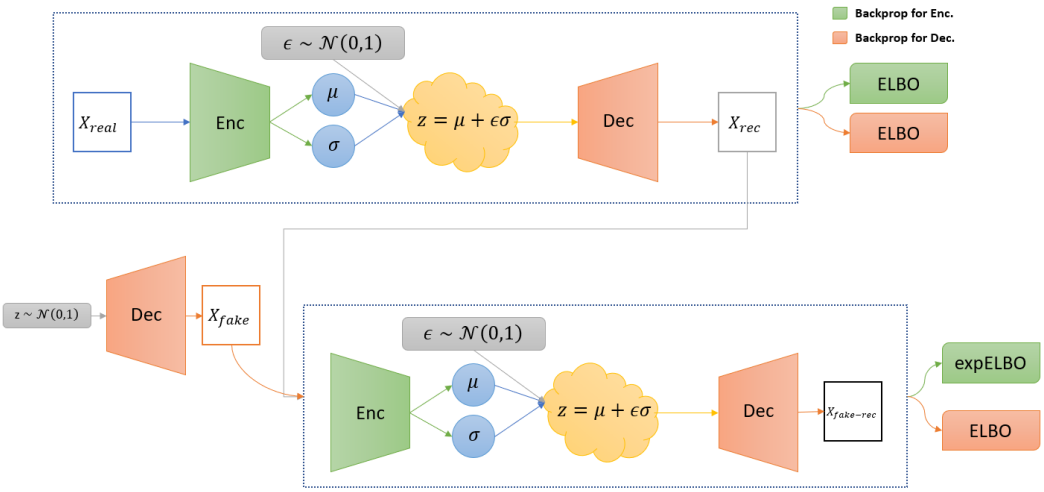


Figure 7: Training flow of Soft-IntroVAE. The ELBO for real samples is optimized for both encoder and decoder, while the encoder also optimizes the expELBO to 'push away' generated samples from the latent space, and the decoder optimizes the ELBO for the generated samples to 'fool' the encoder.

9. Appendix

9.1. Complete Algorithm

Algorithm 2 depicts the training procedure of Soft-IntroVAE. The difference from Algorithm 1 is the additional generated reconstructions, denoted with X_r, which are given the same treatment as the 'fake' generated data, denoted with X_f. In practice, following [25], we found it better to consider all generated data from the decoder, reconstructions ($X_r = D(E(x))$) and samples from $p(z)$, as 'fake' samples to speed-up convergence. In Algorithm 2, L_{rec} is the reconstruction error function (e.g. mean squared error – MSE) and KL is a function that calculates the KL divergence. The training flow of Soft-IntroVAE is further depicted in Figure 7.

9.1.1 IntroVAE and Soft-IntroVAE Objectives

Soft-IntroVAE Expanding S-IntroVAE's objective, which is *minimized*, from Eq. 4 with the complete set of hyperparameters:

$$\begin{aligned}\mathcal{L}_{E_\phi}(x,z) &= s\cdot(\beta_{rec}\mathcal{L}_r(x) + \beta_{kl}KL(x)) + \frac{1}{2}\exp(-2s\cdot(\beta_{rec}\mathcal{L}_r(D_\theta(z)) + \beta_{neg}KL(D_\theta(z)))),\\ \mathcal{L}_{D_\theta}(x,z) &= s\cdot\beta_{rec}\mathcal{L}_r(x) + s\cdot(\beta_{kl}KL(D_\theta(z)) + \gamma_r\cdot\beta_{rec}\mathcal{L}_r(D_\theta(z))),\end{aligned} \tag{7}$$

where $\mathcal{L}_r(x) = -\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right]$ denotes the reconstruction error.

IntroVAE Expanding IntroVAE's objective, which is *minimized*, from Eq. 2 with the complete set of hyperparameters:

$$\begin{aligned}\mathcal{L}_E(x,z) &= \beta_{rec}\mathcal{L}_r(x) + \beta_{kl}KL(x) + \beta_{neg}[m - KL(E(D_\theta(z)))]^+\\ \mathcal{L}_D(x,z) &= \beta_{rec}\mathcal{L}_r(x) + \beta_{neg}KL(E(D_\theta(z))).\end{aligned}$$

Note that the difference in hyperparameters from S-IntroVAE objectives is the added m hyperparameter for the hard-margin loss in the encoder. In S-IntroVAE, the objectives also include the reconstruction terms for the generated data, where in the decoder they are preceded by γ which is set $1e-8$ in all experiments. Also, recall that s is a normalizing constant that is set to the inverse of the input dimensions, and is not required in IntroVAE.

Algorithm 2 Training Soft-IntroVAE

Require: $\beta_{rec}, \beta_{kl}, \beta_{neg}, \gamma_r$

$\phi_E, \theta_D \leftarrow$ Initialize network parameters
$s \leftarrow 1/$input dim ▷ Scaling constant
while not converged **do**
  $X \leftarrow$ Random mini-batch from dataset
  $Z \leftarrow E(X)$ ▷ Encode
  $Z_f \leftarrow$ Samples from prior $N(0, I)$
  procedure UPDATEENCODER(ϕ_E)
    $X_r \leftarrow D(Z), X_f \leftarrow D(Z_f)$ ▷ Decode
    $Z_{rf} \leftarrow E(X_r), Z_{ff} \leftarrow E(X_f)$
    $X_{rf} \leftarrow D(Z_{rf}), X_{ff} \leftarrow D(Z_{ff})$
    $\text{ELBO} \leftarrow s \cdot ELBO(\beta_{rec}, \beta_{kl}, X, X_r, Z)$
    $\text{ELBO}_r \leftarrow ELBO(\beta_{rec}, \beta_{neg}, X_r, X_{rf}, Z_{rf})$
    $\text{ELBO}_f \leftarrow ELBO(\beta_{rec}, \beta_{neg}, X_f, X_{ff}, Z_{ff})$
    $\text{expELBO}_r \leftarrow 0.5 \exp(2s \cdot \text{ELBO}_r)$
    $\text{expELBO}_f \leftarrow 0.5 \exp(2s \cdot \text{ELBO}_f)$
    $L_E \leftarrow \text{ELBO} - 0.5 \cdot (\text{expELBO}_r + \text{expELBO}_f)$
    $\phi_E \leftarrow \phi_E + \eta \nabla_{\phi_E}(L_E)$ ▷ Adam update (ascend)
  end procedure
  procedure UPDATEDECODER(θ_D)
    $X_r \leftarrow D(Z), X_f \leftarrow D(Z_f)$ ▷ Decode
    $Z_{rf} \leftarrow E(X_r), Z_{ff} \leftarrow E(X_f)$
    $X_{rf} \leftarrow sg(D(Z_{rf})), X_{ff} \leftarrow sg(D(Z_{ff}))$
    $\text{ELBO} \leftarrow \beta_{rec} L_{rec}(X, X_r)$
    $\text{ELBO}_r \leftarrow ELBO(\gamma_r \cdot \beta_{rec}, \beta_{kl}, X_r, X_{rf}, Z_{rf})$
    $\text{ELBO}_f \leftarrow ELBO(\gamma_r \cdot \beta_{rec}, \beta_{kl}, X_f, X_{ff}, Z_{ff})$
    $L_D \leftarrow s \cdot (\text{ELBO} + 0.5 \cdot (\text{ELBO}_r + \text{ELBO}_f))$
    $\theta_D \leftarrow \theta_D + \eta \nabla_{\theta_D}(L_D)$ ▷ Adam update (ascend)
  end procedure
end while

function ELBO($\beta_{rec}, \beta_{kl}, X, X_r, Z$)
  $ELBO \leftarrow -1 \cdot (\beta_{rec} L_{rec}(X, X_r) + \beta_{kl} KL(Z))$
  return $ELBO$
end function

9.2. Datasets, Architectures and Hyperparameters

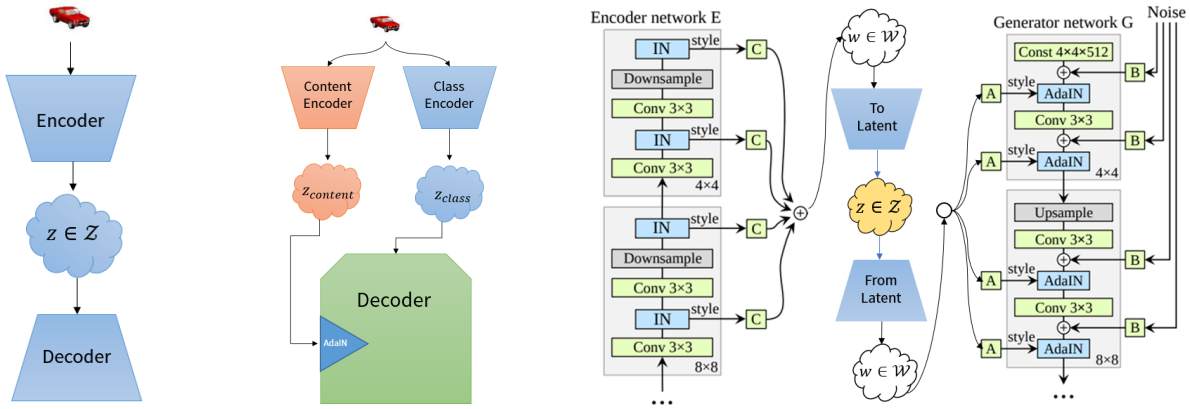
We implement our method in PyTorch [43]. For all experiments, we used the Adam [31] optimizer with the default parameters, and γ_r was set to $1e-8$ independently of the dataset. In practice, γ_r should be set to a small value. Note that setting $\gamma_r = 0$ can cause a degradation in performance. Experiments with the style-based architecture were run on a machine with 4 Nvidia RTX 2080 GPUs, while the rest used a machine with one GPU of the same type. In what follows, we detail the dataset-specific hyperparameters.

9.2.1 2D Experiments

The architecture for all methods is a simple 3-layer fully-connected network with 256 hidden units and ReLU activations, and the latent space dimension is 2. We used a learning rate of $2e-4$, batch size of 512 and ran a total of 30,000 iterations per dataset. We ran an extensive hyperparameter grid search of 81 runs for the standard VAE, 210 runs for S-IntroVAE, and 1260 runs for IntroVAE. The range of the search was $[0.05, 1.0]$ for β_{kl} and β_{rec}, $[\beta_{kl}, 5\beta_{kl}]$ for β_{neg} and $[1, 10]$ for m. The best combinations of hyperparameters are provided in Table 5.

		VAE	IntroVAE	Soft-IntroVAE
8 Gaussians	β_{rec}	0.8	0.3	0.2
	β_{kl}	0.05	0.5	0.3
	β_{neg}	-	1.0	0.9
	m	-	1.0	-
Spiral	β_{rec}	1.0	0.2	0.2
	β_{kl}	0.05	0.5	0.5
	β_{neg}	-	0.5	1.0
	m	-	2.0	-
Checkerboard	β_{rec}	0.8	0.4	0.2
	β_{kl}	0.1	0.2	0.1
	β_{neg}	-	0.2	0.2
	m	-	8.0	-
Rings	β_{rec}	0.8	0.8	0.2
	β_{kl}	0.05	0.5	0.2
	β_{neg}	-	0.5	1.0
	m	-	5.0	-

Table 5: Hyperparameters for the 2D datasets.



(a) Standard Architecture (b) Disentanglement Architecture (c) Style-based Architecture inspired by [44]

Figure 8: Different architectures used in our experiments. (a) Standard MLP/CNN encoder-decoder architectures; (b) Based on the architecture proposed by [12]: separate encoders with different latent spaces are learned to disentangle class from content. The decoder uses adaptive Instance Normalization to account for the class latent variable when decoding the content; (c) Style-based architecture proposed by [44]: the encoder extracts styles which are then mapped to the latent space. The latent variable is then mapped back to styles, which are finally decoded with a style-based decoder.

9.2.2 Image Generation

CIFAR-10 [35] consists of 60,000 32x32 colour images in 10 classes, with 6,000 images per class. We use the official split of 50,000 training images and 10,000 test images and evaluate in both unconditional and class-conditional settings. We use IntroVAE's [25] architecture[4] with a latent dimension of 128 and train the model for 220 epochs with a learning rate of $2e-4$ and batch size of 32. The hyperparameters are $\beta_{rec} = \beta_{kl} = 1$ and $\beta_{neg} = 256$. IntroVAE's encoder-decoder general architecture with residual-based convolutional layers for images at resolution of 1024×1024 is depicted in Table 6. For CIFAR-10 we used 3 residual blocks in both encoder and decoder with channels (64, 128, 256).

CelebA-HQ [28] is an improved version of CelebA [37], and consists of a subset of 30,000 high-quality 1024x1024 images of celebrities, which are split to 29,000 train images and 1,000 test images. FFHQ [29] is a high-quality image dataset consisting of 70,000 images of people faces aligned and cropped at resolution of 1024x1024, split to 60,000 train images and 10,000 test images.

[4] https://github.com/hhb072/IntroVAE

Encoder	Act.	Output shape
Input image	–	$3 \times 1024 \times 1024$
Conv	$5 \times 5, 16$	$16 \times 1024 \times 1024$
AvgPool	–	$16 \times 512 \times 512$
Res-block	$\begin{bmatrix} 1 \times 1, & 32 \\ 3 \times 3, & 32 \\ 3 \times 3, & 32 \end{bmatrix}$	$32 \times 512 \times 512$
AvgPool	–	$32 \times 256 \times 256$
Res-block	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 3 \times 3, & 64 \end{bmatrix}$	$64 \times 256 \times 256$
AvgPool	–	$64 \times 128 \times 128$
Res-block	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 3 \times 3, & 128 \end{bmatrix}$	$128 \times 128 \times 128$
AvgPool	–	$128 \times 64 \times 64$
Res-block	$\begin{bmatrix} 1 \times 1, & 256 \\ 3 \times 3, & 256 \\ 3 \times 3, & 256 \end{bmatrix}$	$256 \times 64 \times 64$
AvgPool	–	$256 \times 32 \times 32$
Res-block	$\begin{bmatrix} 1 \times 1, & 512 \\ 3 \times 3, & 512 \\ 3 \times 3, & 512 \end{bmatrix}$	$512 \times 32 \times 32$
AvgPool	–	$512 \times 16 \times 16$
Res-block	$\begin{bmatrix} 1 \times 1, & 512 \\ 3 \times 3, & 512 \\ 3 \times 3, & 512 \end{bmatrix}$	$512 \times 16 \times 16$
AvgPool	–	$512 \times 8 \times 8$
Res-block	$\begin{bmatrix} 3 \times 3, & 512 \\ 3 \times 3, & 512 \end{bmatrix}$	$512 \times 8 \times 8$
AvgPool	–	$512 \times 4 \times 4$
Res-block	$\begin{bmatrix} 3 \times 3, & 512 \\ 3 \times 3, & 512 \end{bmatrix}$	$512 \times 4 \times 4$
Reshape	–	$8192 \times 1 \times 1$
FC-1024	–	$1024 \times 1 \times 1$
Split	–	$512, 512$

Decoder	Act.	Output shape
Latent vector	–	$512 \times 1 \times 1$
FC-8192	ReLU	$8192 \times 1 \times 1$
Reshape	–	$512 \times 4 \times 4$
Res-block	$\begin{bmatrix} 3 \times 3, & 512 \\ 3 \times 3, & 512 \end{bmatrix}$	$512 \times 4 \times 4$
Upsample	–	$512 \times 8 \times 8$
Res-block	$\begin{bmatrix} 3 \times 3, & 512 \\ 3 \times 3, & 512 \end{bmatrix}$	$512 \times 8 \times 8$
Upsample	–	$512 \times 16 \times 16$
Res-block	$\begin{bmatrix} 3 \times 3, & 512 \\ 3 \times 3, & 512 \end{bmatrix}$	$512 \times 16 \times 16$
Upsample	–	$512 \times 32 \times 32$
Res-block	$\begin{bmatrix} 1 \times 1, & 256 \\ 3 \times 3, & 256 \\ 3 \times 3, & 256 \end{bmatrix}$	$256 \times 32 \times 32$
Upsample	–	$256 \times 64 \times 64$
Res-block	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 3 \times 3, & 128 \end{bmatrix}$	$128 \times 64 \times 64$
Upsample	–	$128 \times 128 \times 128$
Res-block	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 3 \times 3, & 64 \end{bmatrix}$	$64 \times 128 \times 128$
Upsample	–	$64 \times 256 \times 256$
Res-block	$\begin{bmatrix} 1 \times 1, & 32 \\ 3 \times 3, & 32 \\ 3 \times 3, & 32 \end{bmatrix}$	$32 \times 256 \times 256$
Upsample	–	$32 \times 512 \times 512$
Res-block	$\begin{bmatrix} 1 \times 1, & 16 \\ 3 \times 3, & 16 \\ 3 \times 3, & 16 \end{bmatrix}$	$16 \times 512 \times 512$
Upsample	–	$16 \times 1024 \times 1024$
Res-block	$\begin{bmatrix} 3 \times 3, & 16 \\ 3 \times 3, & 16 \end{bmatrix}$	$16 \times 1024 \times 1024$
Conv	$5 \times 5, 3$	$3 \times 1024 \times 1024$

Table 6: IntroVAE's general architecture for images at resolution 1024×1024.

Style-based architecture The decoder in the style-based architecture borrows the same properties of StyleGAN's [29] generator, while the encoder is designed after the novel architecture in ALAE [44][5]. Every layer in StyleGAN's generator is driven by a style input $w \in \mathcal{W}$, which requires that the encoder will also encode *styles*. Thus, in the style-based architecture the layers in the encoder and decoder are symmetric, such that every layer extracts and injects styles, correspondingly. This is made possible by using Instance Normalization (IN) layers [26], which provide instance means and standard deviations for every *channel*.

Mathematically, let y_i^E denote the output of the i-th layer in the encoder E, the IN module extracts the statistics $\mu(y_i^E)$ and $\sigma(y_i^E)$, representing the style at that level. The second output of the IN module is the normalized version of the input which continues down the pipeline with no more style information. Finally, the latent style variable, w, is a weighted sum of

[5] https://github.com/podgorskiy/ALAE

the extracted styles:

$$w = \sum_{i=1}^{N} C_i \begin{bmatrix} \mu(y_i^E) \\ \sigma(y_i^E) \end{bmatrix},$$

where C_i's are learned parameters and N is the number of layers. The style latent variable is then mapped with a fully-connected network to the mean, μ_q, and standard deviation, σ_q, of the Gaussian latent variable $z \in \mathcal{N}(\mu_q, \sigma_q^2)$, which is done efficiently using the reparameterization trick.

Symmetrically, the latent variable z is mapped back to a style latent variable w using a fully-connected network, which serves as inputs to the Adaptive Instance Normalization (AdaIN) layers [26] in the decoder. The complete style-based architecture is depicted in Figure 8c. In our experiments the latent variables z and w have the same dimensions of 512, the mapping from style to latent in E has 3 layers, while the mapping from latent to style in D has 8 layers, both with 512 hidden units in each layer.

The training using the style-based architecture is done is a progressive growing fashion, similar to [28, 29, 44], where we start from low resolution (4×4 pixels) and progressively increase the resolution by smoothly blending in new blocks to E and D. For CelebA-HQ $\beta_{rec} = 0.05$, and for FFHQ $\beta_{rec} = 0.1$, while for both datasets $\beta_{kl} = 0.2$ and $\beta_{neg} = 512$, and the maximal learning rate is $1.5e-3$. The CelebA-HQ model is trained for 230 epochs and the FFHQ model for 270 epochs, where the training reaches the 256×256 resolution at epoch 180 (30 epoch per resolution until 256×256).

9.3. Image Translation

For the image translation experiments, we use the architecture proposed in LORD [12], and use two encoders, one for the class and one for the content, where the latent representation of the class controls the adaptive Instance Normalization [26] of the latent representation of the content in the decoder. More specifically, the encoder is composed of convolutional layers with channels (64, 128, 256, 256), followed by 3 fully-connected layers to produce the parameters of the Gaussian latent variable. The decoder consists of 3 fully-connected layers followed by 6 convolutional layers, where the first 4 are preceded by an upsampling layer and followed by AdaIN normalization, that uses the latent representation of the class. All layers are activated with LeakyReLU. This architecture is depicted in Figure 8b.

For this specific choice of architecture, note that the KL terms update each encoder separately, the reconstruction term jointly updates both encoders, as it is a function of the latents from both encoders.

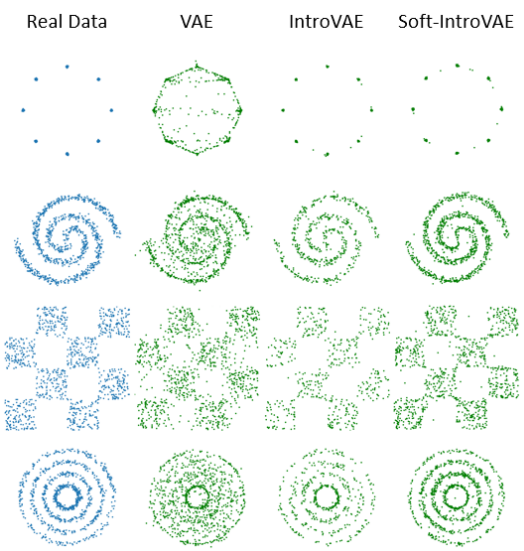
Similar to [12], all images are resized to 64×64 resolution and we set the latent dimension of the class to be 256, and 128 for the content. In all experiments we used Adam optimizer with a learning rate of $1e-4$, batch size of 64 and ran a total of 400 train epochs.

Cars3D dataset The Cars3D dataset [46] consists of 183 car CAD models, labelled with 24 azimuth directions and 4 elevations. For this dataset, the class is considered to be the car model and the azimuth and elevation as the content. We use 163 car models for training and the other 20 are held out for testing. As Cars3D includes ground-truth labels, we are able to test the quality of disentanglement using the same evaluation procedure as in [12], by measuring the content transfer reconstruction loss. As suggested by [12], we replace the pixel-wise MSE reconstruction loss with the VGG perceptual loss as implemented by [24]. The hyperparameters used for this dataset: $\beta_{kl}^{content} = \beta_{kl}^{class} = 1.0, \beta_{rec} = 0.5, \beta_{neg}^{content} = 2048$ and $\beta_{kl}^{class} = 1024$.

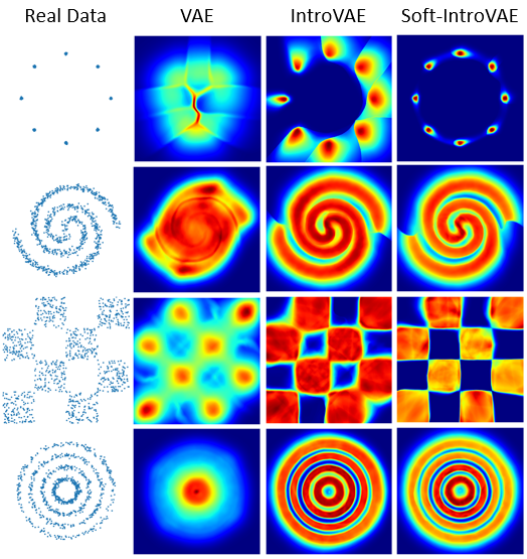
KTH dataset We further evaluate on the KTH dataset [48] which contains videos of 25 people performing different activities. For training our model, we extract grayscale image frames from all of the videos. As there are no ground-truth labels, we *assume* the class is the person identity and the content is other unlabeled transitory attributes such as skeleton position. Similarly to [12], due to the very limited amount of subjects, we use all the identities for training, holding out 10% of the images for testing. Moreover, we found that using MSE pixel-wise loss worked better for the grayscale images than the VGG perceptual loss. The hyperparameters used for this dataset: $\beta_{kl}^{content} = \beta_{kl}^{class} = 0.5, \beta_{rec} = 1.0, \beta_{neg}^{content} = 2048$ and $\beta_{kl}^{class} = 1024$.

9.4. Additional Results

In this section, we provide additional results from the experiments we described.



(a) Samples from the trained models.



(b) Density estimation with the trained models.

Figure 9: Unsupervised learning of 2D datasets.

9.4.1 2D Toy Datasets

The complete set of samples and density estimation can be found in Figure 9.

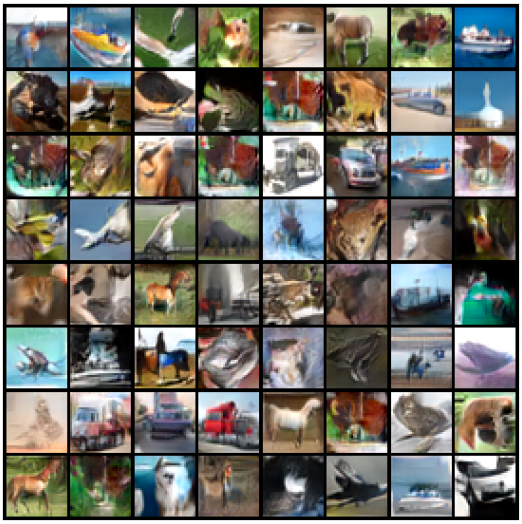
9.4.2 Image Generation and Reconstruction

CIFAR-10 dataset We trained two types of models: (1) unconditional model and (2) class-conditional model. In Figure 10a we present random (i.e., no cherry-picking) samples from a trained unconditional model (FID: 4.6), and Figure 10b presents random reconstructions from the test set. For the conditional model, we used a one-hot vector representation for the class, and trained a conditional VAE (CVAE) using Soft-IntroVAE's objectives. Random samples from the class-conditional model can be seen in Figure 11a (FID: 4.07), and random reconstructions from the test set in Figure 11b. It can be seen that when including a supervision signal (class labels), the samples tend to be slightly more structured, which is also reflected in the FID score.

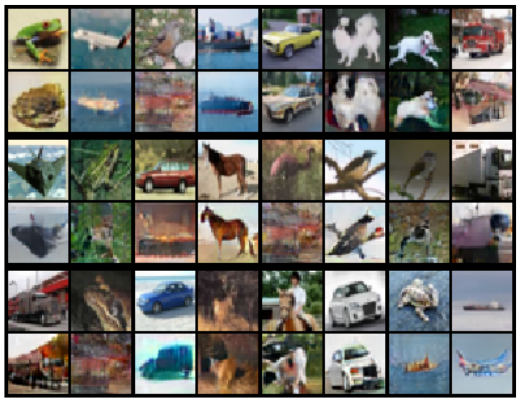
CelebA-HQ dataset Results from a style-based S-IntroVAE trained on CelebA-HQ at resolution 256×256 (FID: 18.63) are presented in Figure 14. Additional random (i.e., no cherry-picking) generated images from a style-based S-IntroVAE trained on CelebA-HQ at resolution 256×256 are presented in Figure 12 and random reconstructions of unseen data during training are presented in Figure 13.

FFHQ dataset Additional random (i.e., no cherry-picking) generated images from a style-based S-IntroVAE trained on FFHQ at resolution 256×256 are presented in Figure 16 (FID: 17.55) and random reconstructions of unseen data during training are presented in Figure 17.

LSUN Bedroom LSUN Bedroom is a subset of the larger LSUN [56] dataset, and includes a training set of 3,033,042 images of different bedrooms. We train a style-based S-IntroVAE at a resolution of 128×128. Samples from the trained model are presented in Figure 15, and we report FID of 15.88.

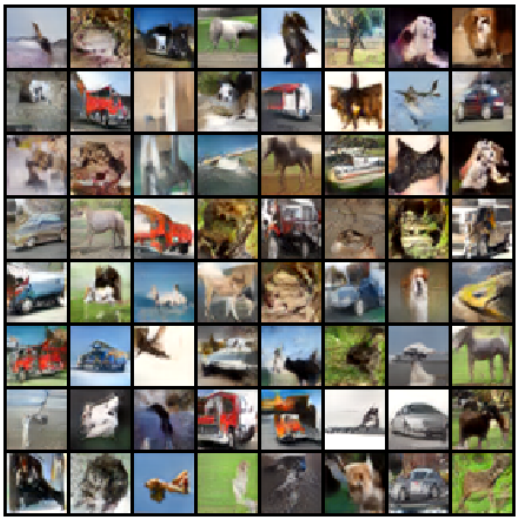


(a) Generated samples (FID: 4.6).

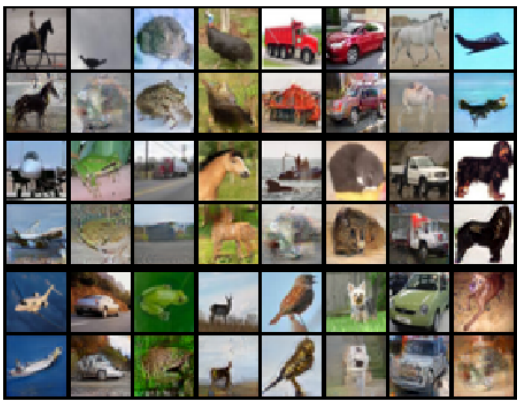


(b) Reconstructions (odd row: real, even row: reconstruction).

Figure 10: Generated samples (left) and reconstructions (right) of test data from an unconditional S-IntroVAE trained on CIFAR-10.



(a) Generated samples (FID: 4.07).



(b) Reconstructions (odd row: real, even row: reconstruction).

Figure 11: Generated samples (left) and reconstructions (right) of test data from a class-conditional S-IntroVAE trained on CIFAR-10.

9.4.3 Interpolation in the Latent Space

One of the desirable properties of VAEs is the continuous learned latent space. Figure 18 shows smooth interpolation between the latent vectors of four images from S-IntroVAE trained on the CelebA-HQ dataset. The interpolation is performed as

Figure 12: Generated samples from a style-based S-IntroVAE trained on CelebA-HQ at 256x256 resolution (FID: 18.63).

follows: the four images are encoded to the latent space, and the latent codes serve as the corners of a square. We then perform 7-step linear interpolation between the latent codes of the corner images, such that each intermediate code is a mixture of the corner latent code, depending on the location on the grid. The intermediate latent codes are then decoded to

Figure 13: Reconstructions of test data from a style-based S-IntroVAE trained on CelebA-HQ at 256x256 resolution (left: real, right: reconstruction).

produce the images comprising the square. Mathematically, let z_a, z_b, z_c and z_d denote the latent codes of images X_a, X_b, X_c and X_d, respectively. The intermediate latent code z_m is constructed as follows:

$$z_m = z_a \cdot (1 - \frac{i}{7})(1 - \frac{j}{7}) + z_b \cdot \frac{j}{7}(1 - \frac{i}{7}) + z_c \cdot (1 - \frac{j}{7})\frac{i}{7} + z_d \cdot \frac{i}{7} \cdot \frac{j}{7},$$

(a) Generated samples from S-IntroVAE (FID: 18.63). (b) Reconstructions (left: real, right: reconstruction).

Figure 14: Generated samples (left) and reconstructions (right) of test data from a style-based S-IntroVAE trained on CelebA-HQ at 256×256 resolution. It is recommended to zoom-in.

where $i, j = 1, .., 6$ denote the current location on the grid.

9.4.4 Image Translation

We provide further image translation results for the Cars3D dataset in Figure 19 and for the KTH dataset in Figure 20. The content transfer is performed as follows: for given two images, we encode both of them, and use the class latent code of the first one and the content latent code from the second one as input to the decoder. The output image should contain an object from the class of the first image (e.g., car model or person identity), with the content of the second (e.g. rotation or skeleton position).

9.5. Posterior Collapse

Posterior collapse [3], often occurs in image, text or autoregressive-based VAEs, happens when the approximate posterior distribution collapses onto the prior completely, that is, a trivial optimum is reached, a solution where the generator ignores the latent variable z when generating x, and the KL term in the ELBO vanishes. Preventing posterior collapse has been addressed previously [15, 20, 39], mainly by annealing the KL coefficient term (β_{kl}), adding auxiliary costs or changing the cost function altogether. [12] also noticed that when using a VAE formulation to train the specific disentanglement-oriented architecture on images, the KL term vanishes and the learned representations are uninformative. Empirically, posterior collapse can happen when the optimization of the VAE is more focused on the KL term, i.e., when $\beta_{kl} > \beta_{rec}$. Interestingly, we found that for the same β_{kl} and β_{kl}, the expELBO term in the encoder's objective adds a 'repulsion' force that prevents this collapse. This is demonstrated on the 2D "8 Gaussians" dataset in Figure 21, where we train a standard VAE with $\beta_{kl} = 1.0$ and $\beta_{rec} = 0.5$, and a Soft-IntroVAE model with the same hyperparameters, but with $\beta_{neg} = 5.0$. For the standard VAE, the KL term quickly vanishes during training, resulting in a trivial solution where the decoder ignores the latent variable z when generating x. Moreover, [12] analysis showed that using a standard β-VAE for the image translation task results in sub-optimal results compared to a regular autoencoder due to the KL term vanishing. Our results on the image translation task show that with the added objectives of S-IntroVAE, it is possible to use a VAE for this task.

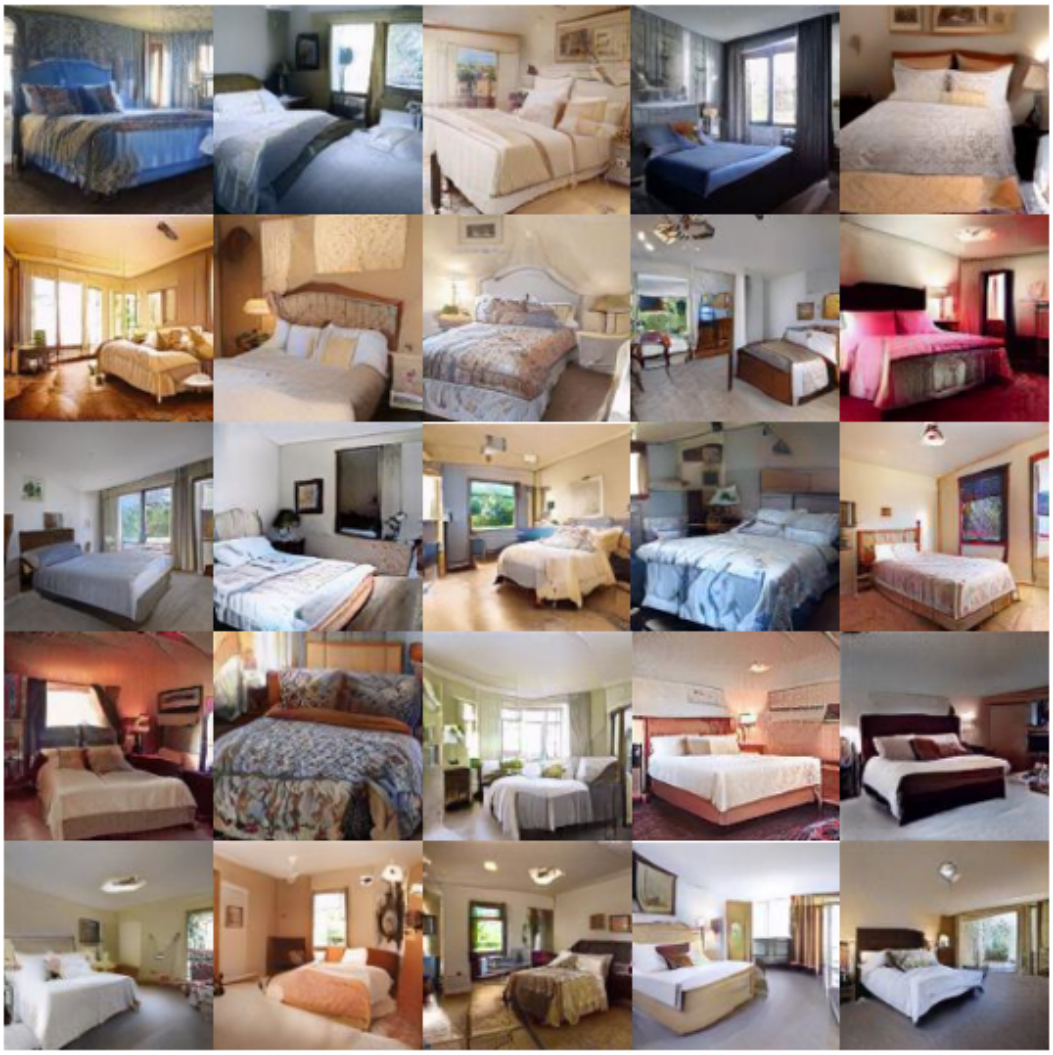


Figure 15: Samples a style-based S-IntroVAE trained on LSUN Bedroom at 128×128 resolution (FID: 15.88).

Figure 16: Generated samples from a style-based S-IntroVAE trained on FFHQ at 256x256 resolution (FID: 17.55).

Figure 17: Reconstructions of test data from a style-based S-IntroVAE trained on FFHQ at 256x256 resolution (left: real, right: reconstruction).

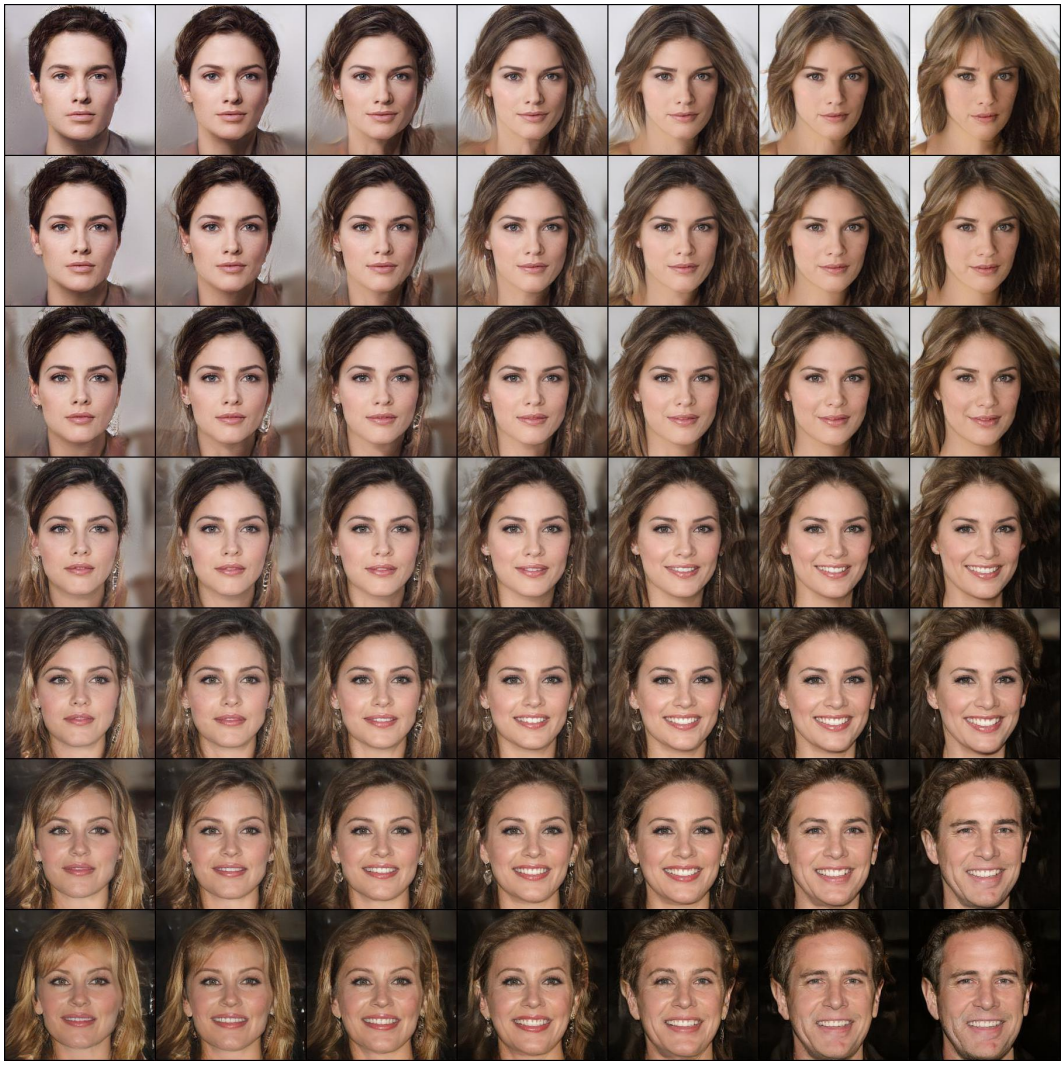


Figure 18: Interpolation in the latent space between four images, using a style-based S-IntroVAE trained on CelebA-HQ at 256x256 resolution.

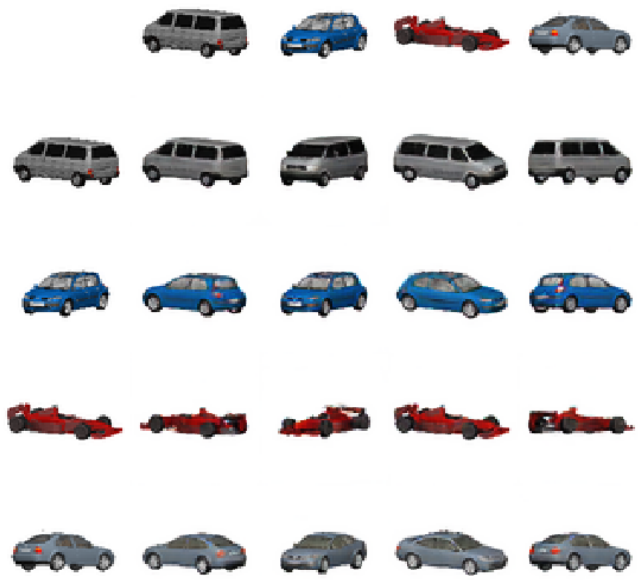


Figure 19: Qualitative results for content transfer on test data from the Cars3D dataset. The class is the car model, and the content is the rotation and azimuth.

Figure 20: Qualitative results for content transfer on test data from the KTH dataset. The class is the person identity, and the content is the skeleton position.

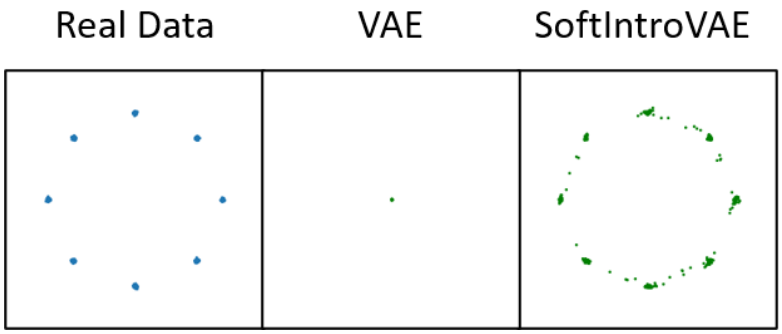


Figure 21: Demonstration of posterior collapse. Generated samples from trained models are shown, where both the standard VAE and S-IntroVAE were trained on the "8 Gaussians" 2D dataset with $\beta_{kl} = 1.0$ and $\beta_{rec} = 0.5$, and for S-IntroVAE, $\beta_{neg} = 5.0$. For the standard VAE, the KL term vanishes, resulting in posterior collapse.

9.6. Theoretical Results

In this section, we analyze the equilibrium of the S-IntroVAE model. We analyze the case of a general $\alpha \geq 1$, and the results in the main text for $\alpha = 1$ are a special case. For the readers ease, we first recapitulate our definitions. Recall that the encoder is represented by the approximate posterior distribution $q \doteq q(z|x)$ and that the decoder is represented using $d \doteq p_d(x|z)$. These are the controllable distributions in our generative model. The latent prior is denoted $p(z)$ and is not controlled. Slightly abusing notation, we also denote $p_d(x) = \mathbb{E}_{p(z)}[p_d(x|z)]$. The data distribution is denoted $p_{data}(x)$. For some distribution $p(x)$, let $H(p) = -\mathbb{E}[\log p(x)]$ denote its Shannon entropy.

The ELBO, denoted $W(x; d, q)$, is given by:

$$W(x; d, q) \doteq \mathbb{E}_{q(z|x)}[\log p_d(x|z)] - KL(q(z|x)||p(z)). \tag{8}$$

From the Radon-Nikodym Theorem [6] of measure theory the following equality holds:

$$\mathbb{E}_{z\sim p_z(z)}[\exp(\alpha W(D_\theta(z); d, q))] = \mathbb{E}_{x\sim p_d(x)}[\exp(\alpha W(x; d, q))], \tag{9}$$

and similarly:

$$\mathbb{E}_{z\sim p_z(z)}[W(D_\theta(z); d, q)] = \mathbb{E}_{x\sim p_d(x)}[W(x; d, q)]. \tag{10}$$

The ELBO satisfies the following property:

$$W(x; d, q) = \log p_d(x) - KL(q(z|x)||p_d(z|x)) \leq \log p_d(x). \tag{11}$$

We consider a non-parametric setting, where d and q can be any distribution. For some z, let $D(z)$ denote a sample from $p_d(x|z)$. The objective functions for q and d are given by (note that we drop the dependence on θ, ϕ because of the non-parametric setting):

$$\begin{aligned} \mathcal{L}_E(x, z) &= W(x; d, q) - \frac{1}{\alpha}\exp(\alpha W(D(z)); d, q), \\ \mathcal{L}_D(x, z) &= W(x; d, q) + \gamma W(D(z); d, q), \end{aligned} \tag{12}$$

where $\alpha \geq 1$ and $\gamma \geq 0$ are hyper-parameters. The complete S-IntroVAE objective, takes an expectation of the losses above over real and generated samples:

$$\begin{aligned} L_q(q, d) &= \mathbb{E}_{p_{data}}[W(x; q, d)] - \mathbb{E}_{p_d}\left[\alpha^{-1}\exp(\alpha W(x; q, d))\right], \\ L_d(q, d) &= \mathbb{E}_{p_{data}}[W(x; q, d)] + \gamma\mathbb{E}_{p_d}[W(x; q, d)]. \end{aligned} \tag{13}$$

A Nash equilibrium point (q^*, d^*) satisfies $L_q(q^*, d^*) \geq L_q(q, d^*)$ and $L_d(q^*, d^*) \geq L_d(q^*, d)$ for all q, d. Given some d, let $q^*(d)$ satisfy $L_q(q^*(d), d) \geq L_q(q, d)$ for all q.

Lemma 4. *If $p_d(x) \leq p_{data}(x)^{\frac{1}{\alpha+1}}$ for all x for which $p_{data}(x) > 0$, we have that $q^*(d)$ satisfies $q^*(d)(z|x) = p_d(z|x)$, and $W(x; q^*(d), d) = \log p_d(x)$.*

Proof. Plugging equation 11 in equation 13 we have that:

$$\begin{aligned} L_q(q, d) &= \mathbb{E}_{p_{data}}[\log p_d(x) - KL(q(z|x)||p_d(z|x))] - \frac{1}{\alpha}\mathbb{E}_{p_d}[\exp(\alpha\log p_d(x) - \alpha KL(q(z|x)||p_d(z|x)))] \\ &= \mathbb{E}_{p_{data}}[\log p_d(x) - KL(q(z|x)||p_d(z|x))] - \frac{1}{\alpha}\mathbb{E}_{p_d}[p_d^\alpha(x)\exp(-\alpha KL(q(z|x)||p_d(z|x)))] \\ &= \sum_x p_{data}(x)(\log p_d(x) - KL(q(z|x)||p_d(z|x))) - \frac{1}{\alpha}p_d^{\alpha+1}(x)\exp(-\alpha KL(q(z|x)||p_d(z|x))). \end{aligned} \tag{14}$$

Consider some x for which $p_{data}(x) > 0$. We have that $q^*(d)(z|x)$ is the maximizer of

$$p_{data}(x)\left(\log p_d(x) - KL(q(z|x)||p_d(z|x)) - \frac{1}{\alpha}\cdot\frac{p_d^{\alpha+1}(x)}{p_{data}(x)}\exp(-\alpha KL(q(z|x)||p_d(z|x)))\right). \tag{15}$$

Consider now the function $g(y) = y - \frac{a}{\alpha}\exp(\alpha y)$. We have that $g'(y) = 1 - a\exp(\alpha y)$, and therefore the function obtains a maximum at $y = -\frac{1}{\alpha}\log(a)$. In our case, $a = \frac{p_d^{\alpha+1}(x)}{p_{data}(x)}$ and $y = -KL(q(z|x)\|p_d(z|x)) \leq 0$. Therefore, if $\frac{p_d^{\alpha+1}(x)}{p_{data}(x)} > 1$, then the maximum is obtained for $-KL(q(z|x)||p_d(z|x)) = -\frac{1}{\alpha}\log\left(\frac{p_d^{\alpha+1}(x)}{p_{data}(x)}\right)$, and if $\frac{p_d^{\alpha+1}(x)}{p_{data}(x)} \leq 1$ then the maximum is obtained for $-KL(q(z|x)||p_d(z|x)) = 0$.

For x such that $p_{data}(x) = 0$, we have that $q(z|x)$ is the maximizer of $-\frac{1}{\alpha}\cdot p_d^{\alpha+1}(x)\exp(-\alpha KL(q(z|x)||p_d(z|x)))$. Since $KL(\cdot||\cdot) \geq 0$ and $p_d^{\alpha+1}(x) \geq 0$, a maximum is obtained for $-KL(q(z|x)||p_d(z|x)) = 0$. Thus, given the assumption in the Lemma, for every x the maximum is obtained for $KL(q(z|x)||p_d(z|x)) = 0$. □

Define d^* as follows:

$$d^* \in \arg\min_d \left\{KL(p_{data}||p_d) + \gamma H(p_d(x))\right\}, \tag{16}$$

where $H(\cdot)$ is the Shannon entropy. We make the following assumption.

Assumption 5. *For all x such that $p_{data}(x) > 0$ we have that $p_{d^*}(x) \leq p_{data}(x)^{\frac{1}{\alpha+1}}$.*

For $\alpha = 1$, we get that Assumption 5 is equivalent to Assumption 1 in the main text. We now claim that the equilibrium point of the optimization in equation 13 is $(q^*(d^*), d^*)$ as defined in equation 16.

Theorem 6. *Denote $q^* = p_{d^*}(z|x)$, with d^* defined in equation 16, and let Assumption 5 hold. Then (q^*, d^*) is a Nash equilibrium of equation 13.*

Proof. From Lemma 4 we have that $q^*(d^*)(z|x) = p_{d^*}(z|x)$.

Let d be some decoder parameters (i.e., $p_d(x|z)$). From equation 11 we have that $W(x; q^*(d), d) = \log(p_d(x)) - KL(q^*(z|x)\|p_d(z|x))$. Now, we have that

$$\begin{aligned}
L_d(q^*(d), d) &= \mathbb{E}_{p_{data}}\left[W(x; q^*(d), d)\right] + \gamma\mathbb{E}_{p_d}\left[W(x; q^*(d), d)\right] \\
&= \mathbb{E}_{p_{data}}\left[\log(p_d(x)) - KL(q^*(z|x)\|p_d(z|x))\right] + \gamma\mathbb{E}_{p_d}\left[\log(p_d(x)) - KL(q^*(z|x)\|p_d(z|x))\right] \\
&= -KL(p_{data}\|p_d) + \mathbb{E}_{p_{data}}\left[\log(p_{data}(x))\right] - \gamma H(p_d(x)) \\
&\quad - \mathbb{E}_{p_{data}}\left[KL(q^*(z|x)\|p_d(z|x))\right] - \gamma\mathbb{E}_{p_d}\left[KL(q^*(z|x)\|p_d(z|x))\right].
\end{aligned} \tag{17}$$

Since $KL(q^*(d)\|p_d(z|x)) \geq 0 = KL(q^*(d^*)\|p_{d^*}(z|x))$, and $p_{d^*} = \arg\min_{p_d}\left\{KL(p_{data}\|p_d) + \gamma H(p_d(x))\right\}$, we have that $d^* \in \arg\max_d L_d(q^*(d), d)$. Also, since $KL(q^*\|p_d(z|x)) \geq 0 = KL(q^*\|p_{d^*}(z|x))$, we have that $d^* \in \arg\max_d L_d(q^*, d)$. We conclude that (q^*, d^*) is a Nash equilibrium of equation 13. □

Theorem 3 assumes that $p_{d^*}(x) \leq p_{data}(x)^{\frac{1}{\alpha+1}}$ for all x. We now claim that for any p_{data}, there exists some $\gamma > 0$ such that this assumption holds.

Theorem 7. *For any p_{data}, there exists $\gamma > 0$ such that p_{d^*}, as defined defined in equation 16, satisfies Assumption 3.*

Proof. We will show that for $\gamma = 0$ the condition holds, and that p_{d^*} is continuous in γ.

Since $\alpha \geq 1$, for $\gamma = 0$ we have that $p_{d^*} = p_{data}$. Therefore $\frac{(p_{d^*}(x))^{\alpha+1}}{p_{data}(x)} = p_{d^*}^{\alpha}(x) \leq 1$.[6]

By Theorem 2 of Milgrom and Segal [42] (the Envelope theorem) we have that the value function $V(\gamma) = \min_d\left\{KL(p_{data}\|p_d) + \gamma H(p_d(x))\right\}$ is continuous in γ. Therefore, for every $\epsilon > 0$ there exists some γ for which $V(\gamma) - V(0) \leq \epsilon$, which yields

$$\begin{aligned}
&\min_d\left\{KL(p_{data}\|p_d) + \gamma H(p_d(x))\right\} - \min_d\left\{KL(p_{data}\|p_d)\right\} \\
&= \min_d\left\{KL(p_{data}\|p_d) + \gamma H(p_d(x))\right\} \leq \epsilon.
\end{aligned} \tag{18}$$

Let d^* satisfy $d^* \in \arg\min_d\left\{KL(p_{data}\|p_d) + \gamma H(p_d(x))\right\}$. Since the entropy H is non-negative, we have from equation 18 that

$$KL(p_{data}\|p_{d^*}) \leq \epsilon.$$

[6] The condition $p_d(x) \leq 1$ is obvious for discrete distributions. For continuous distributions, it is satisfied in a differential sense $p_d(x)dx \leq 1$, since $\int_x p_d(x)dx = 1$ and $p_d(x) \geq 0$.

Let $D(p_{data}\|p_{d^*}) = \sup_x |p_{data}(x) - p_{d^*}(x)|$ denote the total variation distance. From Pinsker's inequality we have that

$$D(p_{data}\|p_{d^*}) \leq \sqrt{0.5KL(p_{data}\|p_{d^*})} \leq \sqrt{0.5\epsilon}. \tag{19}$$

Choose ϵ such that

$$\sqrt{0.5\epsilon} \leq \min_{x:p_{data}(x)>0} \left\{ -p_{data}(x) + p_{data}(x)^{\frac{1}{\alpha+1}} \right\}, \tag{20}$$

and note that since $p_{data}(x) \leq 1$ and $\alpha \geq 1$, then $-p_{data}(x) + p_{data}(x)^{\frac{1}{\alpha+1}} \geq 0$, and therefore we can find an $\epsilon > 0$ that satisfies equation 20. We thus have that for any x such that $p_{data}(x) > 0$:

$$p_{d^*}(x) \leq p_{data}(x) + \sqrt{0.5\epsilon} \leq p_{data}(x)^{\frac{1}{\alpha+1}}, \tag{21}$$

where the first inequality is from the definition of the total variation distance and equation 19, and the second inequality is by equation 20. □