APPENDIX A PROOFS

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

$$D_{KL}\left(q_{\phi,\psi}(z|x) \parallel p_\psi(z)\right) \leq D_{KL}\left(q_\phi(y|x) \parallel \mathcal{N}(y; 0, I)\right). \tag{3}$$

If g_ψ 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

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

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 $\mathcal{Y}$ and $\mathcal{Z}$, and we have by construction a measurable map $g_\psi : (\mathcal{Y}, \mathcal{F}) \to (\mathcal{Z}, \mathcal{G})$. We can actually define the measurable space $(\mathcal{Z}, \mathcal{G})$ as the image of $(\mathcal{Y}, \mathcal{F})$ by g_ψ, then g_ψ is automatically both surjective and measurable.[2] We also assume that there exists a measure on $\mathcal{Y}$, which we denote ξ, and denote with ν the corresponding pushforward measure by g_ψ on $\mathcal{Z}$. We further have $\nu(A) = \xi(g_\psi^{-1}(A))$ for any $A \in \mathcal{G}$.[3]

We start by proving Eq. (3), where the Kullback-Leibler (KL) divergence between the two pushforward measures[4] $q_{\phi,\psi} \triangleq q_\phi \circ g_\psi^{-1}$ and $p_\psi \triangleq p \circ g_\psi^{-1}$ is upper bounded by $D_{\mathrm{KL}}\left(q_\phi(y|x) \parallel p(y)\right)$, 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 $D_{\mathrm{KL}}\left(q_\phi(y|x) \parallel \mathcal{N}(y; 0, I)\right)$ is finite, as otherwise the result is trivially true, which in turn implies $q_\phi \ll p$.[5] For any $A \in \mathcal{G}$, we have that if $p_\psi(A) = p \circ g_\psi^{-1}(A) = p(g_\psi^{-1}(A)) = 0$ then this implies $q_\phi(g_\psi^{-1}(A)) = q_\phi \circ g_\psi^{-1}(A) = q_{\phi,\psi}(A) = 0$. As such, we have that $q_{\phi,\psi} \ll p_\psi$ and so the $D_{\mathrm{KL}}\left(q_{\phi,\psi}(z|x) \parallel p_\psi(z)\right)$ is also defined.

Our next significant step is to show that

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

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

$$\begin{aligned}
\int_\mathcal{Z} \mathbb{1}_A \, \frac{q_\phi \circ g_\psi^{-1}}{p \circ g_\psi^{-1}} \, p \circ g_\psi^{-1} \, d\nu &= \int_\mathcal{Z} \mathbb{1}_A \, q_\phi \circ g_\psi^{-1} \, d\nu = \int_\mathcal{Z} \mathbb{1}_A \, d(q_\phi \circ g_\psi^{-1}) \\
&\overset{(a)}{=} \int_\mathcal{Y} (\mathbb{1}_A \circ g_\psi) \, dq_\phi = \int_\mathcal{Y} (\mathbb{1}_A \circ g_\psi) \, q_\phi \, d\xi \\
&\overset{(b)}{=} \int_\mathcal{Y} (\mathbb{1}_A \circ g_\psi) \, \frac{q_\phi}{p} \, p \, d\xi \\
&\overset{(c)}{=} \int_\mathcal{Y} (\mathbb{1}_A \circ g_\psi) \, \mathbb{E}_{p(y)}\left[\frac{q_\phi}{p} \,\middle|\, \sigma(g_\psi)\right] \, p \, d\xi \\
&\overset{(d)}{=} \int_\mathcal{Y} (\mathbb{1}_A \circ g_\psi) \, (h \circ g_\psi) \, p \, d\xi = \int_\mathcal{Y} (\mathbb{1}_A \circ g_\psi) \, (h \circ g_\psi) \, dp \\
&\overset{(e)}{=} \int_\mathcal{Z} \mathbb{1}_A \, h \, d(p \circ g_\psi^{-1}) = \int_\mathcal{Z} \mathbb{1}_A \, h \, (p \circ g_\psi^{-1}) \, d\nu,
\end{aligned}$$

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

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

[4] We denote the pushforward of a probability measure χ along a map g by $\chi \circ g^{-1}$.

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

where we have leveraged the definition of pushforward measures in (a & e); the absolute continuity of q_ϕ 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 $q_\phi \circ g_\psi^{-1} / p \circ g_\psi^{-1} = h$, almost-surely with respect to $q_\phi \circ g_\psi^{-1}$ and thus that Eq. (A.1) is verified.

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

$$
\begin{aligned}
D_{\mathrm{KL}}\left(q_{\phi,\psi}(z|x) \parallel p_\psi(z)\right) &\overset{(a)}{=} \int_{\mathcal{Z}} \log\left(\frac{q_{\phi,\psi}}{p_\psi}\right) q_{\phi,\psi}\, d\nu \\
&\overset{(b)}{=} \int_{\mathcal{Z}} \log\left(\frac{q_{\phi,\psi}}{p_\psi}\right) \frac{q_{\phi,\psi}}{p_\psi}\, p_\psi\, d\nu \\
&\overset{(c)}{=} \int_{\mathcal{Z}} f\left(\frac{q_{\phi,\psi}}{p_\psi}\right) p_\psi\, d\nu = \int_{\mathcal{Z}} f\left(\frac{q_{\phi,\psi}}{p_\psi}\right) d(p \circ g_\psi^{-1}) \\
&\overset{(d)}{=} \int_{\mathcal{Y}} f\left(\frac{q_{\phi,\psi}}{p_\psi} \circ g_\psi\right) dp = \int_{\mathcal{Y}} f\left(\frac{q_\phi \circ g_\psi^{-1}}{p \circ g_\psi^{-1}} \circ g_\psi\right) p\, d\xi \\
&\overset{(e)}{=} \int_{\mathcal{Y}} f\left(\mathbb{E}_{p(y)}\left[\frac{q_\phi}{p} \,\middle|\, \sigma(g_\psi)\right]\right) p\, d\xi \\
&\overset{(f)}{\leq} \int_{\mathcal{Y}} \mathbb{E}_{p(y)}\left[f\left(\frac{q_\phi}{p}\right) \,\middle|\, \sigma(g_\psi)\right] p\, d\xi \\
&\overset{(g)}{=} \int_{\mathcal{Y}} f\left(\frac{q_\phi}{p}\right) p\, d\xi \\
&\overset{(h)}{=} \int_{\mathcal{Y}} \log\left(\frac{q_\phi}{p}\right) \frac{q_\phi}{p}\, p\, d\xi \\
&\overset{(i)}{=} \mathbb{E}_{q_\phi(y|x)}\left[\log\left(\frac{q_\phi(y|x)}{p(y)}\right)\right] \\
&\overset{(j)}{=} D_{\mathrm{KL}}\left(q_\phi(y|x) \parallel p(y)\right),
\end{aligned}
$$

where we leveraged the definition of the KL divergence in (a & j); the absolute continuity of q_ϕ 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_ψ is invertible. As f is strictly convex, this happens if and only if $\frac{q_\phi}{p} = \mathbb{E}_{p(y)}\left[\frac{q_\phi}{p} \,\middle|\, \sigma(g_\psi)\right]$. A sufficient condition for this to be true is for $\frac{q_\phi}{p}$ to be measurable w.r.t. $\sigma(g_\psi)$ which is satisfied when $g_\psi : \mathcal{Y} \mapsto \mathcal{Z}$ is invertible as $\sigma(g_\psi) \supseteq \mathcal{F}$, as required. We have thus shown that the KL divergences are equal when using an invertible g_ψ.

For the reconstruction term, we instead have

$$
\begin{aligned}
\mathbb{E}_{q_\phi(y|x)}[\log p_\theta(x|g_\psi(y))] &= \int_{\mathcal{Y}} \log p_\theta(x|g_\psi(y)) q_\phi(y|x) d\xi \\
&= \int_{\mathcal{Z}} \log p_\theta(x|z) q_{\phi,\psi}(z|x) d\nu \\
&= \mathbb{E}_{q_{\phi,\psi}(z|x)}[\log p_\theta(x|z)].
\end{aligned}
$$

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.

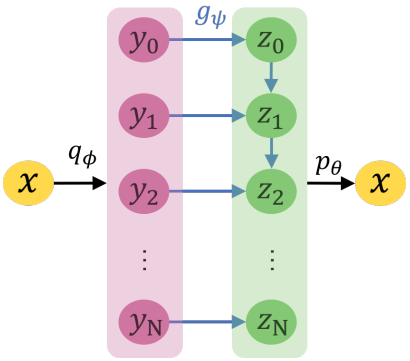


Figure B.1: Graphical model for hierarchical InteL-VAE

Mapping design Following existing ideas from hierarchical VAEs (Sønderby et al., 2016; Zhao et al., 2017), we propose a hierarchical mapping g_ψ. 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 $z = g_\psi(y)$ 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_ψ 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.

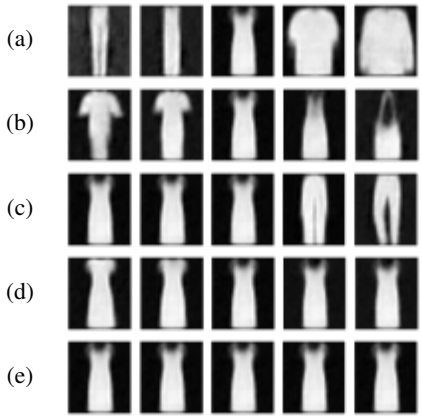


Figure B.2: Manipulating representations of a hierarchical InteL-VAE. The features are split into 5 levels, with each of (a) [highest] to (e) [lowest] corresponding to an example feature from each. We see that high-level features control more complex properties, such as class label or topological structure,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

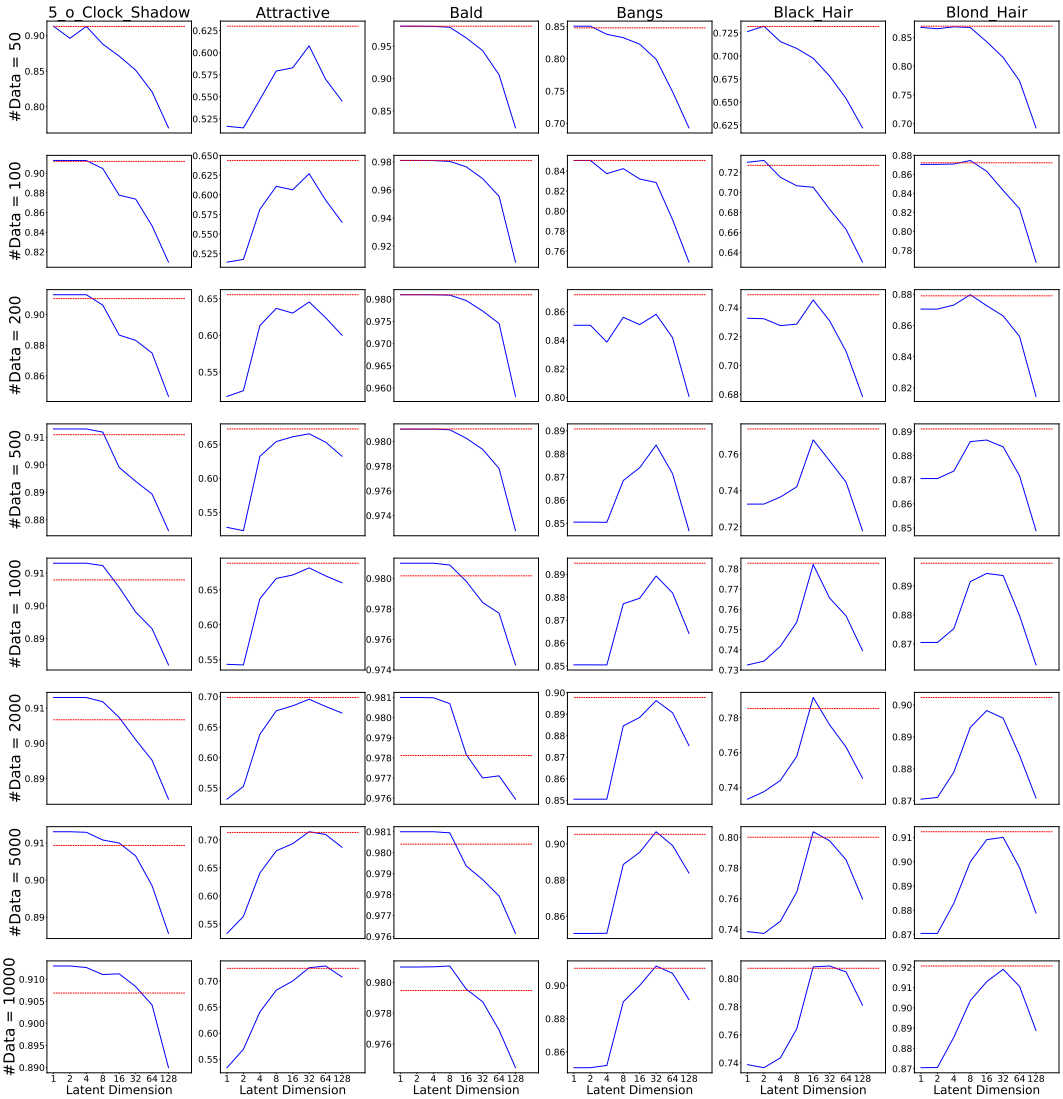


Figure B.3: InteL-VAE's performance of attribute prediction on **CelebA** dataset. Each column shows results on the same feature with different data sizes and each column shows results on different features. In each graph, test accuracy of vanilla-VAE with different latent dimensions are shown in blue 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 As described in the main text, when using the inductive bias of a single hole, we use the following g_ψ to map a Gaussian distribution approximately to a circular distribution,

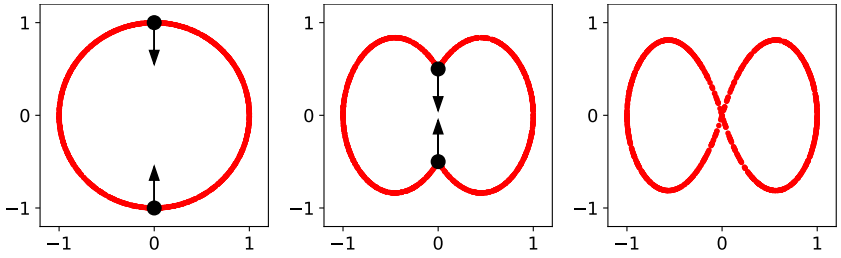
$$g_1(y) = \frac{y}{||y||_2 + \epsilon}. \tag{C.1}$$

As a result, $p_\psi(z) = g_\psi(p(y))$ is approximately the uniform distribution on S^1.

To introduce an additional hole, we simply glue two points on S^1 together to make more holes. For example

$$g_2(y) = \text{Concat}\left(g_1(y)_{[:,1]},\ g_1(y)_{[:,2]}\sqrt{(4/3 - (1 - |g_1(y)_{[:,1]}|)^2)} - \frac{1}{\sqrt{3}}\right), \tag{C.2}$$

which first map y to approximately S^1, and then glues $(0, 1)$ and $(0, -1)$ 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.



(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 For simplicity's sake let us temporarily assume that the dimension of $\mathcal{Y}$ 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. C.2b. For any point y, we can get its component (sector) index $\text{ci}(y)$ as well as its distance from the sector boundary $\text{dis}(y)$. By further defining the radius direction for the k-th sector (cf Fig. C.2c) as

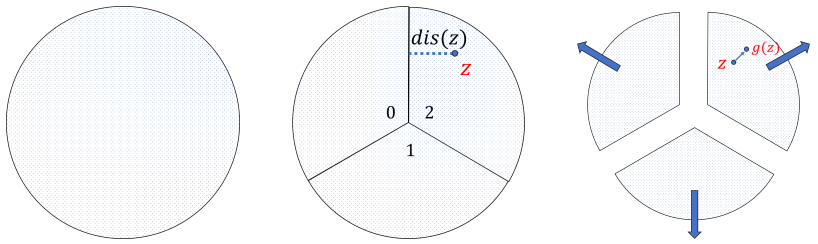
$$\Delta(k) = \left(\cos\left(\frac{2\pi}{K}\left(k + \frac{1}{2}\right)\right), \sin\left(\frac{2\pi}{K}\left(k + \frac{1}{2}\right)\right)\right) \quad \forall k \in \{1, \ldots, K\},$$

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

$$g(y) = y + c_1 \text{dis}(y)^{c_2} \Delta(\text{ci}(y)), \tag{C.3}$$

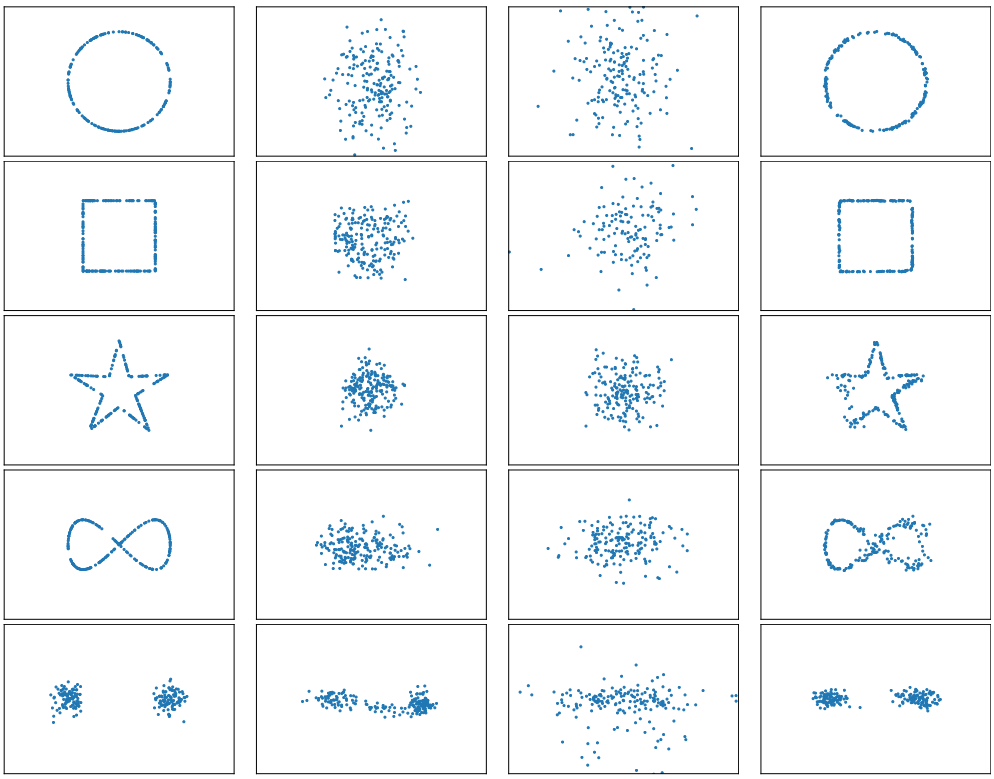
where c_1 and c_2 are constants, which are set to 5 and 0.2 in our experiments. We can see that although g has very different function on different sectors, it is still continuous on the whole plane if we extend g s.t. $g(y) = y$ on sector boundaries, which is desirable for gradient-based training.

When dimension of $\mathcal{Y}$ 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 $\mathcal{Y}_{2i}$ and $\mathcal{Y}_{2i+1}$ into K_i sectors by the Eq. (C.3). As a result, $\mathcal{Y}$ is split into $K = \prod_i k_i$ clusters. When $K = 2$, we find that g only changes the 1-st dimension of $\mathcal{Y}$, so it can be applied to cases where latent dimension is 1.



(a) Original Distribution (b) Split into K sectors (c) Move sectors away from origin

Figure C.2: Illustration of clustered mapping. The circle represents a density isoline of Gaussian distribution. 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 as per Eq. (C.3).



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

Figure C.3: Extension of Fig. 4 showing Vamp-VAE baseline and additional circular target distribution (top row, uses the same single hole g_ψ as the second and third rows).

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 $2\pi\text{Softmax}(u)_i$. 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

Sparsity regularizer Our sparsity regularizer term, $\mathcal{L}_{sp}$, is used to encourage our dimensionality selector network (DS) to produce sparse mappings. It is defined using a mini-batch of samples $\{y_i\}_{i=1}^M$ drawn during training and is given by

$$\mathcal{L}_{sp} = \mathbb{E}\left[\frac{1}{M}\sum_{i=1}^{M}(H\left(DS(y_i)\right)) - H\left(\frac{1}{M}\sum_{i=1}^{M}DS(y_i)\right)\right], \tag{C.4}$$

where $H(v) = -\sum_i \log\left(v_i/\|v\|_1\right)$ is normalized entropy of an non-negative vector v, and the expectation is taken over the process of drawing a mini-batch of datapoints $\{x_i\}_{i=1}^M$ and then independently drawing an intermediary latent for each, $y_i \sim q_\phi(y|x = x_i)$. During training, the first term 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. The sparsity-regularized training objective now becomes

$$\mathcal{L}_{\mathcal{Y}}(\theta, \phi, \psi, \gamma) = \mathcal{L}_{\mathcal{Y}}(\theta, \phi, \psi) + \gamma\ \mathcal{L}_{sp}, \tag{C.5}$$

where γ is a hyper-parameter controlling the degree of sparsity enforced (with higher γ corresponding to more sparsity).

We note that $\mathcal{L}_{sp}$ alone is not expected to induce sparsity without also using the carefully constructed g_ψ 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 $\gamma > 30.0$ we can only slightly increase the sparsity score ($0.230 \rightarrow 0.235$). 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 $z \in \mathbb{R}^D$,

$$\text{Hoyer}(z) = \frac{\sqrt{D} - ||\hat{z}||_1/||\hat{z}||_2}{\sqrt{D} - 1}. \tag{C.6}$$

Here, following Mathieu et al. (2019b), we crucially first normalized each dimension d of z to have standard deviation 1, $\hat{z}_d = z_d/\sigma_d$, to ensure that we only measure sparsity that varies between data 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, $||\hat{z}||_2$ get smaller compared with $||\hat{z}||_1$, 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[6] 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 issues occurring after only a few training steps, undermining our ability to train anything meaningful at all. To account for this, we switched to the code base[7] 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

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

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.

Encoder	**Decoder**
Input 64 x 64 x 3	Input dim
4x4 conv. 64 stride 2 & BN & LReLU	Dense (8x8x256) & BN & ReLU
4x4 conv. 128 stride 2 & BN & LReLU	4x4 upconv. 256 stride 2 & BN & ReLU
4x4 conv. 256 stride 2 & BN & LReLU	4x4 upconv. 128 stride 2 & BN & ReLU
Dense (dim)	4x4 upconv. 3 stride 2

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

compared to the top row of our Fig. 6. 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.

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.