increases with , reaching extremely high levels. By comparison, DD significantly degrades generation quality and only provides a more modest increase in sparsity, while its sparsity also drops if the regularization coefficient is set too high. The level of sparsity achieved by sparse-VAEs was substantially less than both DD and InteL-VAEs.

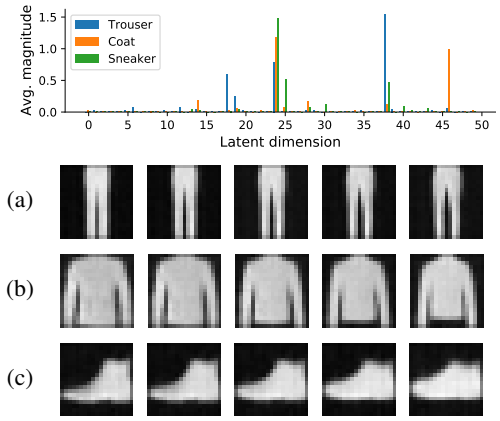


Figure 8: Qualitative evaluation of sparsity. [Top] Average magnitude of each latent dimension for three example classes in **Fashion-MNIST**; less than 10% dimensions are activated for each class. [Bottom] Activated dimensions are different between classes: (a-c) show the results of separately manipulating an activated dimension for each class. (a) Trouser separation (Dim 18). (b) Coat length (Dim 46). (c) Shoe style (formal/sport, Dim 25).

To further evaluate the quality of the learned features for downstream tasks, we trained a classifier to predict class labels from the latent representations. For this, we choose a random forest (Breiman, 2001) with maximum depth 4 as it is well-suited for sparse features. We vary the size of training data given to the classifier to measure the data efficiency of each model. Fig. 7 (*right*) shows that InteL-VAE typically outperforms other the models, especially in few-shot scenarios.

Finally, to verify InteL-VAE's effectiveness on larger and higher-resolution datasets, we also make comparisons on **CelebA** (Liu et al., 2015). From Table 4, we can see that InteL-VAE increase sparse scores to 0.46 without sacrificing generation quality. By comparison, the maximal sparse score that sparse-VAE gets is 0.30, with unacceptable sample quality. Interestingly, InteL-VAEs with ly low regulation achieved particularly good generative sample quality, outperforming even the Vamp-VAE and a VAE with a Sylvester NF encoder.

Method	FID (#)	Sparsity (")
VAE	68.6 1.1	0.22 0.01
Vamp-VAE	67.5 1.1	0.22 0.01
VAE with Sylvester NF	66.3 0.4	0.22 0.01
Sparse-VAE (= 0:01)	328 10.1	0.25 0.01
Sparse-VAE (= 0:2)	337 8.1	0.28 0.01
InteL-VAE (= 30)	64.9 0.4	0.25 0.01
InteL-VAE (= 70)	68.0 0.6	0.46 0.02

Table 4: Generation results on CelebA.

Conclusions In this paper, we proposed InteL-VAEs, a general schema for incorporating inductive biases into VAEs. Experiments show that InteL-VAEs can both provide representations with desired properties and improve generation quality, outperforming a variety of baselines such as directly changing the prior. This is achieved while maintaining the simplicity and stability of standard VAEs.

ETHICS STATEMENT

We do not believe that there are direct ethical concerns regarding our paper: the datasets we consider are all already well established and do not contain sensitive information, while the methods and ideas we introduce have no clear direct potential negative societal impacts of their own. From a bigger picture perspective, work like ours that looks to permit more effective incorporation of inductive biases into models can be thought of as allowing more direct human control on how models will behave after training. While this will typically be a force for good, for example by encouraging model interpretability and providing mechanisms to try and induce positive characteristics like fairness, in rare circumstances there may also be the potential for this to be used nefariously by deliberately encouraging undesirable behavior. However, we do not believe that our work is any more prone to such exploitation than existing methods or that the risk of it being used in such as a way is significant.

REPRODUCIBILITY STATEMENT

Full experimental details are given in Appendix C, while anonymized source code for reproducing all our experiments directly is provided at https://github.com/djkdsjwkjerkjermf/InteL-VAE. Together these should make it straightforward for others to reproduce our empirical results. We have been careful to provide quantitative metrics of performance whenever possible, rather than just relying on qualitative or anecdotal evidence. Repeat runs and error bars are provided whenever this is feasible, with the level of variability always found to be sufficiently small to draw reliable and statistically sound conclusions. In fact, the training stability and consistent performance of our general approach under retraining provides a clear advantage in itself compared to many of the baseline methods. Full formal proof for our only theoretical result is given in Appendix A, while the assumptions it makes are clearly stated and easily verifiable.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Subutai Ahmad and Luiz Scheinkman. How can we be so dense? the benefits of using highly sparse representations. *arXiv preprint arXiv:1903.11257*, 2019.

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pp. 159–168. PMLR, 2018.

Abdul Fatir Ansari and Harold Soh. Hyperprior induced unsupervised disentanglement of latent representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3175–3182, 2019.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Gabriel Barello, Adam S. Charles, and Jonathan W. Pillow. Sparse-coding variational auto-encoders. Preprint, Neuroscience, August 2018.

Matthias Bauer and Andriy Mnih. Resampled priors for variational autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 66–75. PMLR, 2019.

Mohamed Ishmael Belghazi, Sai Rajeswar, Olivier Mastropietro, Negar Rostamzadeh, Jovana Mitrovic, Aaron Courville, and AI Element. Hierarchical adversarially learned inference. *stat*, 1050:4, 2018.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

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

Francesco Paolo Casale, Adrian V Dalca, Luca Saglietti, Jennifer Listgarten, and Nicoló Fusi. Gaussian process prior variational autoencoders. In *NeurIPS*, 2018.

Anthony L Caterini, Robert Cornish, Dino Sejdinovic, and Arnaud Doucet. Variational inference with continuously-indexed normalizing flows. 2020.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. *arXiv:1804.00891 [cs, stat]*, September 2018a. URL http://arxiv.org/abs/1804.00891.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)*, 2018b.

Alfredo De la Fuente and Robert Aduviri. Replication/machine learning. 2019.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv:1605.08803 [cs, stat]*, February 2017. URL http://arxiv.org/abs/1605.08803.

Luca Falorsi, Pim de Haan, Tim R. Davidson, Nicola De Cao, Maurice Weiler, Patrick Forré, and Taco S. Cohen. Explorations in homeomorphic variational auto-encoding. *arXiv:1807.04689 [cs, stat]*, July 2018. URL http://arxiv.org/abs/1807.04689.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323. JMLR Workshop and Conference Proceedings, 2011.

Soorya Gopalakrishnan, Zhinus Marzi, Upamanyu Madhow, and Ramtin Pedarsani. Combating adversarial attacks using sparse representations. *arXiv preprint arXiv:1803.03880*, 2018.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *arXiv:1810.01367 [cs, stat]*, October 2018. URL http://arxiv.org/abs/1810.01367.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

Ved Prakash Gupta and Mukund Madhav Mishra. On the topology of certain matrix groups. *THE MATHEMATICS STUDENT*, pp. 61, 2018.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6629–6640, 2017.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. 2016.

Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1133–1141. IEEE, 2017.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. pp. 10, 2018.

Ke Huang and Selin Aviyente. Sparse representation for signal classification. *Advances in neural information processing systems*, 19:609–616, 2006.

Niall Hurley and Scott Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55:4723–4741, 2009.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *IJCAI*, 2017.

Matthew J. Johnson, David Duvenaud, Alexander B. Wiltschko, Sandeep R. Datta, and Ryan P. Adams. Composing graphical models with neural networks for structured representations and fast inference. *arXiv:1603.06277 [stat]*, July 2017. URL http://arxiv.org/abs/1603.06277.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2649–2658. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/kim18b.html.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Alexej Klushyn, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt. Learning hierarchical priors in vaes.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *International Conference on Learning Representations*, 2018.

Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-encoding sequential monte carlo. In *International Conference on Learning Representations*, 2018.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Bing Liu, Liang-Ping Ku, and Wynne Hsu. Discovering interesting holes in data. In *Proceedings of the Fifteenth international joint conference on Artifical intelligence-Volume 2*, pp. 930–935, 1997.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Whye Teh. Filtering variational objectives. In *NIPS*, 2017.

K. V. Mardia and Peter E. Jupp. *Directional Statistics*. Wiley Series in Probability and Statistics. J. Wiley, Chichester ; New York, 2000. ISBN 978-0-471-95333-3.

Emile Mathieu, Charline Le Lan, Chris J. Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with poincar\'e variational auto-encoders. January 2019a. URL https://arxiv.org/abs/1901.06033v3.

Emile Mathieu, Tom Rainforth, N Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders. In *International Conference on Machine Learning*, pp. 4402–4412. PMLR, 2019b.

Christian Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. Variational sequential monte carlo. In *International Conference on Artificial Intelligence and Statistics*, pp. 968–977. PMLR, 2018.

Yoshihiro Nagano, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama. A wrapped normal distribution on hyperbolic space for gradient-based learning. *arXiv:1902.02992 [cs, stat]*, May 2019. URL http://arxiv.org/abs/1902.02992.

Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

Ivan Ovinnikov. Poincar\'e wasserstein autoencoder. January 2019. URL https://arxiv.org/abs/1901.01427v2.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv:1705.07057 [cs, stat]*, June 2018. URL http://arxiv.org/abs/1705.07057.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pp. 324–333. PMLR, 2016.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NIPS*, 2019.

Luis A. Pérez Rey, Vlado Menkovski, and Jacobus W. Portegies. Diffusion variational autoencoders. *arXiv:1901.08991 [cs, stat]*, March 2019. URL http://arxiv.org/abs/1901.08991.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538. PMLR, 2015.

Igal Sason. On data-processing and majorization inequalities for f-divergences with applications. *Entropy*, 21(10):1022, October 2019. ISSN 1099-4300. doi: 10.3390/e21101022.

Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 3839–3848, 2018.

Wenxian Shi, Hao Zhou, Ning Miao, and Lei Li. Dispersed exponential family mixture vaes for interpretable text generation. In *International Conference on Machine Learning*, pp. 8840–8851. PMLR, 2020.

Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. Mixed-curvature variational autoencoders. November 2019. URL https://arxiv.org/abs/1911.08411v2.

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

Tiecheng Song and Hongliang Li. Wavelbp based hierarchical features for image classification. *Pattern Recognition Letters*, 34(12):1323–1328, 2013.

Jakub M Tomczak and Max Welling. Vae with a vampprior. In *21st International Conference on Artificial Intelligence and Statistics, AISTATS 2018*, 2018.

Francesco Tonolini, Bjørn Sand Jensen, and Roderick Murray-Smith. Variational sparse coding. In *Uncertainty in Artificial Intelligence*, pp. 690–700. PMLR, 2020.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6309–6318, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Stefan Webb, Adam Golinski, Robert Zinkov, Siddharth Narayanaswamy, Tom Rainforth, Yee Whye Teh, and Frank Wood. Faithful inversion of generative models for effective amortized inference. In *NeurIPS*, 2018.

John Wright, Allen Y. Yang, Arvind Ganesh, S. Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2): 210–227, 2009. doi: 10.1109/TPAMI.2008.79.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Kenneth Yip and Gerald Jay Sussman. Sparse representations for fast, one-shot learning. In *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence*, pp. 521–527, 1997.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning hierarchical features from generative models. In *International Conference on Machine Learning*, 2017.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 5885–5892, 2019.

APPENDIX A PROOFS

Theorem 1. *Let $p(z)$ and $q_\psi(z|x)$ represent the respective pushforward distributions of $\mathcal{N}(0; I)$ and $q(y|x)$ induced by the mapping $g_\psi : Y \to Z$. The following holds for all measurable g_ψ :*

$$D_{KL}(q_\psi(z|x) \,\|\, p(z)) \leq D_{KL}(q(y|x) \,\|\, \mathcal{N}(y; 0; I)). \tag{3}$$

If g_ψ is also an invertible function then the above becomes an equality and $\mathcal{L}_Y$ equals the standard ELBO on the space of Z as follows

$$\mathcal{L}_Y(x, \psi, \theta, \phi) = \mathbb{E}_{q_\psi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\psi(z|x) \,\|\, p(z)). \tag{4}$$

Proof. We first prove the inequality from Eq. (3), then we show that Eq. (3) is actually an equality when g_ψ is invertible, and finally we prove that the reconstruction term is unchanged by g_ψ .

Let us denote by $\mathcal{F}$ and $\mathcal{G}$ the sigma-algebras of respectively Y and Z, and we have by construction a measurable map $g_\psi : (Y, \mathcal{F}) \to (Z, \mathcal{G})$. We can actually define the measurable space $(Z, \mathcal{G})$ as the image of $(Y, \mathcal{F})$ by g_ψ , then g_ψ is automatically both surjective and measurable.[1] We also assume that there exists a measure on Y, which we denote , and denote with the corresponding pushforward measure by g_ψ on Z. We further have $(A) = (g_\psi^{-1}(A))$ for any $A \in \mathcal{G}$.[2]

We start by proving Eq. (3), where the Kullback-Leibler (KL) divergence between the two pushforward measures[3] $q_\psi \triangleq q \, g_\psi^{-1}$ and $p_\theta \triangleq p \, g_\psi^{-1}$ is upper bounded by $D_{\mathrm{KL}}(q(y|x) \,\|\, p(y))$, where here we have $p(y) = \mathcal{N}(y; 0; I)$ but we will use p as a convenient shorthand. At a high-level, we essentially have that Eq. (3) follows directly the data processing inequality (Sason, 2019) with a deterministic kernel $z = g_\psi(y)$. Nonetheless, we develop in what follows a proof which additionally gives sufficient conditions for when this inequality becomes non-strict. We can assume that $D_{\mathrm{KL}}(q(y|x) \,\|\, \mathcal{N}(y; 0; I))$ is finite, as otherwise the result is trivially true, which in turn implies $q \ll p$.[4] For any $A \in \mathcal{G}$, we have that if $p(A) = p \, g_\psi^{-1}(A) = p(g_\psi^{-1}(A)) = 0$ then this implies $q(g_\psi^{-1}(A)) = q \, g_\psi^{-1}(A) = q_\psi(A) = 0$. As such, we have that $q_\psi \ll p$ and so the $D_{\mathrm{KL}}(q_\psi(z|x) \,\|\, p(z))$ is also defined.

Our next significant step is to show that

$$\mathbb{E}_{p(y)}\left[\frac{q}{p} \,\middle|\, (g_\psi)\right] = \frac{q \, g_\psi^{-1}}{p \, g_\psi^{-1}} \, g_\psi ; \tag{A.1}$$

where (g_ψ) denotes the sigma-algebra generated by the function g_ψ . To do this, let $h : (Z, \mathcal{G}) \to (\mathbb{R}_+, \mathcal{B}(\mathbb{R}_+))$ be a measurable function s.t. $\mathbb{E}_{p(y)}\left[\frac{q}{p} \,\middle|\, (g_\psi)\right] = h \, g_\psi$. To show this, we will demonstrate that they lead to equivalent measures when integrated over any arbitrary set $A \in \mathcal{G}$:

$$\begin{aligned}
\int_Z \mathbb{1}_A \frac{q \, g_\psi^{-1}}{p \, g_\psi^{-1}} \, p \, g_\psi^{-1} \, d &= \int_Z \mathbb{1}_A \, q \, g_\psi^{-1} \, d = \int_Z \mathbb{1}_A \, d(q \, g_\psi^{-1}) \\
&\overset{(a)}{=} \int_Y (\mathbb{1}_A \, g_\psi) \, dq = \int_Y (\mathbb{1}_A \, g_\psi) \, q \, d \\
&\overset{(b)}{=} \int_Y (\mathbb{1}_A \, g_\psi) \, \frac{q}{p} \, p \, d \\
&\overset{(c)}{=} \int_Y (\mathbb{1}_A \, g_\psi) \, \mathbb{E}_{p(y)}\left[\frac{q}{p} \,\middle|\, (g_\psi)\right] \, p \, d \\
&\overset{(d)}{=} \int_Y (\mathbb{1}_A \, g_\psi) \, (h \, g_\psi) \, p \, d = \int_Y (\mathbb{1}_A \, g_\psi) \, (h \, g_\psi) \, dp \\
&\overset{(e)}{=} \int_Z \mathbb{1}_A \, h \, d(p \, g_\psi^{-1}) = \int_Z \mathbb{1}_A \, h \, (p \, g_\psi^{-1}) \, d ;
\end{aligned}$$

[1] We recall that g_ψ is said to be measurable if and only if for any $A \in \mathcal{G}$, $g_\psi^{-1}(A) \in \mathcal{F}$.

[2] The notation $g_\psi^{-1}(A)$ does not imply that g_ψ is invertible, but denotes the preimage of A which is defined as $g_\psi^{-1}(A) = \{y \in Y \mid g_\psi(y) \in A\}$.

[3] We denote the pushforward of a probability measure along a map g by g^{-1}.

[4] We denote the absolute continuity of measures with , where is said to be absolutely continuous w.r.t. , i.e. , if for any measurable set A, $(A) = 0$ implies $(A) = 0$.

where we have leveraged the definition of pushforward measures in (a & e); the absolute continuity of q w.r.t. p in (b); the conditional expectation definition in (c); and the definition of h in (d). By equating terms, we have that $q \quad g^{-1}=p \quad g^{-1} = h$, almost-surely with respect to $q \quad g^{-1}$ and thus that Eq. (A.1) is verified.

Let us define $f : x \mapsto x\log(x)$, which is strictly convex on $[0; 1)$ (as it can be prolonged with $f(0) = 0$). We have the following

$$
\begin{aligned}
D_{\mathrm{KL}}(q_{;} \ (zjx) \ k \ p \ (z)) &\overset{(a)}{=} \int_Z \log\left(\frac{q_{;}}{p}\right) q_{;} \quad d \\
&\overset{(b)}{=} \int_Z \log\left(\frac{q_{;}}{p}\right) \frac{q_{;}}{p} \ p \quad d \\
&\overset{(c)}{=} \int_Z f\left(\frac{q_{;}}{p}\right) p \quad d \quad = \int_Z f\left(\frac{q_{;}}{p}\right) d(p \quad g^{-1}) \\
&\overset{(d)}{=} \int_Y f\left(\frac{q_{;} \quad g}{p \quad g}\right) dp = \int_Y f\left(\frac{q \quad g^{-1}}{p \quad g^{-1}} \quad g\right) p \, d \\
&\overset{(e)}{=} \int_Y f\left(\mathrm{E}_{p(y)}\left[\frac{q}{p} \,\middle|\, (g \ )\right]\right) p \, d \\
&\overset{(f)}{\leq} \int_Y \mathrm{E}_{p(y)}\left[f\left(\frac{q}{p}\right) \,\middle|\, (g \ )\right] p \, d \\
&\overset{(g)}{=} \int_Y f\left(\frac{q}{p}\right) p \, d \\
&\overset{(h)}{=} \int_Y \log\left(\frac{q}{p}\right) \frac{q}{p} \, p \, d \\
&\overset{(i)}{=} \mathrm{E}_{q \ (yjx)}\left[\log\left(\frac{q \ (yjx)}{p(y)}\right)\right] \\
&\overset{(j)}{=} D_{\mathrm{KL}}(q \ (yjx) \ k \ p(y)) ;
\end{aligned}
$$

where we leveraged the definition of the KL divergence in (a & j); the absolute continuity of q w.r.t. p in (b & i); the definition of f in (c & h); the definition of the pushforward measure in (d); Eq. (A.1) in (e); the conditional Jensen inequality in (f) and the law of total expectation in (g). Note that this proof not only holds for the KL divergence, but for any f-divergences as they are defined as in (b) with f convex.

To prove Eq. (4), we now need to show that line (f) above becomes an equality when g is invertible. As f is strictly convex, this happens if and only if $\frac{q}{p} = \mathrm{E}_{p(y)}\left[\frac{q}{p} \,\middle|\, (g \ )\right]$. A sufficient condition for this to be true is for $\frac{q}{p}$ to be measurable w.r.t. $(g \ )$ which is satisfied when $g : Y \mapsto Z$ is invertible as $(g \ ) \quad F$, as required. We have thus shown that the KL divergences are equal when using an invertible g .

For the reconstruction term, we instead have

$$
\begin{aligned}
\mathrm{E}_{q \ (yjx)}[\log p \ (xjg \ (y))] &= \int_Y \log p \ (xjg \ (y)) q \ (yjx) d \\
&= \int_Z \log p \ (xjz) q_{;} \ (zjx) d \\
&= \mathrm{E}_{q_{;} \ (zjx)}[\log p \ (xjz)].
\end{aligned}
$$

Eq. (4) now follows from the fact that both the reconstruction and KL terms are equal.

□

APPENDIX B HIERARCHICAL REPRESENTATIONS

The isotropic Gaussian prior in standard VAEs assumes that representations are independent across dimensions (Kumar et al., 2018). However, this assumption is often unrealistic (Belghazi et al., 2018; Mathieu et al., 2019b). For example, in Fashion-MNIST, high-level features such as object category, may affect low-level features such as shape or height. Separately extracting such global and local information can be beneficial for visualization and data manipulation (Zhao et al., 2017). To try and capture this, we introduce an inductive bias that is tailored to model and learn hierarchical features. We note here that our aim is not to try and provide a state-of-the-art hierarchical VAE approach, as a wide variety of highly–customized and powerful approaches are already well–established, but to show how easily the InteL-VAE framework can be used to induce hierarchical representations in a simple, lightweight, manner.

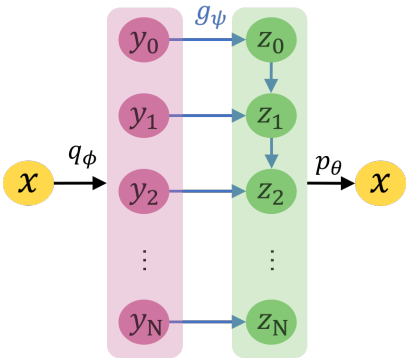


Figure B.1: Graphical model for hierarchical InteL-VAE

Mapping design Following existing ideas from hierarchical VAEs (Sønderby et al., 2016; Zhao et al., 2017), we propose a hierarchical mapping g . As shown in Fig. B.1, the intermediary Gaussian variable y is first split into a set of N layers $[y_0; y_1; ...; y_N]$. The mapping $z = g(y)$ is then recursively defined as $z_i = \text{NN}_i(z_{i-1}; y_i)$, where NN_i is a neural network combining information from higher-level feature z_{i-1} and new information from y_i. As a result, we get a hierarchical encoding $z = [z_0; z_1; ...; z_N]$, where high-level features influence low-level ones but not vice-versa. This g thus endows InteL-VAEs with hierarchical representations.

Experiments While conventional hierarchical VAEs, e.g. (Sønderby et al., 2016; Zhao et al., 2017; Vahdat & Kautz, 2020), use hierarchies to try and improve generation quality, our usage is explicitly from the representation perspective, with our experiments set up accordingly. Fig. B.2 shows some hierarchical features learned by InteL-VAE on **Fashion-MNIST**. We observe that high-level information such as categories have indeed been learned in the top-level features, while low-level features control more detailed aspects.

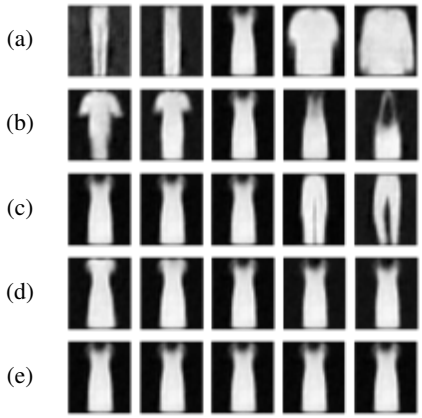


Figure B.2: Manipulating representations of a hierarchical InteL-VAE. The features are split into 5 levels, with each of (a) [highest] to (e) [lowest] corresponding to an example feature from each. We see that high-level features control more complex properties, such as class label or topological structure, while low-level features control simpler details, (e.g. (d) controls collar shape).

To provide more quantitative investigation, we also consider the **CelebA** dataset (Liu et al., 2015) and investigate performance on downstream tasks, comparing to vanilla-VAEs with different latent dimensions. For this, we train a linear classifier to predict all 40 binary labels from the learned features for each method. In order to eliminate the effect of latent dimensions, we compare InteL-VAE (with fixed latent dimension 128) and vanilla VAE with different latent dimensions $(1; 2; 4; 8; 16; 32; 64; 128)$. We show experiment results on some labels as well as the average accuracy on all labels in Table B.1 and Fig. B.3. We first find that the optimal latent dimension increases with the number of data points for the vanilla-VAEs, but is always worse than the InteL-VAE. Notably, the accuracy with InteL-VAE is quite robust, even as the number of data points gets dramatically low, indicating high data efficiency. To the best of our knowledge, this is the first result showing that a hierarchical inductive bias in VAE is beneficial to feature quality.

Related work Hierarchical VAEs (Vahdat & Kautz, 2020; Ranganath et al., 2016; Sønderby et al., 2016; Klushyn et al.; Zhao et al., 2017) seek to improve the fit and generation quality of VAEs by recursively correcting the generative distributions. However, they require careful design of neural

Model	Latent dim	Data size					
		50	100	500	1000	5000	10000
VAE	8	<u>0.791</u>	0.799	0.814	0.815	0.819	0.819
	16	0.788	<u>0.801</u>	0.820	0.824	0.829	0.831
	32	0.769	0.795	0.825	<u>0.832</u>	0.842	0.846
	64	0.767	0.794	<u>0.826</u>	<u>0.832</u>	<u>0.849</u>	<u>0.855</u>
	128	0.722	0.765	0.817	0.825	0.830	0.852
InteL-VAE	64	**0.817**	**0.824**	**0.841**	**0.846**	**0.854**	**0.857**

Table B.1: Average accuracy in predicting all 40 binary labels of **CelebA**. Overall best accuracy is shown in bold and best results of vanilla-VAEs are underlined for comparison. Each experiment is repeated 10 times and differences are significant at the 5% level for data size 1000.

layers, and the hierarchical KL divergence makes training deep hierarchical VAEs unstable (Vahdat & Kautz, 2020). In comparison, InteL-VAE with hierarchical mappings is extremely easy to implement without causing any computational instabilities, while its aims also differ noticeably: our approach successfully learns hierarchical *representations*—something that is rarely mentioned in prior works.

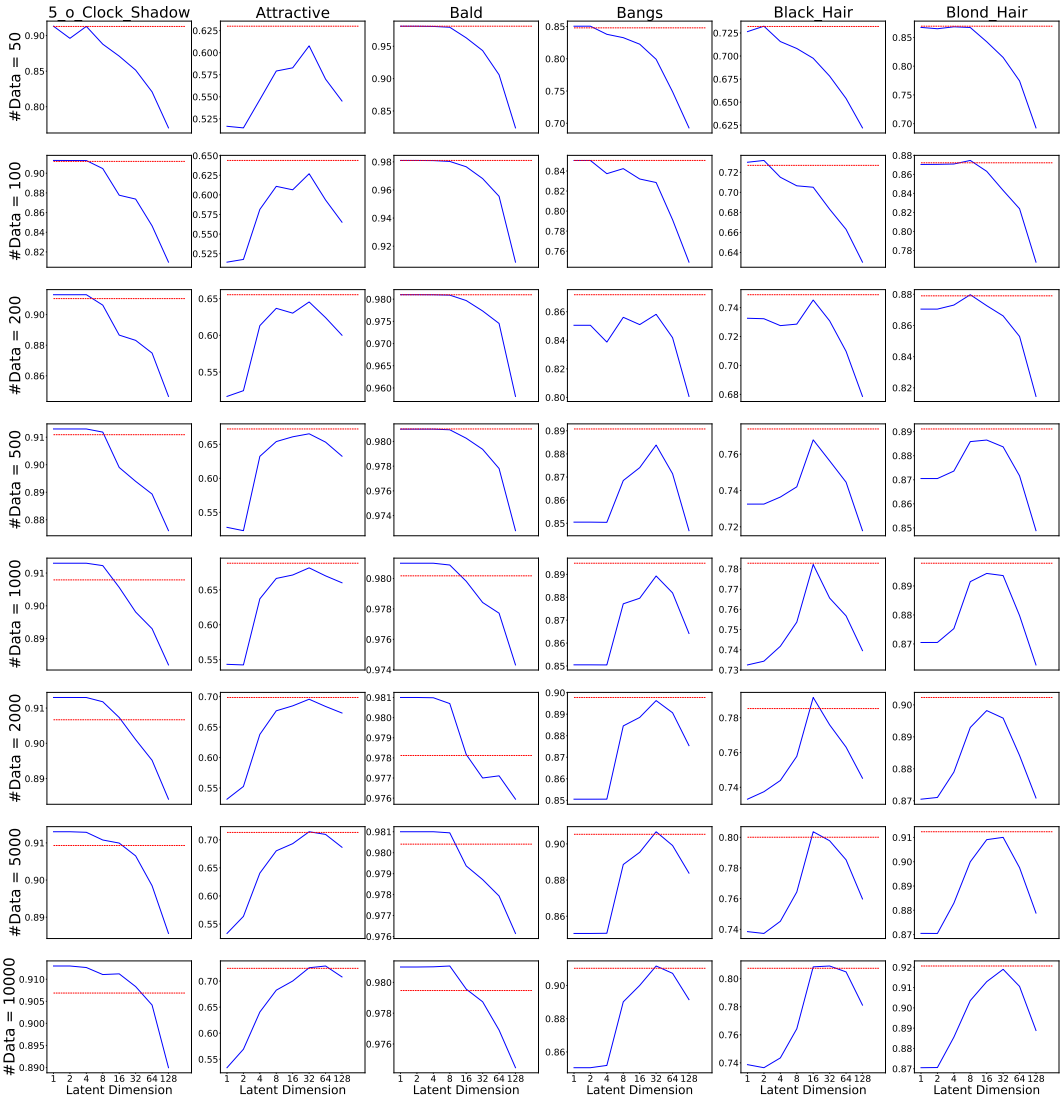


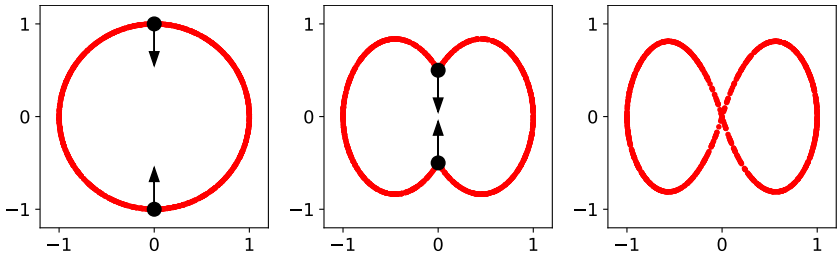
Figure B.3: InteL-VAE's performance of attribute prediction on **CelebA** dataset. Each column shows results on the same feature with different data sizes and each column shows results on different features. In each graph, test accuracy of vanilla-VAE with different latent dimensions are shown in blue line. And results of InteL-VAE with hierarchical prior are shown in red. We find that our method (red line) achieves comparable or even better results compared with vanilla-VAE with all latent dimensions.

APPENDIX C FULL METHOD AND EXPERIMENT DETAILS

In this section, we first provide complete details of the mapping designs used for our different InteL-VAE realizations along with some additional experiments. We then provide other general information about datasets, network structures, and experiment settings to facilitate results reproduction.

C.1 MULTIPLE-CONNECTIVITY

Mapping design Full details for this mapping were given in the main paper. Fig. C.1 provides a further illustration of the gluing process. Additional resulting including the Vamp-VAE are given in Fig. 4.



(a) Circular prior with $h = 1$ (b) Glue point pair (c) Implied prior with $h = 2$

Figure C.1: An illustration of the glue function in multiply-connected mappings.

C.2 MULTI-MODALITY

Mapping design In Sec. 6.2, we see the general idea of designing clustered mappings. In this part, we delve into the details of mapping design as well as extending it to 1 dimensional and high-dimensional cases. For simplicity's sake let us temporarily assume that the dimension of Y is 2. Our approach is based on splitting the original space into K equally sized sectors, where K is the number of clusters we wish to create, as shown in Fig. 5b. For any point y, we can get its component (sector) index $\mathrm{ci}(y)$ as well as its distance from the sector boundary $\mathrm{dis}(y)$. By further defining the radius direction for the k-th sector (cf Fig. 5c) as

$$(k) = \left(\cos\left(\frac{2}{K}\left(k+\frac{1}{2}\right)\right); \sin\left(\frac{2}{K}\left(k+\frac{1}{2}\right)\right)\right) \quad 8k\ 2\ f1;:::;Kg;$$

we can in turn define $g(y)$ as:

$$\mathrm{r}(y) = \quad (\mathrm{ci}(y)); \tag{C.1}$$

$$g(y) = y + c_1 \mathrm{dis}(y)^{c_2} \mathrm{r}(y); \tag{C.2}$$

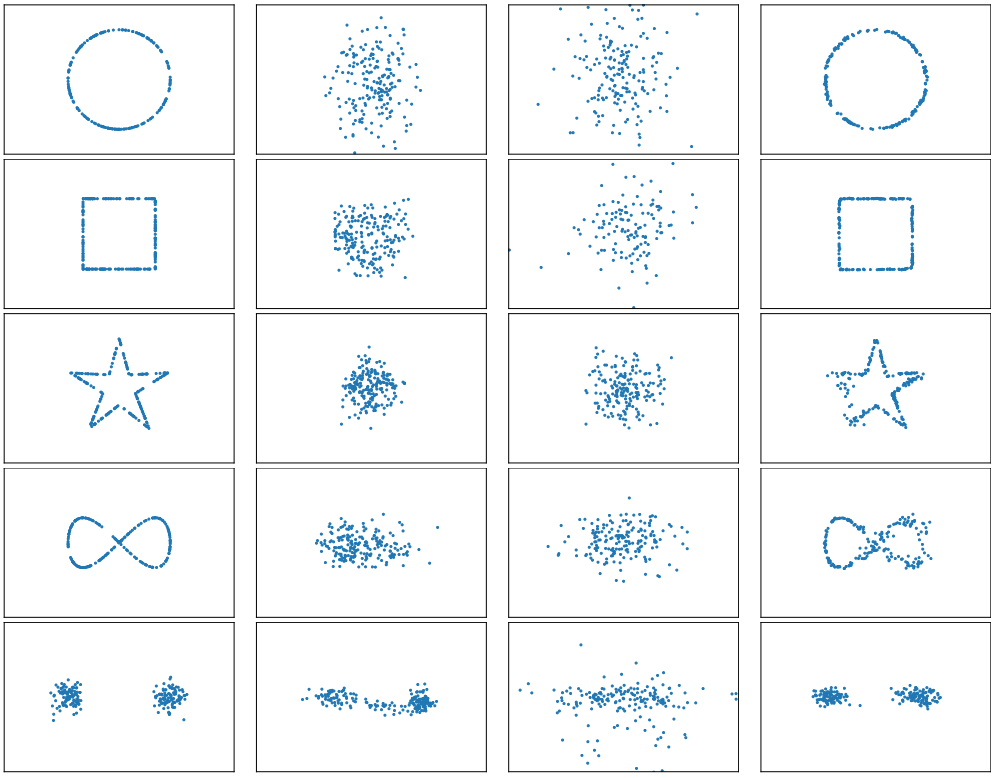
where c_1 and c_2 are constants, which are set to 5 and 0.2 in our experiments. we make sure g still continuous by keeping $g(y) = y$ on boundaries.

When dimension of Y is greater than 2, we have more diverse choice for g. When K is decomposable, i.e., $K = \prod_i K_i$, we can separately cut the plane expanded by Y_{2i} and Y_{2i+1} into K_i sectors by the Eq. (C.1). As a result, Y is split into $K = \prod_i k_i$ clusters. When $K = 2$, we find that g only changes the 1-st dimension of Y, so it can be applied to cases where latent dimension is 1.

Learnable proportions We can also make the mapping more flexible by learning rather than assigning the cluster proportions. To do so, we keep a learnable value u_i for each cluster and set the angle of the i-th sector as $2\ \mathrm{Softmax}(u)_i$. Things are simpler for the 1-dimensional case where we can uniformly translate y by a learnable bias b before splitting the space from the origin.

C.3 SPARSITY

Relationship to soft attention We note that our setup for the sparsity mapping shares some similarities with a soft attention layer (Bahdanau et al., 2014). However, there are also some



(a) Real distribution (b) VAE (c) Vamp-VAE (d) InteL-VAE

Figure C.2: Extension of Fig. 4 showing Vamp-VAE baseline and additional circular target distribution (top row, uses the same single hole g as the second and third rows).

important points of difference. Firstly, soft attention aims to find the weights to blend features from different time steps (for sequential data) or different positions (for image data). In contrast, the dimension selector (DS) selects which dimensions to activate or deactivate for the same latent vector. Secondly, the weights of features are usually calculated by inner products of features for soft attention, while DS relies on a network to directly output the logits.

Sparsity regularizer Our sparsity regularizer term, L_{sp}, is used to encourage our dimensionality selector network (DS) to produce sparse mappings. It is defined using a mini-batch of samples $fy_i g_{i=1}^{M}$ drawn during training as per (7). During training, the first term of L_{sp} decreases the number of activated dimensions for each sample, while the second term prevents the samples from all using the same set of activated dimensions, which would cause the model to degenerate to a vanilla VAE with a lower latent dimensionality.

We note that L_{sp} alone is not expected to induce sparsity without also using the carefully constructed g of the suggested InteL-VAE. We confirm this empirically by performing an ablation study on **MNIST** where we apply this regularization directly to a vanilla VAE. We find that even when using very large values of $> 30{:}0$ we can only slightly increase the sparsity score ($0{:}230$ $!$ $0{:}235$). Moreover, unlikely for the InteL-VAE, this substantially deteriorates generation quality, with the FID score raising to more than $80{:}0$ at the same time.

Sparse metric We use the Hoyer extrinsic metric (Hurley & Rickard, 2009) to measure the sparsity of representations. For a representation $z \; 2 \; \mathrm{R}^{D}$,

$$\text{Hoyer}(z) = \frac{p\overline{D} \quad jj\hat{z}jj_1 = jj\hat{z}jj_2}{p\overline{D} \quad 1}: \tag{C.3}$$

Here, following Mathieu et al. (2019b), we crucially first normalized each dimension d of z to have standard deviation 1, $\hat{z}_d = z_d = \; _d$, to ensure that we only measure sparsity that varies between data

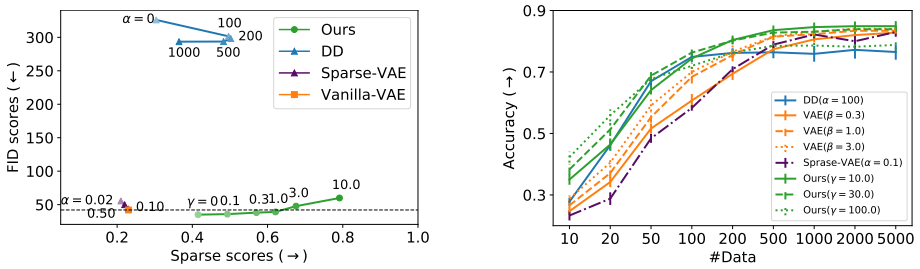


Figure C.3: Results on **MNIST**. The *left* figure shows FID and sparsity scores. Lower FID scores (#) represent better sample quality while higher sparse scores (!) indicate sparser features. The *right* figure shows the performance of sparse features from InteL-VAE on downstream classification tasks. See Sec. 6.3 for details and results for **MNIST**.

Parameters	Synthetic	MNIST	Fashion-MNIST	MNIST-01	CelebA
Dataset sizes	Unlimited	55k/5k/10k	55k/5k/10k	10k/1k/2k	163k/20k/20k
Input space	$\mathbf{R}^2$	Binary 28x28	Binary 28x28	Binary 28x28	RGB 64x64x3
Encoder net	MLP	CNN	CNN	CNN	CNN
Decoder net	MLP	CNN	CNN	CNN	CNN
Latent dimension	2-10	50	50	1-10	1-128
Batch size	10-500	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam	Adam
Learning rate	1e-3	1e-3	1e-3	1e-3	1e-3

Table C.1: Hyperparameters used for different experiments.

points (as is desired), rather than any tendency to uniformly 'switch off' certain latent dimensions (which is tangential to our aims). In other words, this normalization is necessary to avoid giving high scores to representations whose length scales vary between dimensions, but which are not really sparse.

By averaging Hoyer(z) over all representations, we can get the sparse score of a method. For the sparsest case, where each representation has a single activated dimension, the sparse score is 1. And when the representations get denser, $jj\hat{z}jj_2$ get smaller compared with $jj\hat{z}jj_1$, leading to smaller sparse scores.

Reproduction of Sparse-VAE We tried two different code bases for Sparse-VAE (Tonolini et al., 2020). The official code base[5] gives higher sparse scores for MNIST and FashionMNIST (though still lower than InteL-VAE), but is very unstable during training, with runs regularly failing after diverging and producing NaNs. This issue gets even more severe on CelebA which occurs after only a few training steps, undermining our ability to train anything meaningful at all. To account for this, we switched to the codebase[6] from De la Fuente & Aduviri (2019) that looked to replicate the results of the original paper. We report the results from this code base because it solves the instability issue and achieves reasonable results on CelebA. Interestingly, though its generation quality is good on MNIST and Fashion-MNIST, it fails to achieve a sparse score significantly higher than vanilla-VAE. As the original paper does not provide any quantitative evaluation of the achieved sparsity, it is difficult to know if this behavior is expected. We note though that the qualitative results shown in the paper appear to be substantially less sparse than those we show for the InteL-VAE, cf their Figure 5 compared to the top row of our Fig. 8. In particular, their representation seems to mostly 'switch off' some latents entirely, rather than having diversity between datapoints that is needed to score well under the Hoyer metric.

[5] https://github.com/ftonolini45/Variational_Sparse_Coding
[6] https://github.com/Alfo5123/Variational-Sparse-Coding

Encoder
Input 64 x 64 x 3
4x4 conv. 64 stride 2 & BN & LReLU
4x4 conv. 128 stride 2 & BN & LReLU
4x4 conv. 256 stride 2 & BN & LReLU
Dense (dim)

Decoder
Input dim
Dense (8x8x256) & BN & ReLU
4x4 upconv. 256 stride 2 & BN & ReLU
4x4 upconv. 128 stride 2 & BN & ReLU
4x4 upconv. 3 stride 2

Table C.2: Encoder and Decoder structures for CelebA, where dim is the latent dimension.

C.4 Additional Experiment Details

Datasets Both synthetic and real datasets are used in this paper. All synthetic datasets (sphere, square, star, and mixture of Gaussian) are generated by generators provided in our codes. For real datasets, We load MNIST, Fashion-MNIST, and CelebA directly from Tensorflow (Abadi et al., 2015), and we resize images from CelebA to 64x64 following Hou et al. (2017). For experiments with a specified number of training samples, we randomly select a subset of the training data. We use the same random seed for each model in the same experiment and different random seeds when repeating experiments.

Model structure For low-dimensional data, the encoder and decoder are both simple multilayer perceptrons with 3 hidden layers (10-10-10) and ReLU (Glorot et al., 2011) activation. For MNIST and Fashion-MNIST, we use the same encoder and decoder as Mathieu et al. (2019b). For CelebA, the structure of convolutional networks are shown in Table C.2.

Experiment settings Other hyperparameters are shown in Table C.1. All experiments are run on a GTX-1080-Ti GPU.