A DERIVATION OF THE LOSS FUNCTION

In this section, we describe the details of our VAELLS implementation. We define the loss function E to be minimized as the approximate negative ELBO in (9), restated here for convenience:

$$L_x(z) \equiv \log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x)$$
$$E(x) \equiv -\mathbb{E}_{u,\varepsilon}\left[L_x(T_\Psi(l(u;b))f_\phi(x) + \gamma\varepsilon)\right] + \frac{\eta}{2}\sum_{m=1}^{M}\|\Psi_m\|_F^2$$
$$\varepsilon \sim \mathcal{N}(0, I) \quad u \sim \text{ Unif}\left(-\frac{1}{2}, \frac{1}{2}\right)^M.$$

In practice, when optimizing this loss function we approximate it with a set of N_s samples:

$$\widehat{E}(x) \equiv -\frac{1}{N_s}\sum_{s=1}^{N_s} L_x(T_\Psi(l(u_s;b))f_\phi(x) + \gamma\varepsilon_s) + \frac{\eta}{2}\sum_{m=1}^{M}\|\Psi_m\|_F^2 \tag{10}$$
$$\varepsilon_s \sim \mathcal{N}(0, I) \quad u_s \sim \text{ Unif}\left(-\frac{1}{2}, \frac{1}{2}\right)^M.$$

The deterministic mapping $l(u;b)$ is an inverse transform that maps independent uniform variates to the following factorial Laplace distribution:

$$q(c) = \prod_{m=1}^{M} q(c_m),$$

where for $b > 0$,

$$q(c_m) = \frac{1}{2b}\exp(-\frac{|c_m|}{b}).$$

Specifically, we sample independently from each marginal $q(c_m)$ by defining the mth element of $l(u;b)$ as follows:

$$l_m(u;b) = -b\,\text{sgn}(u_m)\log(1 - 2\,|u_m|).$$

Next we derive expressions for each expanded term in (10). For the likelihood term, we have

$$\begin{aligned} -\log p_\theta(x \mid z) &= -\log\left[(2\pi)^{\frac{-D}{2}}\sigma^{-D}\exp\left(-\frac{\|x - g_\theta(z)\|_2^2}{2\sigma^2}\right)\right] \\ &= C_1 + \zeta_1\|x - g_\theta(z)\|_2^2, \end{aligned}$$

where $\zeta_1 = 2^{-1}\sigma^{-2}$ is treated as a hyperparameter and $C_1 = \frac{D}{2}\log(2\pi) + D\log\sigma$. For the variational posterior term, we have

$$\log q_\phi(z \mid x) = \log\int_c q_\phi(z, c \mid x)dc \tag{11}$$
$$\approx \max_c \log\left[q_\phi(z \mid c, x)q(c)\right], \tag{12}$$

where the approximation in (12) is motivated by the fact that the sparsity-inducing Laplace prior on c typically results in joint distributions with z that are tightly peaked in the coefficient space, as described in Olshausen and Field (1997). Due to this tight peak in $q_\phi(z, c \mid x)$, we can approximate the integral in (11) by approximating the volume under the surface in a small neighborhood around the density maximizer. Specifically, we approximate this volume by evaluating the joint density at it's maximum, weighted by the volume of a ball with a small radius centered at the maximizer. This volume approximation results in an additive constant in the log posterior that reflects the area of the chosen neighborhood, which we omit from this log-likelihood computation.

We have

$$\log\left[q_\phi(z \mid c, x)q(c)\right] = C_2 - \zeta_2\|z - T_\Psi(c)f_\phi(x)\|_2^2 - \zeta_3\sum_{m=1}^{M}|c_m|,$$

where $\zeta_2 = 2^{-1}\gamma^{-2}$ and $\zeta_3 = b^{-1}$ are treated as a hyperparameters and $C_2 = -\frac{d}{2}\log(2\pi) - d\log\gamma + M\log\frac{1}{2b}$. Using the notation

$$c^*(z, x; \zeta) = \underset{c}{\operatorname{argmin}} \left[\zeta_2 \left\| z - T_\Psi(c) f_\phi(x) \right\|_2^2 + \zeta \sum_{m=1}^{M} |c_m| \right], \tag{13}$$

we have (with hyperparameter ζ_q)

$$\log q_\phi(z \mid x) \approx C_2 - \zeta_2 \left\| z - T_\Psi(c^*(z, x; \zeta_q)) f_\phi(x) \right\|_2^2 - \zeta_3 \sum_{m=1}^{M} \left| c_m^*(z, x; \zeta_q) \right|. \tag{14}$$

Finally, for the prior distribution (with hyperparameter ζ_p) we have

$$\begin{aligned} -\log p_\theta(z) &= -\log \frac{1}{N_a} \sum_{i=1}^{N_a} q_\phi(z \mid a_i) \\ &\approx \log N_a - C_2 - \log \sum_{i=1}^{N_a} \exp\left(-\zeta_2 \left\| z - T_\Psi(c^*(z, a_i; \zeta_p)) f_\phi(a_i) \right\|_2^2 - \zeta_3 \sum_{m=1}^{M} \left| c_m^*(z, a_i; \zeta_p) \right| \right), \end{aligned}$$

where we use the same approximation as (12).

All together, dropping additive constants and letting $z_s = T_\Psi(l(u_s)) f_\phi(x) + \gamma\varepsilon_s$, we have

$$\begin{aligned} \widehat{E}(x) = \frac{1}{N_s} \sum_{s=1}^{N_s} \zeta_1 \left\| x - g_\theta(z_s) \right\|_2^2 - \zeta_2 \left\| z_s - T_\Psi(c^*(z_s, x; \zeta_q)) f_\phi(x) \right\|_2^2 - \zeta_3 \sum_{m=1}^{M} \left| c_m^*(z_s, x; \zeta_q) \right| \\ - \log \sum_{i=1}^{N_a} \exp\left(-\zeta_2 \left\| z_s - T_\Psi(c^*(z_s, a_i; \zeta_p)) f_\phi(a_i) \right\|_2^2 - \zeta_3 \sum_{m=1}^{M} \left| c_m^*(z_s, a_i; \zeta_p) \right| \right) + \frac{\eta}{2} \sum_{m=1}^{M} \left\| \Psi_m \right\|_F^2 \end{aligned}$$

In practice, one may wish to construct the prior with different constants than those used for the variational posterior term (i.e., constants ζ_4, ζ_5 instead of ζ_2, ζ_3). This substitution results in the final VAELLS objective

$$\begin{aligned} \widehat{E}(x) = \frac{1}{N_s} \sum_{s=1}^{N_s} \zeta_1 \left\| x - g_\theta(z_s) \right\|_2^2 - \zeta_2 \left\| z_s - T_\Psi(c^*(z_s, x; \zeta_q)) f_\phi(x) \right\|_2^2 - \zeta_3 \sum_{m=1}^{M} \left| c_m^*(z_s, x; \zeta_q) \right| \\ - \log \sum_{i=1}^{N_a} \exp\left(-\zeta_4 \left\| z_s - T_\Psi(c^*(z_s, a_i; \zeta_p)) f_\phi(a_i) \right\|_2^2 - \zeta_5 \sum_{m=1}^{M} \left| c_m^*(z_s, a_i; \zeta_p) \right| \right) + \frac{\eta}{2} \sum_{m=1}^{M} \left\| \Psi_m \right\|_F^2 \end{aligned} \tag{15}$$

We allow the user to tune all of the hyperparameters during training.

B DETAILS ON VAELLS TRAINING PROCEDURE

In this section we provide additional details on the general training procedure for all of the experiments. Algorithm 1 provides a detailed view of the VAELLS training steps. For specific architecture details and parameter selections, see each experiment's respective Appendix section.

Network training To enhance the generative capability of the decoder, in some experiments, we use a warm-up as in Tomczak and Welling (2018). Our warm-up includes updates to the network weights driven by only the reconstruction loss. During warm up there is no Gaussian sampling in the latent space and the sampled Laplace distribution for the transport operator coefficients has a large ζ_3 parameter which encourages sampling coefficients that are very close to zero . As mentioned in Section 4, during VAELLS training we alternate between steps where we update the network weights and anchor points while keeping the transport operators fixed and steps where we update the transport operators while keeping the network weights and anchor points fixed. As we alternate between these steps, we vary the weights on the objective function terms. Specifically, we decrease the importance of the prior terms during the steps updating the network weights and decrease the importance of the reconstruction term during the steps updating the transport operators.

Transport operator learning As mentioned in Section 4, one component of computing the prior and posterior objective is the coefficient inference between the sampled point z and the neural network encoding $f_\phi(x)$ as well as

all the encoded anchor points $f_\phi(a_i)$. Note that the transport operator objective is non-convex which may result in coefficient inference optimization arriving at a poor local minima. This issue can be avoided by performing the coefficient inference between the same point pair several times with different random initializations of the coefficients and selecting the inferred coefficients that result in the lowest final objective function. We leave the number of random initializations as a parameter to select during training.

Transport operator coefficient inference is best performed when the magnitude of the latent vector entries is close to the range $[-1, 1]$. Because of this, we allow for the selection of a scale factor that scales the latent vectors prior to performing coefficient inference. In practice, we inspect the magnitude of the latent vectors after warm-up training steps and select a scale factor that adapts the largest magnitudes of the latent vector entries to around 1.

During every transport operator update step, our optimization routine checks whether the transport operator update improves the portion of the objective that explicitly incorporates the transport operator:

$$\begin{aligned}\widehat{E}_{\text{transopt}}(x) = \frac{1}{N_s}\sum_{s=1}^{N_s} -\zeta_2 \left\|z_s - T_\Psi(c^*(z_s, x; \zeta_q))f_\phi(x)\right\|_2^2 - \zeta_3 \sum_{m=1}^{M} |c_m^*(z_s, x; \zeta_q)| \\ - \log \sum_{i=1}^{N_a} \exp\left(-\zeta_4 \left\|z_s - T_\Psi(c^*(z_s, a_i; \zeta_p))f_\phi(a_i)\right\|_2^2 - \zeta_5 \sum_{m=1}^{M} |c_m^*(z_s, a_i; \zeta_p)|\right)\end{aligned} \tag{16}$$

If this portion of the objective does not improve with a gradient step on the dictionary then we reject this step and decrease the transport operator learning rate. If the transport operator portion of the objective does improve with the gradient step on the dictionary then we accept this step and increase the transport operator learning rate. This helps us settle on an appropriate learning rate and prevents us from making ineffective updates to the dictionaries. We also set a maximum transport operator learning rate which varies based on the experiment.

Another unique consideration during transport operator training is that we generally assume that the transport operator training points are close on the manifold. In the formulation of the variational posterior, this is a reasonable assumption because z is a sample originating from $f_\phi(x)$. However, in the prior formulation, while the anchor points are generally sampled in a way that encourages them to be evenly spaced in the data space, it is unlikely that every latent vector associated with a data point is close to every anchor point. In order to aid in constraining training to points that are relatively close on the manifold, we provide the training option of defining the prior with respect to only the anchor point closest to z rather than summing over all the anchor points:

$$p_\theta(z) = q_\phi(z \mid a^*), \tag{17}$$

where a^* is the anchor that is estimated to be closest to z. Since we do not have ground truth knowledge of which anchor point is closest to a given training point on the data manifold, we estimate this by inferring the coefficients that represent the estimated path between z and every a_i. We then select a^* as the anchor point with the lowest objective function (i.e., $\zeta_4 \left\|z_s - T_\Psi(c^*(z_s, a_i; \zeta_p))f_\phi(a_i)\right\|_2^2 + \zeta_5 \sum_{m=1}^{M} |c_m^*(z_s, a_i; \zeta_p)|$) after coefficient inference. This objective function defines how well a_i can be transformed to z_s using the current transport operator dictionary elements Ψ.

There are several hyperparameters that need to be tuned in this model. However, we have found through experimentation that the model is robust to changes in several of the parameters (such as ζ_2, ζ_3,ζ_4). The hyperparameters that were shown to have the largest effect on training effectiveness were:

- The weight on the reconstruction term (ζ_1) - use this in combination with warm-up steps to ensure reasonable reconstruction accuracy from the decoder.
- The posterior coefficient inference weight (ζ_q)- this is the weight on the sparsity regularizer term used in the objective (13) during coefficient inference between points in the *posterior* term. If this weight is too large, then inference can result in zero coefficients for all the operators which is not informative.
- The prior coefficient inference weight (ζ_p) - this is the weight on the sparsity regularizer term used during coefficient inference between points in the *prior* term.
- Number of restarts used during coefficient inference for transport operator training.

- Starting lr_ψ - As mentioned above, we do vary lr_ψ during transport operator training depending on whether our training steps are successful or not. If this learning rate starts too high, it can result in many unsuccessful steps with no updates on the transport operator dictionaries which greatly slows down training.

Algorithm 1: Training of network weights, transport operators, and anchor points

Data: Training samples $\mathcal{X}$, anchor points $\{a_1, ..., a_{N_a}\}$ selected from the input space
Result: Trained transport operator dictionary elements $\{\Psi_1, ...\Psi_M\}$, network weights ϕ and θ, fine-tuned anchor points $\{a_1, ..., a_{N_a}\}$

Ψ, ϕ, $\theta \leftarrow$ randomly initialize dictionaries, network weights;
for $k = 0, ..., N$ **do**
  Sample mini-batch from $\mathcal{X}$: $\{x_1, ..., x_{N_s}\}$;
  for $s = 0, ..., N_s$ **do**
    Encode samples: $\mu_s \leftarrow f_\phi(x_s)$;
    Sample $u_s \sim$ Unif $\left(-\frac{1}{2}, \frac{1}{2}\right)^M$;
    $\widehat{c}_s \leftarrow l(u_s)$;
    Sample z_s from $q_\phi(z_s \mid \widehat{c}_s, x_s) \sim T_\Psi(\widehat{c}_s)\mu_s + \gamma\varepsilon_s$;
    Decode sampled vectors: $\widehat{x}_s \leftarrow g_\theta(z_s)$;
    $c^*(z_s, x_s) \leftarrow$ Infer_Coefficients$(z_s, x_s, d_{\text{start}}, d_{\text{stop}})$;
    for $i = 0, ..., N_a$ **do**
      for $r = 0, ..., \text{num\_restart}$ **do**
        $c^{(r)}(z_s, a_i) \leftarrow$ Infer_Coefficients$(z_s, a_i, d_{\text{start}}, d_{\text{stop}})$;
        $E_c(c^{(r)}(z_s, a_i)) \leftarrow \log\left[q_\phi(z_s \mid c^{(r)}(z_s, a_i), a_i) q(c^{(r)}(z_s, a_i))\right]$
      end
      $c^*(z_s, a_i) \leftarrow \text{argmax}_r\, E_c(c^{(r)}(z_s, a_i))$
    end
  end
  Calculate $\widehat{E}$ in (15) using $\widehat{x}_s$, $c^*(z_s, x_s)$, and $c^*(z_s, a_i)$ for $s = 0, ..., N_s$ and $i = 0, ..., N_a$;
  $\Psi_{\text{new}} \leftarrow \Psi - lr_\Psi \frac{\delta \widehat{E}}{\delta \Psi}$;
  if $\widehat{E}_{\text{transopt}}(\Psi_{\text{new}}) < \widehat{E}_{\text{transopt}}(\Psi)$ **then**
    $\Psi \leftarrow \Psi_{\text{new}}$;
    $lr_\Psi \leftarrow \frac{lr_\Psi}{\text{decay}}$
  else
    $lr_\Psi \leftarrow lr_\Psi \cdot \text{decay}$
  end
  $\phi \leftarrow \phi - lr_{\text{net}} \frac{\delta \widehat{E}}{\delta \phi}$;
  $\theta \leftarrow \theta - lr_{\text{net}} \frac{\delta \widehat{E}}{\delta \theta}$;
  $a \leftarrow a - lr_{\text{anchor}} \frac{\delta \widehat{E}}{\delta a}$;
end

Algorithm 2: Infer_Coefficients$(z, x, d_{\text{start}}, d_{\text{stop}})$

Data: Latent vector z, input data x
Result: Inferred coefficient vector c
Initialize c: $c_m \sim \text{Unif}[d_{\text{start}}, d_{\text{stop}}]$;
Fix c to $c^* \leftarrow \text{argmax}_c \log\left[q_\phi(z \mid c, x) q(c)\right]$;

Comparison techniques We implemented the hyperspherical VAE (Davidson et al., 2018) using the code provided by the authors: `https://github.com/nicola-decao/s-vae-pytorch`. For the concentric circle and swiss roll experiments we used the network specified in Table 3. For MNIST experiments, we used the network architecture from their MNIST experiments, and we dynamically binarized the MNIST inputs as they did. We implemented the VAE with VampPrior (Tomczak and Welling, 2018) model using the code provided by the authors: `https://github.com/jmtomczak/vae_vampprior`. For the concentric circle and swiss roll experiments,

we used the network architecture specified in Table 3 and 100 pseudoinputs for each experiment. For the MNIST experiments we adapted the network in Table 6 to add a linear layer between the final convTranspose layer and the sigmoid layer. We used 500 pseudoinputs in each of the MNIST experiments. The VAE with VampPrior MNIST tests were also performed on dynamically binarized MNIST data. We implemented our own VAE code with the same network architectures detailed in Tables 3 and 6.

C VARIATIONAL POSTERIOR CONTOURS

In order to more intuitively visualize the learned variational posterior we show contour plots of the variational posterior given an input point. The variational posterior (14) consists of two terms: the data fidelity term $\left(-\zeta_2 \left\| z - T_\Psi(c^*(z,x;\zeta_q))f_\phi(x)\right\|_2^2\right)$ and the coefficient prior term $\left(-\zeta_3 \sum_{m=1}^M |c_m^*(z,x;\zeta_q)|\right)$. The data fidelity term expresses the probability of z given an input point x. Fig. 8a shows a contour of the data fidelity term. The red x is the location of the encoded point and the black dots are encoded data samples. This shows how the data fidelity term maps out the contour of the latent swiss roll manifold around the encoded point. The coefficient prior is a Laplace distribution which encourages the coefficients to be tightly peaked around zero. Fig. 8b shows the contour of the prior term. There is a high log probability in the slice of the space that can be reached by applying small coefficients to the transport operators and the log probability reduces in parts of the space that require larger coefficient values to reach.

The final variational posterior in Fig. 8c is a combination of these two terms. In our experimental setting, the data fidelity term is the dominant term that characterizes the variational posterior. However, as the weights ζ_2 and ζ_3 vary, the contribution of each term towards the variational posterior will change.

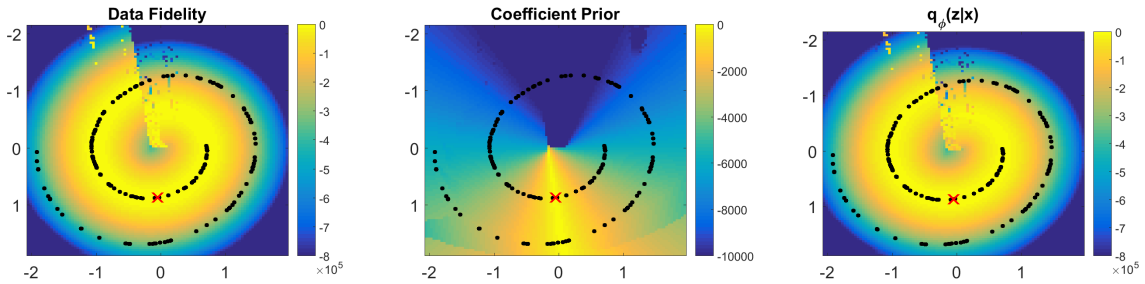


Figure 8: Variational posterior contour plots. The red x is the location of the encoded point z. The black dots are encoded data points. (a) Contour of data fidelity term (b) Contour of coefficient prior term (c) Contour of $\log q_\phi(z \mid x)$

D ESTIMATED LOG-LIKELIHOOD

We estimate the log-likelihood using the importance sampling from Burda et al. (2015) with 500 datapoints and 100 samples per datapoint. We follow the derivation from Appendix A for computing $\log p_\theta(x \mid z)$, $\log q_\phi(z \mid x)$, and $\log p_\theta(z)$ and include all the constants defined there. We set ζ_1 to the same value used during training. While we hand-select ζ_2, ζ_3, ζ_4, and ζ_5 during training, when computing the estimated log-likelihood, we compute each of these values based on the parameters γ and b which are used for sampling the latent vectors as specified in (5). To be explicit, $\zeta_2 = \zeta_4 = 2^{-1}\gamma^{-2}$ and $\zeta_3 = \zeta_5 = b^{-1}$. When we infer coefficeints as in (13), we set $\zeta_2 = 1$ and we set the hyperparameters ζ_q and ζ_p (which are used for the posterior and prior respectively) to the same values used during training which yield successful coefficient inference. As noted by (12), the approximation of $\log \int_c q_\phi(z, c \mid x)dc$ will result in an additive constant that we do not include in our estimated log-likelihood computation.

The results presented in Table 2 show a wide variation in estimated log-likelihood over the trials. These values are largely dominated by the data fidelity term in the prior which decreases significantly as the number of dictionaries increase from 2 to 8. Because this experiment sets $\gamma = 0.001$, any variation in the data fidelity terms for the $\log q_\phi(z \mid x)$ and $\log p_\theta(z)$ are magnified when they are multiplied by $\zeta_2 = \zeta_4 = \frac{1}{2(0.001^2)}$.

Table 3: Network Architecture for Swiss Roll and Concentric Circle Experiments

Encoder Network	Decoder Network
Input $\in \mathbb{R}^{20}$	Input $\in \mathbb{R}^2$
Linear: 512 Units	Linear: 512 Units
ReLU	ReLU
Linear: 2 Units	Linear: 20 Units

Table 4: Training Parameters for Swiss Roll Experiment

VAELLS Training - Swiss Roll
batch size: 30
training steps: 3000
latent space dimension (z_{dim}): 2
N_s : 1
lr_{net} : 10^{-4}
lr_{anchor} : 10^{-4}
starting lr_{Ψ} : 5×10^{-5}
max lr_{Ψ} : 0.05
ζ_1 : 0.01
ζ_2 : 1
ζ_3 : 1
ζ_4 : 1
ζ_5 : 0.01
ζ_q: 1×10^{-6}
ζ_p: 5×10^{-5}
η : 0.01
number of network and anchor update steps: 20
weight on prior terms during net update steps: 0.01
number of Ψ update steps: 20
weight on recon term during net update steps: 0.001
γ_{post} : 0.001
warm-up steps: 0
number of restarts for coefficient inference: 2
M : 1
number of anchors: 4
latent space scale: 1
define prior with respect to closest anchor point: yes

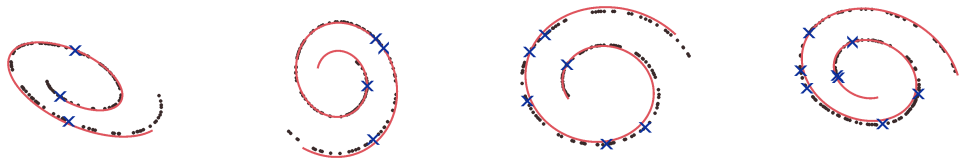
E SWISS ROLL EXPERIMENT

Tables 3 and 4 contain the VAELLS network architecture and parameters for the swiss roll experiment. In this experiment, we sample our ground truth 2D data manifold from a swiss roll and then map it to the 20-dimensional input space using a random linear mapping. We use 1000 swiss roll training points and randomly sample swiss roll test points. We initialize anchor points as points that are spaced out around the swiss roll prior to mapping to the 20-dimensional input space. We allow for the anchor points to be updated; however, these updates result in negligible changes to the anchor points. As described in Section B, in this experiment we define the prior only with respect to the anchor points that are estimated to be closest to each training point.

F ANALYSIS OF SENSITIVITY TO ANCHOR POINTS

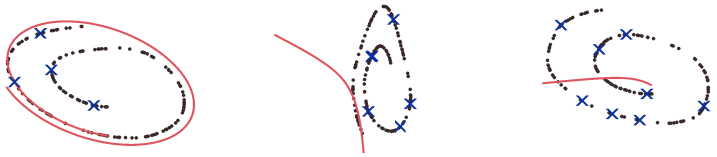
The anchor points are a key feature needed to specify the learned latent prior and its important to understand the role that anchor point selection plays in the success of learning the data manifold. We investigate this question on the swiss roll manifold with a known latent structure that will allow us to determine the success of the VAELLS training procedure as we vary the anchor points.

We begin by varying the number and location of anchor points. In the initial formulation of the swiss roll experiment, the anchor points are selected to be well distributed around the swiss roll manifold. Fig. 3b shows the locations of the anchor points used for experiment detailed in the paper. The anchor points are well spaced around the swiss roll manifold. Fig. 3a shows the learned operator from this experiment which successfully generates paths with the desired swiss roll manifold structure. While the experiment we presented uses four well-spaced anchor points, we can vary both the number and locations of the anchor points and still learn a mapping to a swiss roll latent structure and a transport operator that generates paths along the swiss roll. Fig. 9 shows examples of tests where VAELLS successfully learns the swiss roll structure in the latent space with different numbers of anchor points. While we achieve success with randomly positioned anchor points, there are tests where the anchor points are relatively evenly spaced over the manifold and the transport operators do not learn the swiss roll structure. Fig. 10 shows examples of tests where VAELLS fails to learn the swiss roll manifold even when the anchor points are relatively evenly spaced on the encoded manifold. These results indicate that the success of learning transport operators to represent the true latent manifold depends on a factor other than the anchor point locations.



(a) (b) (c) (d)

Figure 9: Outputs of trials where VAELLS learns the correct swiss roll manifold while the number of anchor points is varied and the anchor point locations are randomly selected on the manifold. The black dots are encoded data points, the blue x's are the encoded anchor point locations, and the red line is the generated orbit of the learned transport operator. Each plot shows a result with a different number of anchor points: (a) 3 anchor points (b) 4 anchor points (c) 6 anchor points (d) 8 anchor points.

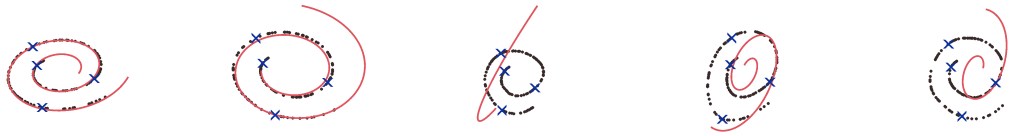


(a) (b) (c)

Figure 10: Outputs of trials where VAELLS fails to learn the correct swiss roll manifold while the number of anchor points is varied and the anchor point locations are randomly selected on the manifold. The black dots are encoded data points, the blue x's are the encoded anchor point locations, and the red line is the generated orbit of the learned transport operator. Each plot shows a result with a different number of anchor points: (a) 4 anchor points (b) 6 anchor points (c) 8 anchor points.

Another possible factor important for successful learning of the latent manifold is the initialization of the dictionary elements. To analyze the impact of dictionary initialization, we train 10 separate instances of VAELLS which we initialize with the same network weights and anchor point locations and a different dictionary element weights. Fig. 11 shows the final training outputs of five of these trials. The trials shown in Fig. 11(a-b) successfully learn a transport operator that traverses a swiss roll manifold and the trials shown in Fig. 11(c-e) fail to learn a transport operator that traverses the swiss roll manifold. This experiment shows that the final transport operator orbits and latent space encoding can vary significantly with different initializations of the dictionary element and this indicates that the dictionary initialization is a more important factor for successful training than the anchor point locations.

The transport operator objective is a non-convex optimization surface which can result in the optimization settling in local minima that do not represent the true data manifold. There are a few cues to observe when determining whether a training run is successfully learning the data manifold. First, as mentioned in Appendix B, during



(a) (b) (c) (d) (e)

Figure 11: Outputs of trials that are initialized with the same network weights and anchor point locations and different dictionary weights. The black dots are encoded data points, the blue x's are the encoded anchor point locations, and the red line is the generated orbit of the learned transport operator.(a-b) Trials that successfully learn a transport operator that traverses the swiss roll structure. (c-e) Trials that fail to learn a transport operator that traverses the swiss roll structure.

every transport operator update step, we check whether the update improves the portion of the objective that incorporates the transport operator. If this portion of the objective does not improve with a gradient step on the dictionary, we reject the step and decrease the learning rate. Often, if the initialized dictionary does not yield effective learning of the data manifold, then we see a large number of the gradient steps that do not improve the transport operator portion of the objective and those steps are rejected. Therefore, examining the number of rejected steps can be an indicator of the effectiveness of transport operator training. Another indicator for poor performance is the amount of time it takes to perform coefficient inference when computing the prior. As the transport operator model becomes a better match for the latent manifold, the coefficient inference will require fewer steps which will reduce the amount of inference time. When comparing a VAELLS model that learns to match the data manifold to one that is a poor match, the inference time is often noticeably lower for the successful model.

G CONCENTRIC CIRCLE EXPERIMENT

The concentric circle experiment uses the same network architecture as the swiss roll experiment which is specified in Table 3. Table 5 shows the training parameters for the concentric circle experiment. In this experiment, we sample our ground truth 2D data manifold from two concentric circles and then map those sampled points to the 20-dimensional input space using a random linear mapping. We use 400 training points and randomly sample test points.

It should be noted that, in this experiment, we do not alternate between steps where we update the network weights and anchor points while fixing the transport operator weights and steps where we update the transport operator weights while keeping the network weights and anchor points fixed. Instead the network weights, anchor points, and transport operator weights are all updated simultaneously. We use three anchor points per circular manifold and initialize them by evenly spacing them around each circle prior to mapping into the 20-dimensional input space. While the anchor points are allowed to update during training, the changes in the anchor points are negligible. For each input point, the prior is computed as a sum over the variational posterior conditioned on only the anchor points on the same circle as the input point.

Fig. 12 shows the encoded latent points overlaid with the orbits of the learned transport operators. These orbits are generated by selecting one point on each circular manifold and applying a single operator as its trajectory evolves over time. Notice that one of the operators clearly represents the circular structure of the latent space while the other three have much smaller magnitudes and a limited effect on the latent space transformations. The Frobenius norm regularizer in the objective function often aids in model order selection by reducing the magnitudes of operators that are not used to represent transformations between points on the manifold. To see the magnitudes more clearly, Fig. 13 shows the magnitude of each of the transport operators after training the VAELLS model in the concentric circle test case.

Fig. 14a shows latent points sampled from the prior using the sampling described in (5) with larger standard deviation and scale parameters to aid in visualization. Fig. 14(b-c) show two example inferred paths between points encoded on the concentric circle manifold.

Table 5: Training Parameters for Concentric Circle Experiment

VAELLS Training - Concentric Circle
batch size: 30
training steps: 4000
latent space dimension (z_{dim}): 2
N_s : 1
lr_{net} : 0.005
lr_{anchor} : 0.0001
starting lr_Ψ : 4×10^{-4}
max lr_Ψ : 0.1
ζ_1 : 0.01
ζ_2 : 1
ζ_3 : 1
ζ_4 : 1
ζ_5 : 0.01
ζ_q: 1×10^{-6}
ζ_p: 5×10^{-6}
η : 0.01
number of network and updates steps: N/A
number of Ψ update steps: N/A
γ_{post} : 0.001
warm-up steps: 0
number of restarts for coefficient inference: 1
M : 4
number of anchors per class: 3
latent space scale: 1
define prior with respect to closest anchor point: no

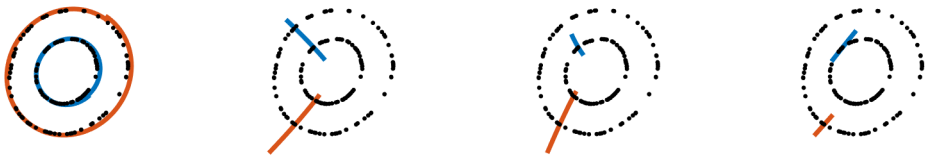


Figure 12: The orbits of each of the transport operators learned in the concentric circle test case plotted on top of encoded points.

H ROTATED MNIST EXPERIMENT

We split the MNIST dataset into training, validation, and testing sets. The training set contains 50,000 images from the traditional MNIST training set. The validation set is made up of the remaining 10,000 image from the traditional MNIST training set. We use the traditional MNIST testing set for our testing set. The input images are normalized by 255 to keep the pixel values between 0 and 1. To generate a batch of rotated MNIST digits, we randomly select points from the MNIST training set and rotate those images to a random angle between 0 and 350 degrees. Separate anchor points are selected for each training example. Anchor points are generated by rotating the original MNIST sample by angles that are evenly spaced between 0 and 360 degrees. Because we have separate anchor points for each training sample, they are not updated during training. Tables 6 and 7 contain the VAELLS network architecture and parameters for the rotated MNIST experiment. As described in Appendix B, in this experiment, we define the prior only with respect to the anchor points that are estimated to be closest to each training point. Fig. 15 shows more examples of images decoded from latent vectors sampled from the posterior of the models trained on rotated MNIST digits.

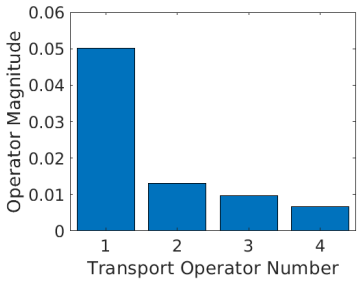
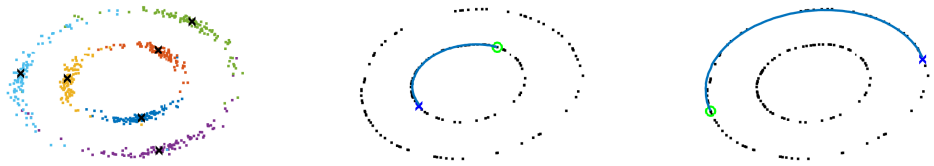


Figure 13: Magnitude of the operators after training in the 2D concentric circle experiment.



(a) (b) (c)

Figure 14: (a) Latent points sampled from the anchor points showing the concentric circle structure learned by the VAELLS model. (b-c) Transport operator paths inferred between points on the same concentric circle manifold.

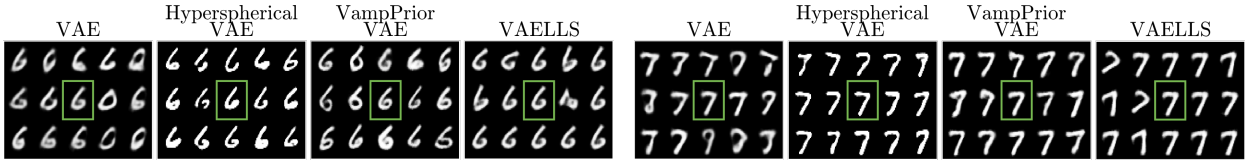
I NATURAL MNIST EXPERIMENT

This experiment uses the same training/validation/testing separation as described in the rotated MNIST experiment in Section H. The only pre-processing step for the digit images is normalizing by 255. Our qualitative results use a network that is trained with eight anchor points per digit class. These anchor points are initialized by randomly sampling eight examples of each class at the beginning of training. The anchor points are allowed to update during training but the changes in the anchor points during training are negligible. We use the same network architecture as in the rotated MNIST experiment (shown in Table 6). Table 8 shows the training parameters for this experiment. As described in Appendix B, in this experiment, we define the prior only with respect to the anchor points that are estimated to be closest to each training point.

Fig. 16 shows more examples of images decoded from latent vectors sampled from the posterior of the models trained on MNIST digits. Fig. 17 and Fig. 18 show the effect that each of the eight learned transport operators has on digits. To generate each figure, input images randomly selected from each class are encoded into the latent space and a single learned operator is applied to each of those latent vectors. The decoded version of the input image is shown in the middle column (in a green box). The images to the left of the middle column show the result of applying the operator with a negative coefficient and the images to the right of the middle column show the result of applying the operator with a positive coefficient.

Table 6: Network Architecture for MNIST Experiments

Encoder Network	Decoder Network
Input $\in \mathbb{R}^{28\times 28}$	Input $\in \mathbb{R}^2$
conv: chan: 64 , kern: 4, stride: 2, pad: 1	Linear: 3136 Units
ReLU	ReLU
conv: chan: 64, kern: 4, stride: 2, pad: 1	convTranpose: chan: 64, kern: 4, stride: 1, pad: 1
ReLU	ReLU
conv: chan: 64, kern: 4, stride: 1, pad: 0	convTranpose: chann: 64, kern: 4, stride: 2, pad: 2
ReLU	ReLU
Linear: 2 Units	convTranpose: chan: 1, kernel: 4, stride: 2, pad: 1
	Sigmoid



(a) (b)

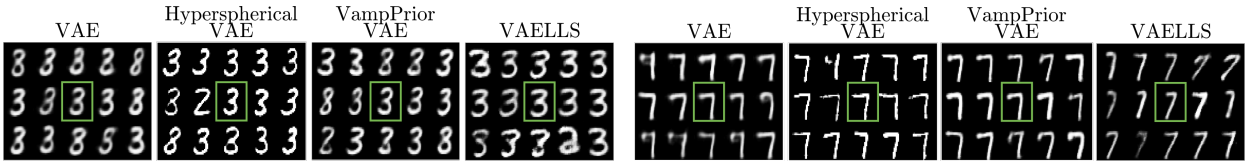
Figure 15: Examples of images decoded from latent vectors sampled from the posterior of models trained on rotated MNIST digits. In each example, the center digit (in the green box) is the decoded version of the input digit and the surrounding digits are images decoded from the sampled latent vectors. Sampling in the VAELLS latent space results in rotations in the sampled outputs.

Table 7: Training Parameters for Rotated MNIST Experiment

VAELLS Training - Rotated MNIST
batch size: 32
training steps: 35000
latent space dimension (z_{dim}): 10
N_s : 1
lr_{net} : 10^{-4}
lr_{anchor} : N/A
starting lr_Ψ : 1×10^{-5}
max lr_Ψ : 0.008
ζ_1 : 1
ζ_2 : 1
ζ_3 : 1
ζ_4 : 1
ζ_5 : 0.01
ζ_q: 1×10^{-6}
ζ_p: 1×10^{-6}
η : 0.01
number of network and anchor update steps: 20
weight on prior terms during net update steps: 0.0001
number of Ψ update steps: 60
weight on recon term during net update steps: 0.0001
γ_{post} : 0.001
warm-up steps: 30000
number of restarts for coefficient inference: 1
M : 1
number of anchors per class: 10
latent space scale: 10
define prior with respect to closest anchor point: yes

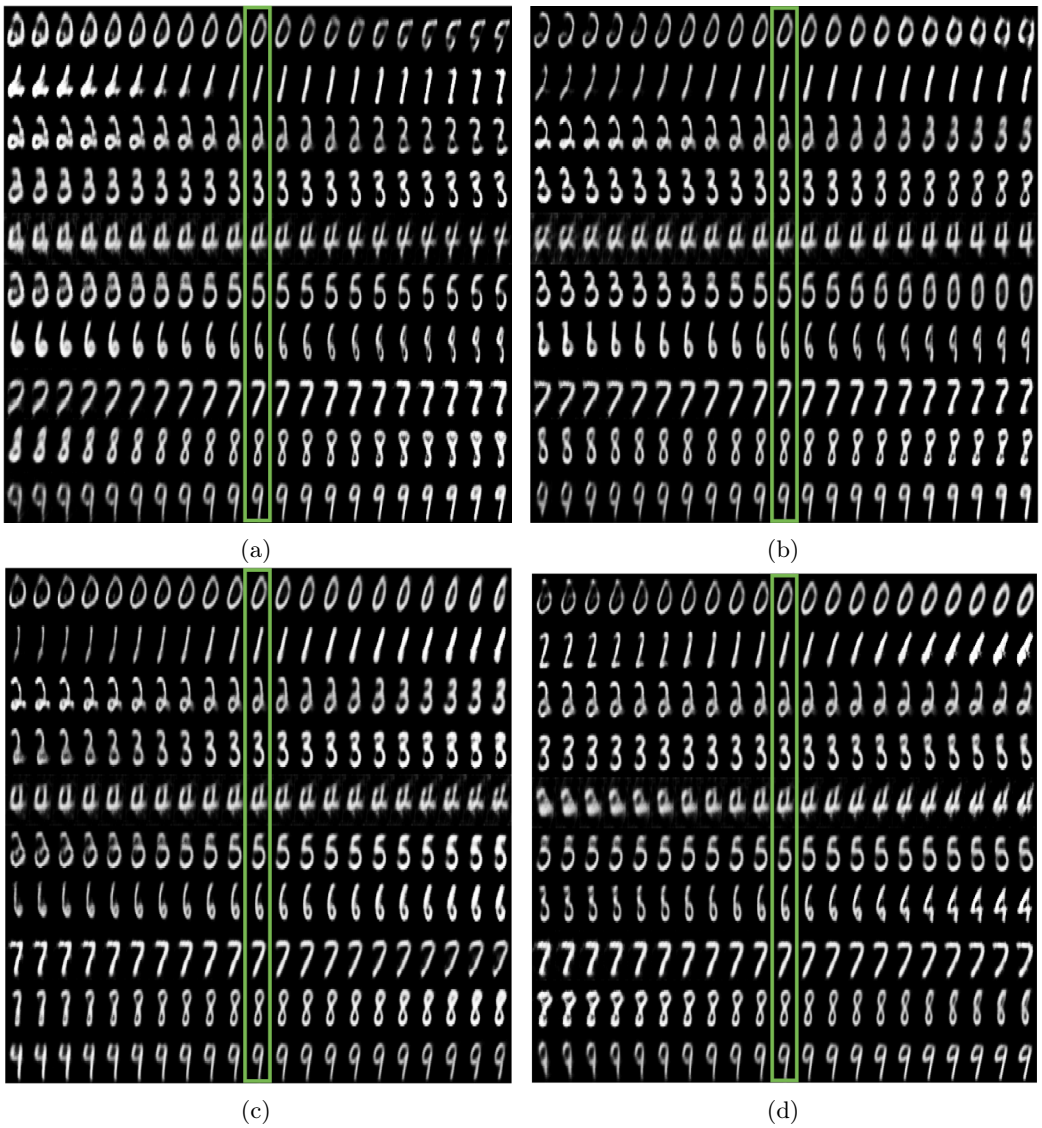
Table 8: Training Parameters for Natural MNIST Experiment

VAELLS Training - MNIST
batch size: 32
training steps: 34600
latent space dimension (z_{dim}): 6
N_s : 1
lr_{net} : $10^{-4}1$
lr_{anchor} : 10^{-4}
starting lr_{Ψ} : 1×10^{-5}
max lr_{Ψ} : 0.008
ζ_1 : 1
ζ_2 : 1
ζ_3 : 1
ζ_4 : 1
ζ_5 : 0.01
ζ_q: 1×10^{-6}
ζ_p: 1×10^{-6}
η : 0.01
number of network and anchor update steps: 20
weight on prior terms during net update steps: 0.0001
number of Ψ update steps: 60
weight on recon term during net update steps: 0.0001
γ_{post} : 0.001
warm-up steps: 30000
number of restarts for coefficient inference: 1
M : 4
number of anchors per class: 8
latent space scale: 10
define prior with respect to closest anchor point: yes



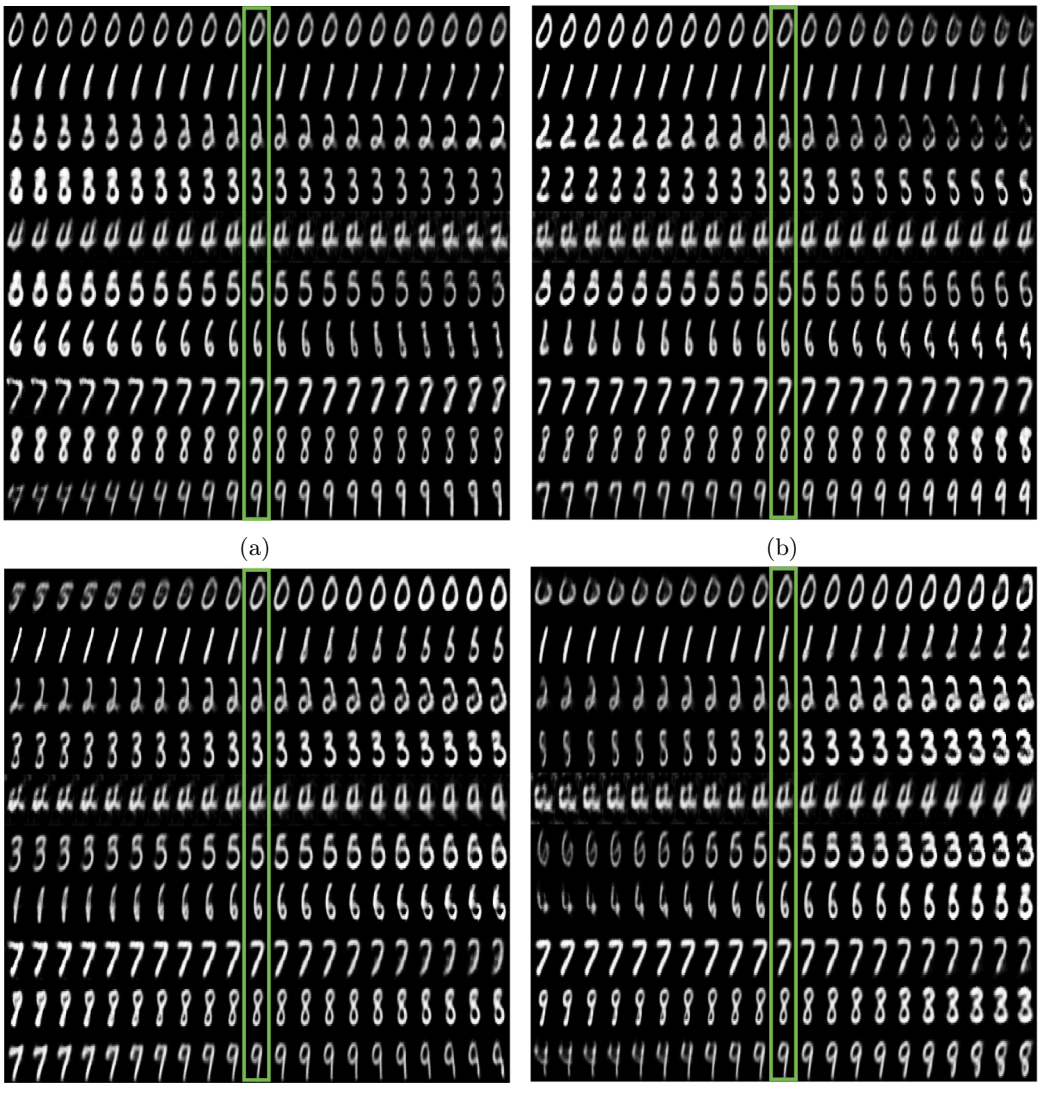
(a) (b)

Figure 16: Examples of images decoded from latent vectors sampled from the posterior of models trained on natural MNIST digits. In each example, the center digit (in the green box) is the decoded version of the input digit and the surrounding digits are images decoded from the sampled latent vectors.



(a) (b) (c) (d)

Figure 17: Extrapolated paths using the first four transport operators learned on natural MNIST digit variations.



(a) (b) (c) (d)

Figure 18: Extrapolated paths using the second four transport operators learned on natural MNIST digit variations.