Appendix for Disentangling Disentanglement in Variational Autoencoders

Emile Mathieu*, Tom Rainforth*, N. Siddharth*, Yee Whye Teh

A. Proofs for Disentangling the β-VAE

Theorem 1. *The β-VAE target $\mathcal{L}_\beta(\boldsymbol{x})$ can be interpreted in terms of the standard ELBO, $\mathcal{L}\left(\boldsymbol{x};\pi_{\theta,\beta},q_\phi\right)$, for an adjusted target $\pi_{\theta,\beta}(\boldsymbol{x},\boldsymbol{z}) \triangleq p_\theta(\boldsymbol{x}\mid\boldsymbol{z})f_\beta(\boldsymbol{z})$ with annealed prior $f_\beta(\boldsymbol{z}) \triangleq p(\boldsymbol{z})^\beta/F_\beta$ as*

$$\mathcal{L}_\beta(\boldsymbol{x}) = \mathcal{L}\left(\boldsymbol{x};\pi_{\theta,\beta},q_\phi\right) + (\beta-1)H_{q_\phi} + \log F_\beta \tag{3}$$

where $F_\beta \triangleq \int_{\boldsymbol{z}} p(\boldsymbol{z})^\beta d\boldsymbol{z}$ is constant given β, and H_{q_ϕ} is the entropy of $q_\phi(\boldsymbol{z}\mid\boldsymbol{x})$.

Proof. Starting with (2), we have

$$\begin{aligned}
\mathcal{L}_\beta(\boldsymbol{x}) =& \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}\mid\boldsymbol{z})] + \beta H_{q_\phi} \\
&+ \beta\,\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{z})] \\
=& \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}\mid\boldsymbol{z})] + (\beta-1)H_{q_\phi} + H_{q_\phi} \\
&+ \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\Big[\log p(\boldsymbol{z})^\beta - \log F_\beta\Big] + \log F_\beta \\
=& \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}\mid\boldsymbol{z})] + (\beta-1)H_{q_\phi} \\
&- \mathrm{KL}(q_\phi(\boldsymbol{z}\mid\boldsymbol{x})\,\|\,f_\beta(\boldsymbol{z})) + \log F_\beta \\
=& \mathcal{L}\left(\boldsymbol{x};\pi_{\theta,\beta},q_\phi\right) + (\beta-1)H_{q_\phi} + \log F_\beta
\end{aligned}$$

as required. □

Corollary 1. *If $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z};0,\Sigma)$ and $q_\phi(\boldsymbol{z}\mid\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z};\mu_\phi(\boldsymbol{x}),S_\phi(\boldsymbol{x})\right)$, then,*

$$\mathcal{L}_\beta(\boldsymbol{x};\theta,\phi) = \mathcal{L}\left(\boldsymbol{x};\theta',\phi'\right) + \frac{(\beta-1)}{2}\log|S_{\phi'}(\boldsymbol{x})| + c \tag{4}$$

where θ' and ϕ' represent rescaled networks such that

$$\begin{aligned}
p_{\theta'}(\boldsymbol{x}\mid\boldsymbol{z}) &= p_\theta\Big(\boldsymbol{x}\mid\boldsymbol{z}/\sqrt{\beta}\Big), \\
q_{\phi'}(\boldsymbol{z}|\boldsymbol{x}) &= \mathcal{N}\left(\boldsymbol{z};\mu_{\phi'}(\boldsymbol{x}),S_{\phi'}(\boldsymbol{x})\right), \\
\mu_{\phi'}(\boldsymbol{x}) &= \sqrt{\beta}\mu_\phi(\boldsymbol{x}), \quad S_{\phi'}(\boldsymbol{x}) = \beta S_\phi(\boldsymbol{x}),
\end{aligned}$$

and $c \triangleq \frac{D(\beta-1)}{2}\left(1+\log\frac{2\pi}{\beta}\right) + \log F_\beta$ is a constant, with D denoting the dimensionality of $\boldsymbol{z}$.

Proof. We start by noting that

$$\begin{aligned}
\pi_{\theta,\beta}(\boldsymbol{x}) &= \mathbb{E}_{f_\beta(\boldsymbol{z})}[p_\theta(\boldsymbol{x}\mid\boldsymbol{z})] = \mathbb{E}_{p(\boldsymbol{z})}\Big[p_\theta\Big(\boldsymbol{x}\mid\boldsymbol{z}/\sqrt{\beta}\Big)\Big] \\
&= \mathbb{E}_{p(\boldsymbol{z})}[p_{\theta'}(\boldsymbol{x}\mid\boldsymbol{z})] = p_{\theta'}(\boldsymbol{x})
\end{aligned}$$

Now considering an alternate form of $\mathcal{L}\left(\boldsymbol{x};\pi_{\theta,\beta},q_\phi\right)$ in (3),

$$\begin{aligned}
&\mathcal{L}\left(\boldsymbol{x};\pi_{\theta,\beta},q_\phi\right) \\
&= \log\pi_{\theta,\beta}(\boldsymbol{x}) - \mathrm{KL}(q_\phi(\boldsymbol{z}\mid\boldsymbol{x})\,\|\,\pi_{\theta,\beta}(\boldsymbol{z}\mid\boldsymbol{x})) \\
&= \log p_{\theta'}(\boldsymbol{x}) - \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left(\frac{q_\phi(\boldsymbol{z}\mid\boldsymbol{x})p_{\theta'}(\boldsymbol{x})}{p_\theta(\boldsymbol{x}\mid\boldsymbol{z})f_\beta(\boldsymbol{z})}\right)\right] \\
&= \log p_{\theta'}(\boldsymbol{x}) \\
&\quad - \mathbb{E}_{q_{\phi'}(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left(\frac{q_\phi\big(\boldsymbol{z}/\sqrt{\beta}\mid\boldsymbol{x}\big)p_{\theta'}(\boldsymbol{x})}{p_\theta\big(\boldsymbol{x}\mid\boldsymbol{z}/\sqrt{\beta}\big)f_\beta\big(\boldsymbol{z}/\sqrt{\beta}\big)}\right)\right].
\end{aligned} \tag{8}$$

We first simplify $f_\beta(\boldsymbol{z}/\sqrt{\beta})$ as

$$\begin{aligned}
f_\beta(\boldsymbol{z}/\sqrt{\beta}) &= \frac{1}{\sqrt{2\pi|\Sigma/\beta|}}\exp\left(-\frac{1}{2}\boldsymbol{z}^T\Sigma^{-1}\boldsymbol{z}\right) \\
&= p(\boldsymbol{z})\beta^{(D/2)}.
\end{aligned}$$

Further, denoting $\boldsymbol{z}_\dagger = \boldsymbol{z} - \sqrt{\beta}\mu_{\phi'}(\boldsymbol{x})$, and $\boldsymbol{z}_\ddagger = \boldsymbol{z}_\dagger/\sqrt{\beta} = \boldsymbol{z}/\sqrt{\beta} - \mu_{\phi'}(\boldsymbol{x})$, we have

$$\begin{aligned}
q_{\phi'}(\boldsymbol{z}\mid\boldsymbol{x}) &= \frac{1}{\sqrt{2\pi|S_\phi(\boldsymbol{x})\beta|}}\exp\left(-\frac{1}{2\beta}\boldsymbol{z}_\dagger^T S_\phi(\boldsymbol{x})^{-1}\boldsymbol{z}_\dagger\right), \\
q_\phi\left(\frac{\boldsymbol{z}}{\sqrt{\beta}}\mid\boldsymbol{x}\right) &= \frac{1}{\sqrt{2\pi|S_\phi(\boldsymbol{x})|}}\exp\left(-\frac{1}{2}\boldsymbol{z}_\ddagger^T S_\phi(\boldsymbol{x})^{-1}\boldsymbol{z}_\ddagger\right) \\
\text{giving}\quad q_\phi\Big(\boldsymbol{z}/\sqrt{\beta}\mid\boldsymbol{x}\Big) &= q_{\phi'}(\boldsymbol{z}\mid\boldsymbol{x})\beta^{(D/2)}.
\end{aligned}$$

Plugging these back in to (8) while remembering $p_\theta(\boldsymbol{x}\mid\boldsymbol{z}/\sqrt{\beta}) = p_{\theta'}(\boldsymbol{x}\mid\boldsymbol{z})$, we have

$$\begin{aligned}
&\mathcal{L}\left(\boldsymbol{x};\pi_{\theta,\beta},q_\phi\right) \\
&= \log p_{\theta'}(\boldsymbol{x}) - \mathbb{E}_{q_{\phi'}(\boldsymbol{z}|\boldsymbol{x})}\left[\log\left(\frac{q_{\phi'}(\boldsymbol{z}\mid\boldsymbol{x})p_{\theta'}(\boldsymbol{x})}{p_{\theta'}(\boldsymbol{x}\mid\boldsymbol{z})p(\boldsymbol{z})}\right)\right] \\
&= \mathcal{L}(\boldsymbol{x};\theta,\phi),
\end{aligned}$$

showing that the ELBOs for the two setups are the same. For the entropy term, we note that

$$\begin{aligned}
H_{q_\phi} &= \frac{D}{2}\left(1+\log 2\pi\right) + \frac{1}{2}\log|S_\phi(\boldsymbol{x})| \\
&= \frac{D}{2}\left(1+\log\frac{2\pi}{\beta}\right) + \frac{1}{2}\log|S_{\phi'}(\boldsymbol{x})|.
\end{aligned}$$

Finally substituting for H_{q_ϕ} and $\mathcal{L}\left(\boldsymbol{x};\pi_{\theta,\beta},q_\phi\right)$ in (3) gives the desired result. □

Corollary 2. *Let $[\theta',\phi'] = g_\beta([\theta,\phi])$ represent the transformation required to produced the rescaled networks in Corollary 1. If $0 < |\det\nabla_{\theta,\phi}g([\theta,\phi])| < \infty\ \forall[\theta,\phi]$, then*

$$\nabla_{\theta,\phi}\mathcal{L}_\beta(\boldsymbol{x};\theta,\phi) = \mathbf{0} \;\Leftrightarrow\; \nabla_{\theta',\phi'}\mathcal{L}_{H,\beta}\left(\boldsymbol{x};\theta',\phi'\right) = \mathbf{0}.$$

Thus $[\theta^*,\phi^*]$ being a stationary point of $\frac{1}{n}\sum_{i=1}^n\mathcal{L}_\beta(\boldsymbol{x}_i;\theta,\phi)$ indicates that $g_\beta([\theta^*,\phi^*])$ is a stationary point of $\frac{1}{n}\sum_{i=1}^n\mathcal{L}_{H,\beta}\left(\boldsymbol{x}_i;\theta',\phi'\right)$ and vice-versa.

Proof. Starting from Corollary 1 we have that

$$\begin{aligned}
\nabla_{\theta,\phi}\mathcal{L}_\beta(\boldsymbol{x};\theta,\phi) &= \nabla_{\theta,\phi}\mathcal{L}_{H,\beta}\left(\boldsymbol{x};\theta',\phi'\right) \\
&= \left(\nabla_{\theta,\phi}g_\beta([\theta,\phi])\right)\nabla_{\theta',\phi'}\mathcal{L}_{H,\beta}\left(\boldsymbol{x};\theta',\phi'\right),
\end{aligned}$$

so $\nabla_{\theta',\phi'}\mathcal{L}_{H,\beta}(\boldsymbol{x};\theta',\phi')=\mathbf{0}\Longrightarrow\nabla_{\theta,\phi}\mathcal{L}_{\beta}(\boldsymbol{x};\theta,\phi)=\mathbf{0}$ given our assumption that $|\det\nabla_{\theta,\phi}g([\theta,\phi])|<\infty\,\forall[\theta,\phi]$. Further, as $0<|\det\nabla_{\theta,\phi}g([\theta,\phi])|\;\forall[\theta,\phi]$, $(\nabla_{\theta,\phi}g_{\beta}([\theta,\phi]))^{-1}$ exists and has a finite determinant, so $\nabla_{\theta,\phi}\mathcal{L}_{\beta}(\boldsymbol{x};\theta,\phi)=\mathbf{0}$ also implies $\nabla_{\theta',\phi'}\mathcal{L}_{H,\beta}\left(\boldsymbol{x};\theta',\phi'\right)=\mathbf{0}$. □

Theorem 2. *If* $p(\boldsymbol{z})=\mathcal{N}(\boldsymbol{z};0,\sigma I)$ *and* $q_\phi(\boldsymbol{z}\mid\boldsymbol{x})=\mathcal{N}\left(\boldsymbol{z};\mu_\phi(\boldsymbol{x}),S_\phi(\boldsymbol{x})\right)$*, then for all rotation matrices* R*,*

$$\mathcal{L}_{\beta}(\boldsymbol{x};\theta,\phi)=\mathcal{L}_{\beta}(\boldsymbol{x};\theta^{\dagger}(R),\phi^{\dagger}(R)) \tag{6}$$

where $\theta^{\dagger}(R)$ *and* $\phi^{\dagger}(R)$ *are transformed networks such that*

$$p_{\theta^{\dagger}}(\boldsymbol{x}\mid\boldsymbol{z})=p_{\theta}\big(\boldsymbol{x}\mid R^{T}\boldsymbol{z}\big),$$
$$q_{\phi^{\dagger}}(\boldsymbol{z}|\boldsymbol{x})=\mathcal{N}\left(\boldsymbol{z};R\mu_{\phi}(\boldsymbol{x}),RS_{\phi}(\boldsymbol{x})R^{T}\right).$$

Proof. If $\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $\boldsymbol{y}=R\boldsymbol{z}$ then, by Petersen et al. (§8.1.4 2008)), we have

$$\boldsymbol{y}\sim\mathcal{N}(\boldsymbol{y};R\mu_{\phi}(\boldsymbol{x}),RS_{\phi}(\boldsymbol{x})R^{T}).$$

Consequently, the changes made by the transformed networks cancel to give the same reconstruction error as

$$\begin{aligned}\mathbb{E}_{q_{\phi}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\theta}(\boldsymbol{x}\mid\boldsymbol{z})]&=\mathbb{E}_{q_{\phi^{\dagger}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\theta}(\boldsymbol{x}\mid R^{T}\boldsymbol{z})]\\&=\mathbb{E}_{q_{\phi^{\dagger}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\theta^{\dagger}}(\boldsymbol{x}\mid\boldsymbol{z})].\end{aligned}$$

Furthermore, the KL divergence between $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p_\theta(\boldsymbol{z})$ is invariant to rotation, because of the rotational symmetry of the latter, such that $\mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})\,\|\,p(\boldsymbol{z}))=\mathrm{KL}\big(q_{\phi^{\dagger}}(\boldsymbol{z}|\boldsymbol{x})\,\|\,p(\boldsymbol{z})\big)$. The result now follows from noting that the two terms of the β-VAE are equal under rotation. □

B. Experimental Details

Disentanglement - 2d-shapes: The experiments from Section 6 on the impact of the prior in terms disentanglement are conducted on the **2D Shapes** (Matthey et al., 2017) dataset, comprising of 737,280 binary 64 x 64 images of 2D shapes with ground truth factors [number of values]: shape [3], scale [6], orientation [40], x-position [32], y-position [32]. We use a convolutional neural network for the encoder and a deconvolutional neural network for the decoder, whose architectures are described in Table 1a. We use $[0,1]$ normalised data as targets for the mean of a Bernoulli distribution and negative cross-entropy for $\log p(\boldsymbol{x}|\boldsymbol{z})$. We rely on the Adam optimiser (Kingma and Ba, 2015; Reddi et al., 2018) with learning rate $1e^{-4}$, β_1 = 0.9, and β_2 = 0.999, to optimise the β-VAE objective from (3).

For $p(\boldsymbol{z})=\mathcal{N}(\boldsymbol{z};\mathbf{0},\mathrm{diag}(\sigma))$, experiments were run with a batch size of 64 and for 20 epochs. For $p(\boldsymbol{z})=\prod_d\textsc{Student-t}(z_d;\nu)$, experiments were run with a batch size of 256 and for 40 epochs. In Figure 2, the *PCA initialised anisotropic* prior is initialised so that its standard deviations are set to be the first D singular values of the data. These are then mapped through a softmax function to ensure that the β regularisation coefficient is not implicitly scaled compared to the isotropic case. For the *learned anisotropic* priors, standard deviations are first initialised as just described, and then learned along with the model through a log-variance parametrisation.

Encoder	**Decoder**
Input 64 x 64 binary image	Input $\in\mathbb{R}^{10}$
4x4 conv. 32 stride 2 & ReLU	FC. 128 ReLU
4x4 conv. 32 stride 2 & ReLU	FC. 4x4 x 64 ReLU
4x4 conv. 64 stride 2 & ReLU	4x4 upconv. 64 stride 2 & ReLU
4x4 conv. 64 stride 2 & ReLU	4x4 upconv. 64 stride 2 & ReLU
FC. 128	4x4 upconv. 32 stride 2 & ReLU
FC. 2x10	4x4 upconv. 1. stride 2

(a) 2D-shapes dataset.

Encoder	**Decoder**
Input $\in\mathbb{R}^2$	Input $\in\mathbb{R}^2$
FC. 100. & ReLU	FC. 100 & ReLU
FC. 2x2	FC. 2x2

(b) Pinwheel dataset.

Encoder
Input 32 x 32 x 1 channel image
4x4 conv. 32 stride 2 & BatchNorm2d & LeakyReLU(.2)
4x4 conv. 64 stride 2 & BatchNorm2d & LeakyReLU(.2)
4x4 conv. 128 stride 2 & BatchNorm2d & LeakyReLU(.2)
4x4 conv. 50, 4x4 conv. 50
Decoder
Input $\in\mathbb{R}^{50}$
4x4 upconv. 128 stride 1 pad 0 & BatchNorm2d & ReLU
4x4 upconv. 64 stride 2 pad 1 & BatchNorm2d & ReLU
4x4 upconv. 32 stride 2 pad 1 & BatchNorm2d & ReLU
4x4 upconv. 1 stride 2 pad 1

(c) Fashion-MNIST dataset.

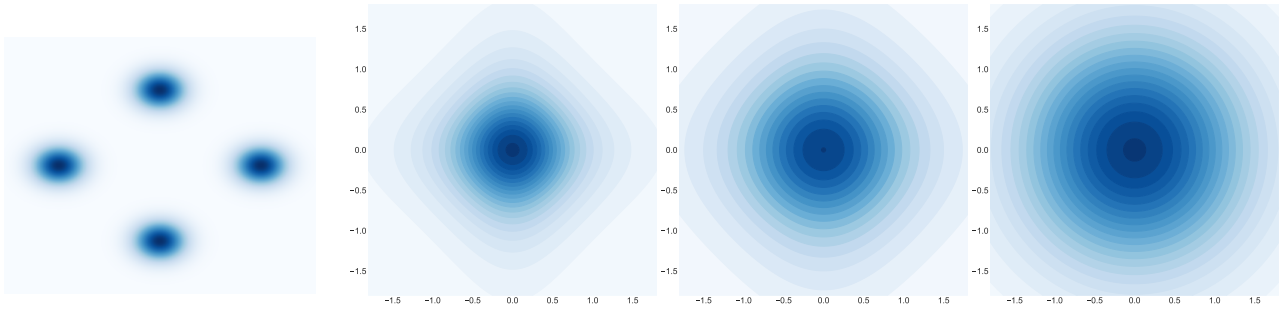
Table 1. Encoder and decoder architectures.

We rely on the metric presented in §4 and Appendix B of Kim and Mnih (2018) as a measure of axis-alignment of the latent encodings with respect to the true (known) generative factors. Confidence intervals in Figure 2 were computed via the assumption of normally distributed samples with unknown mean and variance, with 100 runs of each model.

Clustering - Pinwheel We generated spiral cluster data[1], with $n=400$ observations, clustered in 4 spirals, with radial and tangential standard deviations respectively of 0.1 and 0.30, and a rate of 0.25. We use fully-connected neural networks for both the encoder and decoder, whose architectures are described in Table 1b. We minimise the objective from (7), with $\mathbb{D}$ chosen to be the inclusive KL and $q_\phi(\boldsymbol{z})$ approximated by the aggregate encoding of the full dataset:

$$\begin{aligned}\mathbb{D}\left(q_{\phi}(\boldsymbol{z}),p(\boldsymbol{z})\right)&=\mathrm{KL}\left(p(\boldsymbol{z})||q_{\phi}(\boldsymbol{z})\right)\\&=\mathbb{E}_{p(\boldsymbol{z})}\left[\log(p(\boldsymbol{z}))-\log\left(\mathbb{E}_{p_{\mathcal{D}}(\boldsymbol{x})}[q_{\phi}(\boldsymbol{z}\mid\boldsymbol{x})]\right)\right]\end{aligned}$$

[1] http://hips.seas.harvard.edu/content/synthetic-pinwheel-data-matlab.



(a) MoG (b) Student-t

Figure 6. (a) PDF of Gaussian mixture model prior $p(\boldsymbol{z})$, as per (9). (b) PDF for a 2-dimensional factored Student-t distributions p_ν with degree of freedom $\nu = \{3, 5, 100\}$ (left to right). Note that $p_\nu(\boldsymbol{z}) \to \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I})$ as $\nu \to \infty$.

$$\approx \sum_{j=1}^{B} \left(\log p(\boldsymbol{z}_j) - \log \left(\sum_{i=1}^{n} q_\phi(\boldsymbol{z}_j \mid \boldsymbol{x}_i) \right) \right)$$

with $\boldsymbol{z}_j \sim p(\boldsymbol{z})$. A Gaussian likelihood is used for the encoder. We trained the model for 500 epochs using the Adam optimiser (Kingma and Ba, 2015; Reddi et al., 2018), with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a learning rate of $1e^{-3}$. The batch size is set to $B = n$.

The Gaussian mixture prior (c.f. Figure 6(a)) is defined as

$$\begin{aligned} p(\boldsymbol{z}) &= \sum_{c=1}^{C} \pi^c \, \mathcal{N}(\boldsymbol{z} | \boldsymbol{\mu}^c, \boldsymbol{\Sigma}^c) \\ &= \sum_{c=1}^{C} \pi^c \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{z}_d | \mu_d^c, \sigma_d^c) \end{aligned} \tag{9}$$

with $D = 2$, $C = 4$, $\boldsymbol{\Sigma}^c = 0.03 I_D$, $\pi^c = 1/C$, and $\mu_d^c \in \{0, 1\}$.

Sparsity - Fashion-MNIST The experiments from Section 6 on the latent representation's sparsity are conducted on the **Fashion-MNIST** (Xiao et al., 2017) dataset, comprising of $70,000$ greyscale images resized to 32×32.

To enforce sparsity, we relied on a prior defined as a factored univariate mixture of a standard and low variance normal distributions:

$$p(\boldsymbol{z}) = \prod\nolimits_d \, (1 - \gamma) \, \mathcal{N}(z_d; 0, 1) + \gamma \, \mathcal{N}(z_d; 0, \sigma_0^2)$$

with $\sigma_0^2 = 0.05$. The weight, γ, of the low-variance component indicates how likely samples are to come from that component, hence to be *off*.

We minimised the objective from (7), with $\mathbb{D}(q_\phi(\boldsymbol{z}), p(\boldsymbol{z}))$ taken to be a dimension-wise MMD with a sum of *Cauchy* kernels on each dimension. Equivalently, we can think of this as calculating a single MMD using the single kernel

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{d=1}^{D} \sum_{\ell}^{L} \frac{\sigma_\ell}{\sigma_{\ell=1} + (x_d - y_d)^2}. \tag{10}$$

where $\sigma_\ell \in \{0.2, 0.4, 1, 2, 4, 10\}$ are a set of length scales.

This dimension-wise kernel only enforces a congruence between the marginal distributions of $\boldsymbol{x}$ and $\boldsymbol{y}$ and so, strictly speaking, its MMD does not constitute a valid divergence metric in the sense that we can have $\mathbb{D}(q_\phi(\boldsymbol{z}), p(\boldsymbol{z})) = 0$ when $q_\phi(\boldsymbol{z})$ and $p(\boldsymbol{z})$ are not identical distributions: it only requires their marginals to match to get zero divergence.

The reasons we chose this approach are twofold. Firstly, we found that conventional kernels based on the Euclidean distance between encodings produced gradients with insurmountably high variances, meaning that effectively minimizing the divergence to get $q_\phi(\boldsymbol{z})$ and $p(\boldsymbol{z})$ to match was not possible, even for very large batch sizes and $\alpha \to \infty$.

Secondly, though just matching the marginal distributions is not sufficient to ensure sparsity—as one could have some points with all dimensions close to the origin and some with all dimensions far away—a combination of the need to achieve good reconstructions and noise in the encoder process should prevent this from occurring. In short, provided the noise from the encoder is properly regulated, there is little information that can be stored in latent dimensions near the origin because of the high level of overlap forced in this region. Therefore, for a datapoint to be effectively encoded, it must have at least some of its latents dimensions outside of this region. Coupled with the need for most of the latent values to be near the origin to match the marginal distributions, this, in turn, enforces a sparse representation. Consequently, the loss in sparsity performance relative to using a hypothetical kernel that is both universal and has stable gradient estimates should only be relatively small, as is borne out in our empirical results. This may, however, be why we see a slight drop in sparsity performance for very large values of α.

We use a convolutional neural network for the encoder and a deconvolutional neural network for the decoder, whose architectures come from the DCGAN model (Radford et al., 2016) and are described in Table 1c. We use $[0, 1]$ normalised data as targets for the mean of a Laplace distribution with fixed scaling of 0.1. We rely on the Adam optimiser

with learning rate $5e^{-4}$, $\beta_1 = 0.5$, and $\beta_2 = 0.999$. The model is then trained (on the training set) for 80 epochs with a batch-size of 500.

As an extrinsic measure of *sparsity*, we use the *Hoyer* metric (Hurley and Rickard, 2008), defined for $\boldsymbol{y} \in \mathbb{R}^d$ by

$$\text{Hoyer}\,(\boldsymbol{y}) = \frac{\sqrt{d} - \|\boldsymbol{y}\|_1/\|\boldsymbol{y}\|_2}{\sqrt{d}-1} \in [0,1],$$

yielding 0 for a fully dense vector and 1 for a fully sparse vector. We additionally normalise each dimension to have a standard deviation of 1 under its aggregate distribution, i.e. we use $\bar{z}_d = z_d/\sigma(z_d)$ where $\sigma(z_d)$ is the standard deviation of dimension d of the latent encoding taken over the dataset. Overall sparsity is computed by averaging over the dataset as $\text{Sparsity} = 1/n \sum_i^n \text{Hoyer}\,(\bar{\boldsymbol{z}}_i)$.

As discussed in the main text, we use a trained model with $\alpha = 1000, \beta = 1$, and $\gamma = 0.8$ to perform a qualitative analysis of sparsity using the *Fashion-MNIST* dataset. Figure 7 shows the per-class average embedding magnitude for this model, a subset of which was shown in the main text. As can be seen clearly, the different classes utilise predominantly different subsets of dimensions to encode the image data, as one might expect for sparse representations.

C. Posterior regularisation

The aggregate posterior regulariser $\mathbb{D}(q(\boldsymbol{z}), p(\boldsymbol{z}))$ is a little more subtle to analyse than the entropy regulariser as it involves both the choice of divergence and potential difficulties in estimating that divergence. One possible choice is the exclusive Kullback-Leibler divergence $\text{KL}(q(\boldsymbol{z}) \,\|\, p(\boldsymbol{z}))$, as previously used (without additional entropy regularisation) by (Dilokthanakul et al., 2019; Esmaeili et al., 2019), but also implicitly by (Chen et al., 2018), through the use of a total correlation (TC) term. We now highlight a shortfall with this choice of divergence due to difficulties in its empirical estimation.

In short, the approaches used to estimate the $\text{H}[q(\boldsymbol{z})]$ (noting that $\text{KL}(q(\boldsymbol{z}) \,\|\, p(\boldsymbol{z})) = -\text{H}[q(\boldsymbol{z})] - \mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{z})]$, where the latter term can be estimated reliably by a simple Monte Carlo estimate) can exhibit very large biases unless very large batch sizes are used, resulting in quite different effects from what was intended. In fact, our results suggest they will exhibit behaviour similar to the β-VAE if the batch size is too small. These biases arise from the effects of nesting estimators (Rainforth et al., 2018a), where the variance in the nested (inner) estimator for $q(\boldsymbol{z})$ induces a bias in the overall estimator. Specifically, for any random variable $\hat{Z}$,

$$\mathbb{E}[\log(\hat{Z})] = \log(\mathbb{E}[\hat{Z}]) - \frac{\text{Var}[\hat{Z}]}{2Z^2} + O(\varepsilon)$$

where $O(\varepsilon)$ represents higher-order moments that get dominated asymptotically if $\hat{Z}$ is a Monte-Carlo estimator (see Proposition 1c in Maddison et al. (2017), Theorem 1 in Rainforth et al. (2018b), or Theorem 3 in Domke and Sheldon (2018)). In this setting, $\hat{Z} = \hat{q}(\boldsymbol{z})$ is the estimate used for $q(\boldsymbol{z})$. We thus see that if the variance of $\hat{q}(\boldsymbol{z})$ is large, this will induce a significant bias in our KL estimator.

To make things precise, we consider the estimator used for $\text{H}[q(\boldsymbol{z})]$ in Chen et al. (2018); Dilokthanakul et al. (2019); Esmaeili et al. (2019)

$$\text{H}[q(\boldsymbol{z})] \approx \hat{\text{H}} \triangleq -\frac{1}{B}\sum_{b=1}^{B} \log \hat{q}(\boldsymbol{z}_b), \text{ where} \tag{11a}$$

$$\hat{q}(\boldsymbol{z}_b) = \frac{q_\phi(\boldsymbol{z}_b|\boldsymbol{x}_b)}{n} + \frac{n-1}{n(B-1)} \sum_{b' \neq b} q_\phi(\boldsymbol{z}_b|\boldsymbol{x}'_b), \tag{11b}$$

$\boldsymbol{z}_b \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}_b)$, and $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_B\}$ is the mini-batch of data used for the current iteration for dataset size n. Esmaeili et al. (2019) correctly show that $\mathbb{E}[\hat{q}(\boldsymbol{z}_b)] = \tilde{q}(\boldsymbol{z}_b)$, with the first term of (11b) comprising an exact term in $\tilde{q}(\boldsymbol{z}_b)$ and the second term of (11b) being an unbiased Monte-Carlo estimate for the remaining terms in $\tilde{q}(\boldsymbol{z}_b)$.

To examine the practical behaviour of this estimator when $B \ll n$, we first note that the second term of (11b) is, in practice, usually very small and dominated by the first term. This is borne out empirically in our own experiments, and also noted in Kim and Mnih (2018). To see why this is the case, consider that given encodings of two independent data points, it is highly unlikely that the two encoding distributions will have any notable overlap (e.g. for a Gaussian encoder, the means will most likely be very many standard deviations apart), presuming a sensible latent space is being learned. Consequently, even though this second term is unbiased and may have an expectation comparable or even larger than the first, it is heavily skewed—it is usually negligible, but occasionally large in the rare instances where there is substantial overlap between encodings.

Let the second term of (11b) be T_2 and the event that this it is significant be E_S, such that $\mathbb{E}[T_2 \mid \neg E_s] \approx 0$. As explained above, $\mathbb{P}(E_S) \ll 1$ typically. We now have

$$\begin{aligned}
&\mathbb{E}\left[\hat{\text{H}}\right] \\
&= \mathbb{P}(E_S)\,\mathbb{E}\left[\hat{\text{H}} \mid E_S\right] + (1-\mathbb{P}(E_S))\,\mathbb{E}\left[\hat{\text{H}} \mid \neg E_S\right] \\
&= \mathbb{P}(E_S)\,\mathbb{E}\left[\hat{\text{H}} \mid E_S\right] + (1-\mathbb{P}(E_S)) \\
&\quad \cdot \left(\log n - \tfrac{1}{B}\textstyle\sum_{b=1}^{B} \mathbb{E}[\log q_\phi(\boldsymbol{z}_b|\boldsymbol{x}_b)|\neg E_S] - \mathbb{E}[T_2|\neg E_S]\right) \\
&= \mathbb{P}(E_S)\,\mathbb{E}\left[\hat{\text{H}} \mid E_S\right] + (1-\mathbb{P}(E_S)) \\
&\quad \cdot (\log n - \mathbb{E}[\log q_\phi(\boldsymbol{z}_1|\boldsymbol{x}_1)|\neg E_S] - \mathbb{E}[T_2|\neg E_S]) \\
&\approx \mathbb{P}(E_S)\,\mathbb{E}\left[\hat{\text{H}} \mid E_S\right] \\
&\quad + (1-\mathbb{P}(E_S))(\log n - \mathbb{E}[\log q_\phi(\boldsymbol{z}_1|\boldsymbol{x}_1)])
\end{aligned}$$

where the approximation relies firstly on our previous assumption that $\mathbb{E}[T_2 \mid \neg E_S] \approx 0$ and also that $\mathbb{E}[\log q_\phi(\boldsymbol{z}_1|\boldsymbol{x}_1) \mid \neg E_S] \approx \mathbb{E}[\log q_\phi(\boldsymbol{z}_1|\boldsymbol{x}_1)]$. This second

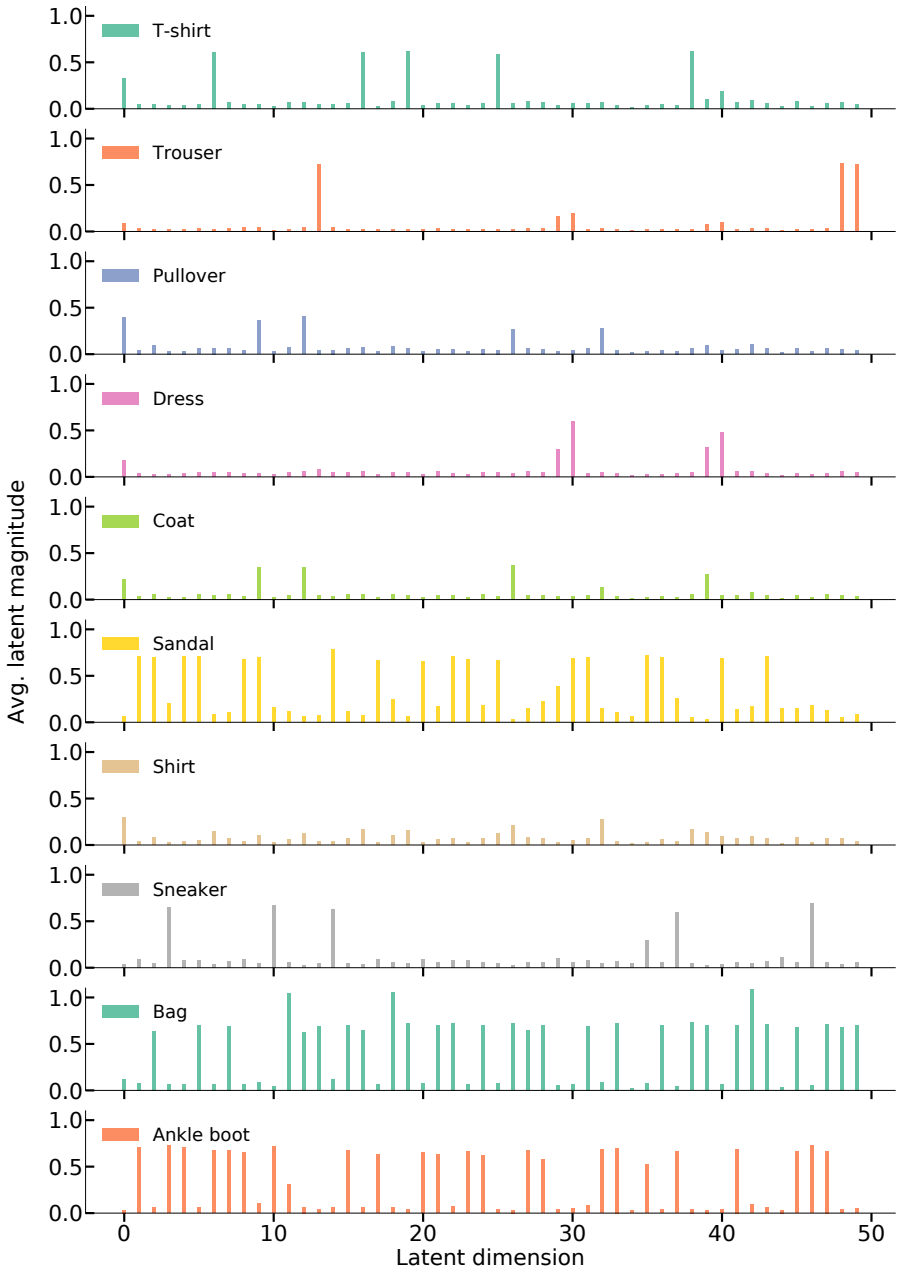


Figure 7. Average encoding magnitude over data for each classes in *Fashion-MNIST*.

assumption will also generally hold in practice, firstly because the occurrence of E_S is dominated by whether two similar datapoints are drawn (rather than by the value of $\boldsymbol{x}_1$) and secondly because $\mathbb{P}(E_S) \ll 1$ implies that

$$\begin{aligned}\mathbb{E}&[\log q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}_1)] \\ &= (1 - \mathbb{P}(E_S))\, \mathbb{E}[\log q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}_1) \mid \neg E_S] \\ &\quad + \mathbb{P}(E_S)\, \mathbb{E}[\log q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}_1) \mid E_S] \\ &\approx \mathbb{E}[\log q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}_1) \mid \neg E_S].\end{aligned}$$

Characterising $\mathbb{E}\left[\hat{\mathrm{H}} \mid E_S\right]$ precisely is a little more challenging, but it can safely be assumed to be smaller than $\mathbb{E}[\log q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}_1)]$, which is approximately what would result from all the $\boldsymbol{x}'_b$ being the same as $\boldsymbol{x}_b$. We thus see that even when the event E_S does occur, the resulting estimates will still, at most, be on a comparable scale to when it does not. Consequently, whenever E_S is rare, the $(1 - \mathbb{P}(E_S))\, \mathbb{E}\left[\hat{\mathrm{H}} \mid \neg E_S\right]$ term will dominate and we thus have

$$\begin{aligned}\mathbb{E}\left[\hat{\mathrm{H}}\right] &\approx \log n - \mathbb{E}[\log q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{x}_1)] \\ &= \log n + \mathbb{E}_{p(\boldsymbol{x})}[\mathrm{H}[q_\phi(\boldsymbol{z} \mid \boldsymbol{x})]].\end{aligned}$$

We now see that the estimator mimics the $\beta-$VAE regularisation up to a constant factor $\log n$, as adding the $\mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{z})]$ back in gives

$$\begin{aligned}&- \mathbb{E}\left[\hat{\mathrm{H}}\right] - \mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{z})] \\ &\quad \approx \mathbb{E}_{p(\boldsymbol{x})}[\mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}))] - \log n.\end{aligned}$$

We should thus expect to empirically see training with this estimator as a regulariser to behave similarly to the $\beta-$VAE with the same regularisation term whenever $B \ll n$. Note that the $\log n$ constant factor will not impact the gradients, but does mean that it is possible, even likely, that negative estimates for $\hat{\mathrm{KL}}$ will be generated, even though we know the true value is positive.

Overcoming the problem can, at least to a certain degree, be overcome by using very large batch sizes B, at an inevitable computational and memory cost. However, the problem is potentially exacerbated in higher dimensional latent spaces and larger datasets, for which one would typically expect the typical overlap of datapoints to decrease.

C.1. Other Divergences

As discussed in the main paper, $\mathrm{KL}(q(\boldsymbol{z}) \parallel p(\boldsymbol{z}))$ is far from the only aggregate posterior regulariser one might use. Though we do not analyse them formally, we expect many alternative divergence-estimator pairs to suffer from similar issues. For example, using Monte Carlo estimators with the inclusive Kullback-Leibler divergence $\mathrm{KL}(p(\boldsymbol{z}) \parallel q(\boldsymbol{z}))$ or the sliced Wasserstein distance (Kolouri et al., 2019) both result in nested expectations analogously to $\mathrm{KL}(q(\boldsymbol{z}) \parallel p(\boldsymbol{z}))$, and are therefore likely to similarly induce substantial bias without using large batch sizes.

Interestingly, however, MMD and generative adversarial network (GAN) regularisers of the form discussed in (Tolstikhin et al., 2018) do not result in nested expectations and therefore are necessarily not prone to the same issues: they produce unbiased estimates of their respective objectives. Though we experienced practical issues in successfully implementing both of these—we found the signal-to-noise-ratio of the MMD gradient estimates to be very low, particularly in high dimensions, while we experienced training instabilities for the GAN regulariser—their apparent theoretical advantages may indicate that they are preferable approaches, particularly if these issues can be alleviated. The GAN-based approach to estimating the total correlation introduced by Kim and Mnih (2018) similarly allows a nested expectation to be avoided, at the cost of converting a conventional optimization into a minimax problem.

Given the failings of the available existing approaches, we believe that further investigation into divergence-estimator pairs for $\mathbb{D}(q(\boldsymbol{z}), p(\boldsymbol{z}))$ in VAEs is an important topic for future work that extends well beyond the context of this paper, or even the general aim of achieving decomposition. In particular, the need for congruence between the posterior (encoder), likelihood (decoder), and marginal likelihood (data distribution) for a generative model, means that ensuring $q(\boldsymbol{z})$ is close to $p(\boldsymbol{z})$ is a generally important endeavour for training VAEs. For example, mismatch between $q(\boldsymbol{z})$ and $p(\boldsymbol{z})$ will cause samples drawn from the learned generative model to mismatch the true data-generating distribution, regardless of the fidelity of our encoder and decoder.

D. Characterising Overlap

Reiterating the argument from the main text, although the mutual information $I(\boldsymbol{x}; \boldsymbol{z})$ between data and latents provides a perfectly serviceable characterisation of overlap in a number of cases, the two are not universally equivalent and we argue that it is overlap which is important in achieving useful representations. In particular, if the form of the encoding distribution is not fixed—as when employing normalising flows, for example—$I(\boldsymbol{x}; \boldsymbol{z})$ does not necessarily characterise overlap well.

Consider, for example, an encoding distribution that is a mixture between the prior and a uniform distribution on a tiny ϵ-ball around the mean encoding $\mu_\phi(x)$, i.e. $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \lambda \cdot \mathrm{Uniform}\left(\|\mu_\phi(\boldsymbol{x}) - \boldsymbol{z}\|_2 < \epsilon\right) + (1-\lambda) \cdot p(\boldsymbol{z})$. If the encoder and decoder are sufficiently flexible to learn arbitrary representations, one now could arrive at *any* value for mutual information simply by an appropriate choice of λ. However, enforcing structuring of the latent space will be effectively impossible due to the lack of any pressure (other than a potentially small amount from internal regularization in the encoder network itself) for similar encodings to correspond to similar datapoints; the overlap between any two

encodings is the same unless they are within ϵ of each other.

While this example is a bit contrived, it highlights a key feature of overlap that $I(\boldsymbol{x}; \boldsymbol{z})$ fails to capture: $I(\boldsymbol{x}; \boldsymbol{z})$ does not distinguish between large overlap with a small number of other datapoints and small overlap with a large number of other datapoints. This distinction is important because we are particularly interested in *how many* other datapoints one datapoint's encoding overlaps with when imposing structure—the example setup fails because each datapoint has the same level of overlap with all the other datapoints.

Another feature that $I(\boldsymbol{x}; \boldsymbol{z})$ can fail to account for is a notion of *locality* in the latent space. Imagine a scenario where the encoding distributions are extremely multimodal with similar sized modes spread throughout the latent space, such as $q(\boldsymbol{z}|\boldsymbol{x}) = \sum_{i=1}^{1000} \mathcal{N}(\boldsymbol{z}; \mu_\phi(\boldsymbol{x}) + m_i, \sigma I)$ for some constant scalar σ, and vectors m_i. Again we can achieve almost any value for $I(\boldsymbol{x}; \boldsymbol{z})$ by adjusting σ, but it is difficult to impose meaningful structure regardless as each datapoint can be encoded to many different regions of the latent space.

References

Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, 2018.

Nat Dilokthanakul, Nick Pawlowski, and Murray Shanahan. Explicit information placement on latent variables using auxiliary generative modelling task, 2019. URL https://openreview.net/forum?id=H1l-SjA5t7.

Justin Domke and Daniel Sheldon. Importance weighting and varational inference. In *Advances in Neural Information Processing Systems*, pages 4471—4480, 2018.

Babak Esmaeili, Hao Wu, Sarthak Jain, N Siddharth, Brooks Paige, and Jan-Willem van de Meent. Hierarchical Disentangled Representations. *Artificial Intelligence and Statistics*, 2019.

Niall P. Hurley and Scott T. Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55:4723–4741, 2008.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Soheil Kolouri, Phillip E. Pope, Charles E. Martin, and Gustavo K. Rohde. Sliced wasserstein auto-encoders. In *International Conference on Learning Representations*, 2019.

Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pages 6573–6583, 2017.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

Tom Rainforth, Robert Cornish, Hongseok Yang, Andrew Warrington, and Frank Wood. On nesting monte carlo estimators. In *International Conference on Machine Learning*, pages 4264–4273, 2018a.

Tom Rainforth, Adam R. Kosiorek, Tuan Anh Le, Chris J. Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. *International Conference on Machine Learning*, 2018b.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

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