# Variational Autoencoder with Learned Latent Structure

Marissa C. Connor Gregory H. Canal
Christopher J. Rozell
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
(marissa.connor,gregory.canal,crozell)@gatech.edu

**Abstract**

The manifold hypothesis states that high-dimensional data can be modeled as lying on or near a low-dimensional, nonlinear manifold. Variational Autoencoders (VAEs) approximate this manifold by learning mappings from low-dimensional latent vectors to high-dimensional data while encouraging a global structure in the latent space through the use of a specified prior distribution. When this prior does not match the structure of the true data manifold, it can lead to a less accurate model of the data. To resolve this mismatch, we introduce the Variational Autoencoder with Learned Latent Structure (VAELLS) which incorporates a learnable manifold model into the latent space of a VAE. This enables us to learn the nonlinear manifold structure from the data and use that structure to define a prior in the latent space. The integration of a latent manifold model not only ensures that our prior is well-matched to the data, but also allows us to define generative transformation paths in the latent space and describe class manifolds by transformations stemming from examples of each class. We validate our model on examples with known latent structure and also demonstrate its capabilities on a real-world dataset.[1]

## 1 Introduction

Generative models represent complex data distributions by defining generator functions that map low-dimensional latent vectors to high-dimensional data outputs. In particular, generative models such as variational autoencoders (VAEs) [1] and generative adversarial networks [2] sample from a latent space with a specified prior distribution in order to generate new, realistic samples. VAEs have the additional benefit of an encoder that maps data inputs into the latent space. This enables the input data points to be embedded into the latent space, making VAEs an effective tool for generating and understanding variations in natural data.

According to the manifold hypothesis, high-dimensional data can often be modeled as lying on or near a low-dimensional, nonlinear manifold [3]. There are many manifold learning techniques that compute embeddings of high-dimensional data but very few of them have the ability to generate new points on the manifold [4, 5, 6, 7]. Meanwhile, VAEs can both embed high-dimensional data in a low-dimensional space and generate outputs from that space, making them a convenient model for representing low-dimensional data manifolds.

However, there are several aspects of the traditional VAE framework that prevent it from faithfully representing complex natural variations on manifolds associated with separate data classes. First, VAEs enforce a global structure in the latent space through the use of a prior distribution, and that prior may not match the true data manifold; this model mismatch can result in a less accurate generative model of the data. Second, natural paths are poorly defined in the latent space of traditional VAEs. In many cases, the data transformations are defined using linear paths in a Euclidean latent space [8]. These simple paths can diverge from the true data manifold, leading to interpolated points that result in unrealistic decoded image outputs. Finally, traditional VAEs encourage points from all data classes to cluster around the origin in the latent

[1]Code is available at `https://github.com/siplab-gt/VAELLS`.

space [1]. Without adequate class separation, traversing the latent space can easily result in a change of class, making it difficult to learn identity-preserving transformations and subsequently use them to understand within-class relationships.

In this paper we incorporate a generative manifold model known as *transport operators* [9] into the latent space of a VAE, enabling us to learn the manifold structure from the data and use that structure to define an appropriate prior in the latent space. This approach not only ensures that the prior is well-matched to the data, but it also allows us to define nonlinear transformation paths in the latent space and describe class manifolds by transformations stemming from examples of each class. Our model, named Variational Autoencoder with Learned Latent Structure (VAELLS), more effectively represents natural data manifolds than existing techniques, leads to more accurate generative data outputs, and results in a richer understanding of data variations.

## 2 Methods

### 2.1 Transport operators

The exact structure of natural data manifolds is typically unknown and manifold learning techniques have been introduced to discover the low dimensional structure of data. While there are a variety of manifold learning techniques [4, 5, 6, 7], we desire a manifold model that allows us to learn the data structure, generate points outside of the training set, and map out smooth manifold paths through the latent space. The transport operator manifold model [9] satisfies these requirements by defining a manifold through learned operators that traverse the low-dimensional manifold surface.

Specifically, the basis of the transport operator approach is a linear dynamical system model $\dot{z} = Az$, that defines the dynamics of point $z \in \mathbb{R}^d$ through $A \in \mathbb{R}^{d\times d}$. The solution to this differential equation defines a temporal path given by $z_t = \text{expm}(At)z_0$, where expm is the matrix exponential. This path definition can be generalized to define the transformation between any two points $z_0, z_1 \in \mathbb{R}^d$ on a low-dimensional natural data manifold without an explicit time component by defining $z_1 = \text{expm}(A)z_0 + n$, where $n$ is white Gaussian noise. To allow for different geometrical characteristics at various points on the manifold, our model should have the flexibility to define a different dynamics matrix $A$ between each pair of points. The transport operator technique achieves this property by defining a dynamics matrix that can be decomposed as a weighted sum of $M$ transport operator dictionary elements ($\Psi_m \in \mathbb{R}^{d\times d}$):

$$A = \sum_{m=1}^{M} \Psi_m c_m, \tag{1}$$

The transport operators $\{\Psi_m\}$ constitute a set of primitives that describe local characteristics over the entire manifold, while for each pair of points (i.e., at each manifold location) the geometry is governed by a small subset of operators through coefficients $c \in \mathbb{R}^M$ specific to each pair.

Assuming a Gaussian prior on the dictionary elements and a Laplace sparsity-inducing prior on the coefficients, the resulting negative log posterior for the transport operator model is:

$$-\log p(\Psi \mid z_0, z_1) \propto \|z_1 - T_\Psi(c)z_0\|_2^2 + \frac{\eta}{2}\sum_m \|\Psi_m\|_F^2 + \zeta\|c\|_1, \tag{2}$$

where $T_\Psi(c) = \text{expm}\left(\sum_{m=1}^{M} \Psi_m c_m\right)$ and $\gamma, \zeta > 0$ are hyperparameters.

Following the unsupervised algorithm in [9], the transport operators can be learned from pairs of points on the same manifold using descent techniques that alternate between inferring the coefficients and updating the transport operators. Once learned, these transport operators represent the manifold structure and can be used to generate new points on the manifold. In this paper we incorporate this structured model into a VAE to encourage the latent space structure to adapt to the true data manifold.

### 2.2 Variational autoencoder

The VAE model learns a low-dimensional latent space by defining a generator function $g : \mathcal{Z} \to \mathcal{X}$ that maps latent points, $z \in \mathbb{R}^d$, to high-dimensional data points, $x \in \mathbb{R}^D$. The desired objective for training a VAE is

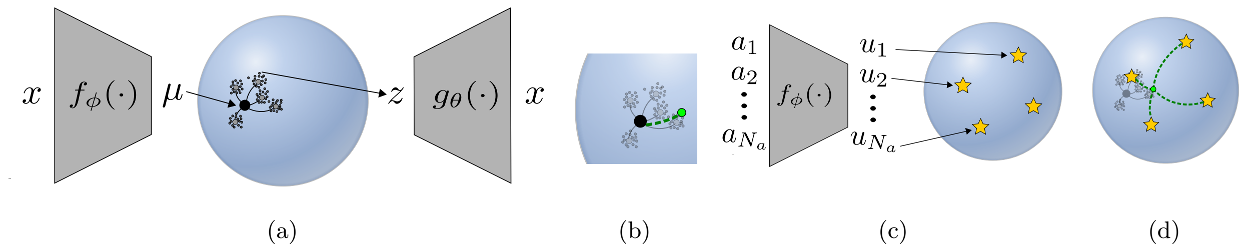


Figure 1: Visualizations of the VAELLS model: (a) Posterior sampling process using transport operator-generated paths and Gaussian noise. (b) Transformation path inferred between $f_\phi(x)$ and $z$ when computing the posterior. (c) Encoding of anchor points into the latent space. (d) Transformation paths inferred between $f_\phi(a_i)$ and $z$ when computing the prior.

maximizing the log-likelihood of a dataset $X = \{x_1, \dots, x_N\}$ given by $\frac{1}{N}\log p(X) = \frac{1}{N}\sum_{i=1}^{N}\log\int p(x_i, z)dz$. However, this objective is difficult to maximize, especially when parameterized by a neural network. To address this complication, VAEs instead maximize the Evidence Lower Bound (ELBO) of the marginal likelihood of each datapoint $x_i$:

$$\log p(x_i) \geq \mathcal{L}(x_i) = \mathbb{E}_{z\sim q_\phi(z|x_i)}\left[-\log q_\phi(z \mid x_i) + \log p_\theta(x_i, z)\right], \tag{3}$$

where $q_\phi(z \mid x)$ is a variational approximation of the true posterior, parameterized by $\phi$. In the VAE neural network model, $\phi$ represents the weights of an encoder network, $f_\phi(x)$.

Kingma and Welling [1] developed an efficient method to approximate the ELBO by introducing the *reparameterization trick* that enables the stochastic latent variable $z$ to be represented by a deterministic function $z = h_\phi(x, \varepsilon)$, where $\varepsilon$ is an auxiliary random variable with a parameter-free distribution. In the traditional VAE framework, the variational posterior is selected to be a multivariate Gaussian distribution, meaning that $z$ is reparameterized around the encoded point $z = f_\phi(x) + \sigma \odot \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, I)$. Additionally, the prior $p_\theta(z)$ is modeled as a zero-mean isotropic normal distribution that encourages the clustering of latent points around the origin.

## 2.3 Variational Autoencoder with Learned Latent Structure

In VAELLS, we fuse the versatile manifold learning capabilities of transport operators with the powerful generative modeling of VAEs. Specifically, we integrate transport operators into both the VAE variational posterior distribution as well as the prior in order to learn a latent probabilistic model that is adapted directly from the data manifold.

For deriving VAELLS, we start with the expanded ELBO from [1]:

$$\mathcal{L}(x) = \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x)\right]. \tag{4}$$

For the likelihood $p_\theta(x \mid z)$, we follow prior work and choose an isotropic normal distribution with mean defined by neural network $g_\theta(z)$ and fixed variance $\sigma^2$, which has worked well in practice.

Our first key contribution lies in the selection of the variational posterior, which we choose as the family of manifold distributions parameterized by learned transport operators as described in Section 2.1. Intuitively, this posterior measures the probability of vector $z$ lying on the manifold in the local neighborhood of the latent encoding of $x$. We encode the latent coordinates of $x$ with a neural network $f_\phi(\cdot)$ and then draw a sample from $q_\phi(z \mid x)$.

To approximate (4) with sampling, first let $L_x(z) \equiv \log p_\theta(x \mid z) + \log p_\theta(z) - \log q_\phi(z \mid x)$ and note that by marginalizing over transport operator coefficients $c$ we have $\mathbb{E}_{z\sim q_\phi(z|x)}\left[L_x(z)\right] = \mathbb{E}_{z,c\sim q_\phi(z,c|x)}\left[L_x(z)\right]$. We draw a sample from $q_\phi(z, c \mid x)$ in two steps: first, as described by the model in (2) we sample a set of coefficients $\widehat{c}$ from a factorized Laplace distribution $q(c)$, and then sample $z$ from $q_\phi(z \mid \widehat{c}, x)$. Both of these sampling steps can be achieved with deterministic mappings on parameter-free random variates, allowing for the use of the reparameterization trick. Specifically,

$$\widehat{c} = l(u) \quad u \sim \text{ Unif}\left(-\frac{1}{2}, \frac{1}{2}\right)^M \quad q_\phi(z \mid \widehat{c}, x) \sim T_\Psi(\widehat{c}) f_\phi(x) + \gamma\varepsilon \quad \varepsilon \sim \mathcal{N}(0, I), \tag{5}$$

where $l(u)$ is a mapping described in Appendix A. Fig. 1a shows this sampling process where the data $x$ is encoded to a mean location $\mu = f_\phi(x)$. The latent vector $z$ is the result of transforming $\mu$ by $T_\Psi(\widehat{c})$ and adding Gaussian noise specified by $\varepsilon$.

The resulting transport operator variational posterior follows as

$$q(c) = 2^{-M} \prod_{m=1}^{M} \exp(-|c_m|) \qquad q_\phi(z \mid c, x) \sim \mathcal{N}\left(T_\Psi(c) f_\phi(x), \gamma^2 I\right)$$
$$q_\phi(z \mid x) = \int_c q_\phi(z \mid c, x) q(c) dc \approx \max_c q_\phi(z \mid c, x) q(c), \tag{6}$$

where the approximation in (6) is used for computational efficiency and is motivated by the fact that the sparsity-inducing Laplace prior on $c$ typically results in joint distributions with $z$ that are highly peaked, as described in [10]. The inferred coefficients $c^*$ that maximize $q_\phi(z \mid c, x)q(c)$ define the estimated transformation between the encoded latent coordinates and the sampled point $z$. Fig. 1b shows a visualization of the inferred transformation path between $f_\phi(x)$ and $z$.

Our next key contribution lies in the construction of a prior distribution learned directly from the underlying data manifold. To gain intuition about this prior, imagine a set of $N_a$ *anchor* points in the data space that correspond to key locations on the manifold, either sampled to uniformly cover the manifold or selected manually by a practitioner (e.g., selecting several anchors per data class). The data manifold structure is represented by a combination of these anchor points and the learned transport operators that can extrapolate the manifold structure in the latent neighborhood of each of them. Fig. 1c shows a set of anchor points encoded into the latent space to represent the scaffold off of which the manifold structure is built. The prior is defined by the same probabilistic manifold model used in the variational posterior, but starting at each anchor point $a_i$ rather than $x$. This prior requires that paths be inferred between $z$ and each encoded anchor point $u_i = f_\phi(a_i)$ as shown in Fig. 1d. The overall prior density for $z$ is then defined as

$$p_\theta(z) = \frac{1}{N_a} \sum_{i=1}^{N_a} q_\phi(z \mid a_i). \tag{7}$$

In [11], a prior is adopted that is similarly composed of a sum of variational posterior terms, and is motivated as an approximation to an optimal prior that maximizes the standard ELBO. However, our motivation for the prior in (7) as a direct sampling of the data manifold is novel, and it structures the prior to align with the manifold since it is constructed from operators that traverse the manifold itself.

The final addition to the VAELLS objective is a Frobenius-norm regularizer on the dictionary magnitudes as used in (2), which can help identify how many transport operators are necessary to represent the manifold. All together, we minimize the following loss function:

$$\mathcal{L}_{VAELLS}(x) = -\mathbb{E}_{u,\varepsilon}\left[L_x(T_\Psi(l(u)) f_\phi(x) + \gamma\varepsilon)\right] + \frac{\eta}{2} \sum \|\Psi_m\|_F^2, \tag{8}$$

We optimize this loss simultaneously over encoder-decoder networks $f_\phi$ and $g_\theta$, anchor points $\{a_i\}_{1:N_a}$, and transport operators $\Psi$. The implementation details of our ELBO and its optimization are described in more detail in Appendix A.

## 3 Related work

There are currently many adaptations of the original VAE that handle a subset of the limitations addressed by VAELLS, such as learning the prior from the data, defining continuous paths in the latent space, and separating the individual class manifolds. Table 1 provides a comparison of techniques which we describe in detail below.

| Model | Adaptive Prior | Defines Paths | Class Separation |
|---|---|---|---|
| VAE [1] | No | Linear | No |
| Hyperspherical VAE [12] | No | Geodesic | No |
| $\Delta$VAE [13] | No | No | No |
| VDAE [18] | **Yes** | Linear | No |
| VAE with VampPrior [11] | **Yes** | No | No |
| $\mathcal{R}$-VAE[19] | **Yes** | **Nonlinear** | No |
| Lie VAE [14] | No | **Nonlinear** | No |
| VAELLS (our approach) | **Yes** | **Nonlinear** | **Yes** |

Table 1: Comparison of VAE techniques

Traditionally, the latent space prior is defined as a Gaussian distribution for model simplicity [1]. However this prior encourages all points to cluster around the origin which may not occur in the natural data manifold. A mismatch between the latent prior distribution of a VAE and the data manifold structure can lead to over regularization and poor data representation. Other models incorporate more complex latent structures such as hyperspheres [12], tori [13, 14], and hyperboloids [15, 16, 17]. These models have demonstrated their suitability for certain datasets by choosing a prior that is the best match out of a predefined set of candidates.

However, these methods are only capable of modeling a limited number of structured priors and are not able to adapt the prior to match the data manifold itself. This is a serious drawback since in most practical cases the latent structure of data is unlikely to easily fit a predefined prior. One example of a VAE prior learned directly from the data is the VampPrior, which is defined using a sum of variational posterior distributions with hyperparameters that are updated during training [11]. The variational diffusion autoencoder (VDAE) and the $\mathcal{R}$-VAE also define the latent space prior directly from the data using Brownian motion on a Riemannian manifold [18, 19].

In addition to differences in how latent space structure is represented, there are several varying approaches for how to define natural paths in this space. In the simplest case, paths in VAEs with a Euclidean latent spaces are modeled as linear paths. Some methods define geodesic transformation paths in the latent space. This can be done by incorporating a structured latent prior, like a hypersphere, on which geodesic paths are natural to compute [12]. Interpolated geodesic paths can also be estimated in a Euclidean latent space using an estimated Riemannian metric [19, 20, 21, 22]. However, these methods are limited to defining extrapolated paths by random walks in the latent space rather than structured paths. Finally, there are other methods that lack straightforward definitions for how to compute continuous paths from one point to another [11, 13].

Another limitation of most VAE models is that they do not encourage class separation and therefore generated paths often do not represent natural identity-preserving transformations on separate class data manifolds. This makes it difficult to understand the within-class relationships in the data. Some techniques encourage class separation through the choice of a prior that does not encourage data clustering [12] but they do not explicitly define separate class manifolds. By defining the prior structure with respect to anchor points on specified data manifolds, VAELLS has the flexibility to define which manifolds it wants to learn transformations on. This results in a latent space structure where transformations correspond to identity-preserving variations in the data.

Two models that have notable similarities to ours are the Lie VAE [14] and the Manifold Autoencoder [23]. Both models also use Lie group representations of transformations in the latent space. The Lie VAE model encodes the data into latent variables that are elements in a Lie group which represent transformations of a reference object. This model requires the type of Lie group transformations that the network will represent (e.g. $SO(3)$) to be specified prior to training which may result in a model mismatch. The Manifold Autoencoder [23] also represents data variation in a latent space using the transport operator model, but it only defines these variations in the context of a deterministic autoencoder mapping. This approach shares the motivation of representing the structured data manifold in the latent space but it lacks the fully probabilistic generative framework modeled in VAELLS.

## 4 Experiments

Our experiments highlight the strengths of VAELLS: the ability to adapt the prior to the true data manifold structure, the ability to define nonlinear paths in the latent space, and the ability to separate classes by learning identity-preserving transformations within classes specified by anchor points. First, we begin with two simple datasets with known ground truth latent structures in order to validate the ability for our model to learn the true latent structure. Next, we apply VAELLS to rotated and naturally varying MNIST digits to show that our prior can adapt to represent rotations of individual digits through a learned operator and extend to real-world data with natural transformations.

The unique characteristics of the VAELLS variational posterior and prior lead to specific training considerations. First, as shown in (6), to compute both the variational posterior and prior distributions we must infer transformation coefficients that maximize $q_\phi(z \mid c, x)q(c)$ and $q_\phi(z \mid c, a_i)q(c)$ respectively. This involves coefficient inference between each sampled point $z$ and its neural network encoding $f_\phi(x)$ as well as between $z$ and all encoded anchor points $f_\phi(a_i)$. The coefficient inference is performed using a conjugate gradient descent optimization solver.

For training the networks weights we use the Adam algorithm [24]. To add stability and improve efficiency of training, we alternate between steps where we update the network 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. Anchor points are initialized by selecting training samples from individual classes in the input space. If the dataset contains multiple classes, each training sample from a specific class is compared against only anchor points from its class. Details of the network architectures and training for each experiment are available in the Appendix.

**Swiss roll:** We begin by applying VAELLS to a dataset composed of 20-dimensional vector inputs that are mapped from a 2D ground truth latent space with a swiss roll structure (Fig. 2a). We selected this classic manifold test structure because many VAE techniques that incorporate specific structured priors into the latent space have not demonstrated the ability to adapt to this specific geometry. The latent space is two-dimensional and the VAELLS prior uses four anchor points.

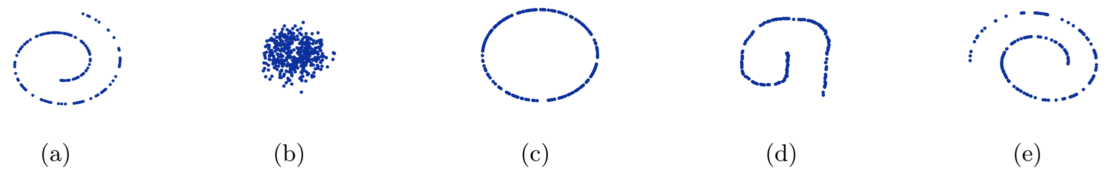


Figure 2: Embedding of swiss roll inputs in VAE latent spaces. (a) Ground truth latent embedding. (b) VAE. (c) Hyperspherical VAE. (d) VAE with VampPrior. (e) VAELLS.

Fig. 2 shows the latent space embedding for several VAE techniques. The traditional VAE with a Gaussian prior in the latent space loses the latent structure of the true data manifold because it encourages all points to cluster around the origin (Fig. 2b). The hyperspherical VAE similarly loses the true data structure because it distributes the latent points on a hypersphere (Fig. 2c). The VAE with VampPrior is able to estimate the spiraling characteristic of the swiss roll structure (Fig. 2d), but it is not a smooth representation of the true data manifold. By contrast, the encoded points in VAELLS (Fig. 2e) clearly adapt to the swiss roll structure of the data.

We also utilize the swiss roll dataset to provide an intuitive understanding of how the prior in our method is formed as a combination of the learned transport operators and the encoded anchor points. Fig. 3a contains the encoded latent points overlaid with the orbit of the operator learned by VAELLS. Specifically, the colored line shows how the transport operator evolves over time when applied to a single starting point: $z_t = \text{expm}(\Psi_m \frac{t}{T})z_0$, $t = 0, ..., T$. Fig. 3b contains latent points sampled from the prior using the sampling described in (5). This shows how the prior has been well-adapted to the swiss roll structure. Finally, we demonstrate how transport operators can be used to define nonlinear paths in the latent space. To generate paths between pairs of points with our learned operators, we first infer the coefficients $c^*$ between each

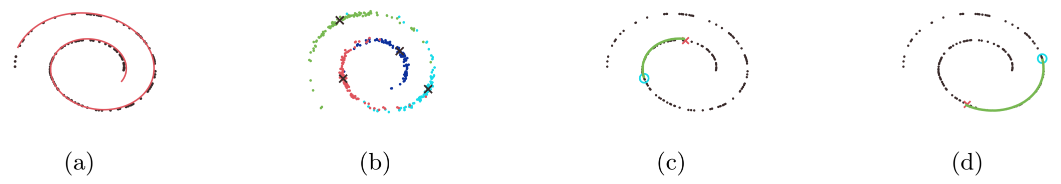


Figure 3: (a) The orbit of the transport operator learned on the swiss roll dataset plotted on top of encoded latent points. (b) Sampling of the latent space prior from each of the anchor points; each color indicates a separate anchor point of origin. (c-d) Transport operator paths inferred between pairs of points on the swiss roll manifold in the latent space.

pair. We then interpolate the path from the starting point $x_0$ as follows: $x_t = \text{expm}\left(\sum_{m=1}^{M} \Psi_m c_m^* t\right) x_0$. Fig. 3(c-d) show two example inferred paths between points encoded on the swiss roll manifold.

**Concentric circle:** Next we apply VAELLS to a dataset composed of 20-dimensional data points that are mapped from a 2D ground truth latent space with two concentric circles (Fig 4a). As in the previous example, our network maps these inputs into a two-dimensional latent space. We select three anchor points per concentric circle with the anchor points evenly spaced.

This dataset in particular is well-suited for assessing how well each method is able to discriminate between the two concentric circle manifolds once points are mapped into the latent space. Fig. 4 shows the encoded latent points for several different VAE approaches. All three comparison techniques (Fig. 4(b-d)) lose the class separation between the ground truth concentric circle manifolds. Additionally, as in Fig. 2 the Gaussian prior of the traditional VAE distorts the true data structure (Fig. 4b), and the VAE with VampPrior encodes a latent structure with similar characteristics to the ground truth manifold but fails to model the exact shape (Fig. 4d). By contrast, the encoded points in the VAELLS latent space maintain the class separation while simultaneously encoding the true circular structure. This verifies two characteristics of our approach that improve upon the traditional VAE model – learning the prior from the data and class separation.

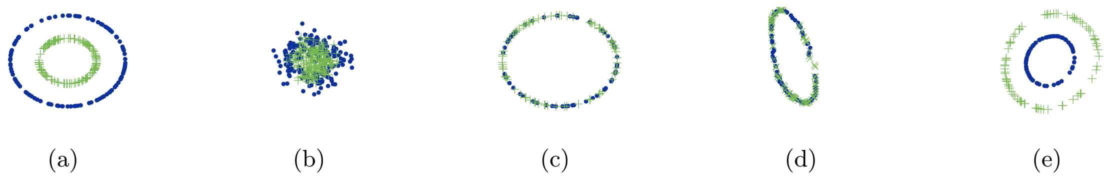


Figure 4: Embedding of concentric circle inputs in VAE latent spaces. (a) Ground truth latent embedding. (b) VAE. (c) Hyperspherical VAE. (d) VAE with VampPrior. (e) VAELLS.

**Rotated MNIST digits:** The rotated MNIST dataset [25] is a natural choice for demonstrating VAELLS because it consists of real images in which we have an intuitive understanding of what the rotational transformations should look like. To define the rotated digit manifold, we specify anchor points as rotated versions of training inputs and aim to learn a transport operator that induces latent space transformations corresponding to digit rotation. In practice this means that for each training sample we select several rotated versions of that digit as anchor points.

First, to highlight that we can learn a transformation model that is adapted to the rotated digit manifold, we show the result of generating data from input points in the test set using the sampling procedure described in (5). Fig. 5 shows the decoded outputs of latent vectors sampled from the posterior for two example test points using four different VAE models. In each example, the center image (enclosed in a green box) is the decoded version of the input test sample. The images surrounding each center are decoded outputs of latent posterior samples. In order to visualize noticeable sampling variations, we increase the standard deviation and spread of the sampling noise in each of these models. The key result is that the VAELLS sampling procedure using the learned transport operator leads to latent space transformations that correspond to

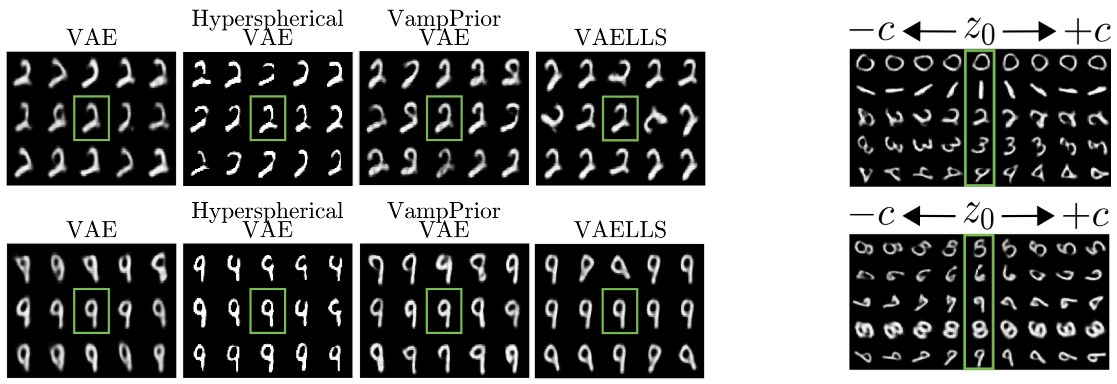

VAE
Hyperspherical VAE
VampPrior VAE
VAELLS
$-c \longleftarrow z_0 \longrightarrow +c$
VAE
Hyperspherical VAE
VampPrior VAE
VAELLS
$-c \longleftarrow z_0 \longrightarrow +c$

Figure 5: (L) 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 encoded digit input and the surrounding digits are from the sampled latent vectors. Sampling in the VAELLS latent space results in rotations in the sampled outputs. (R) Extrapolated rotation paths in the VAELLS latent space. The center image in each row (in the green box) is a decoded version of an input digit, with decoded outputs from extrapolated latent paths with negative and positive transport operator coefficients shown to the left and right respectively.

rotations in decoded outputs. This verifies that the transport operator corresponds to movement on a learned rotated digit manifold, unlike comparison techniques which only capture natural digit variations and not specifically rotation.

In Fig. 5 we show how we can extrapolate rotational paths in the latent space using our learned transport operator. To generate this figure, we randomly select example MNIST digits with zero degrees of rotation and then encoded those digits to get each starting point $z_0$. The decoded versions of these initial points are shown in the middle columns (enclosed by a green box) in each figure. We then apply the learned operator with both positive and negative coefficients to $z_0$ and decode the outputs. The images to the left of center show the path generated with negative coefficients and the images to the right of center show the path generated with positive coefficients. This shows how we can generate rotated paths using the learned transport operator. It also highlights the ability for VAELLS to define identity-preserving transformations with respect to selected input points. In these examples, the class identity of the digit is qualitatively preserved for about 180 degrees of rotation.

**Natural MNIST digits:** In our final experiment, we highlight our ability to learn the natural manifold structure in MNIST digits. The anchor points are initialized by randomly selecting training examples from each digit class. Without a priori knowledge of the manifold structure, we have flexibility in how to parameterize our model; two parameters of specific interest are the number of transport operator dictionary elements $M$ used to define latent space transformations and the number of anchor points per class $N_a$. These parameters impact the two components of the prior definition: the learned transport operator model and the anchor points. The table in Fig. 6 shows how varying these parameters impacts the quantitative performance of VAELLS as measured by estimated log-likelihood (LL) [26] and mean-squared error (MSE) between input and reconstructed images. In general, performance is stable across parameter settings (we select $M = 4$, $N_a = 8$ in the results below).

Fig. 6 shows the result of sampling the variational posterior in a similar manner to Fig. 5. Note that sampling in the VAELLS latent space leads to natural digit transformations in the decoded outputs that maintain the class of the original digit. By contrast, the sampling in the latent space of the comparison techniques can lead to changes in class. Appendix F contains paths generated by each of the learned transport operators that highlight how they represent natural transformation paths.

| $M$ | $N_a$ | LL | MSE |
|---|---|---|---|
| 2 | 8 | -744.9376 | 0.0286 |
| 4 | 8 | -744.6206 | 0.0236 |
| 8 | 8 | -756.7853 | 0.0233 |
| 4 | 12 | -745.7737 | 0.0222 |
| 4 | 16 | -744.1835 | 0.0222 |

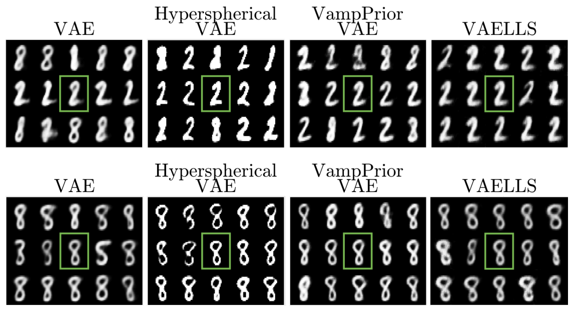


Figure 6: (L) Evaluation metrics of VAELLS trained on MNIST with varying model parameters. (R) Examples of images decoded from latent vectors sampled from the posterior of a model trained on MNIST digits. In each example, the center digit (in the green box) is the encoded digit input and the surrounding digits are from the sampled latent vectors.

## 5 Conclusion

In this paper we developed a model that has the flexibility to learn a structured VAE prior from the training data by incorporating manifold transport operators into the latent space. This adaptable prior allows us to define a generative model with a latent structure that is a better fit to the data manifold. It also enables us to both interpolate and extrapolate nonlinear transformation paths in the latent space and to explicitly incorporate class separation by learning identity-preserving transformations. We verified the performance of this model on datasets with known latent structure and then extended it to real-world data to learn natural transformations. VAELLS can be used to not only develop more realistic generative models of data but it can also be used to more effectively understand natural variations occurring in complex data.

## 6 Acknowledgments

This work is partially supported by NSF grant number CCF-1350954 and ONR grant number N00014-15-1-2619.

## References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[4] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[5] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[6] Piotr Dollár, Vincent Rabaud, and Serge J Belongie. Learning to traverse image manifolds. In *Advances in neural information processing systems*, pages 361–368, 2007.

[7] Yoshua Bengio and Martin Monperrus. Non-local manifold tangent learning. *Advances in Neural Information Processing Systems*, 17:129–136, 2005.

[8] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[9] Benjamin J Culpepper and Bruno A Olshausen. Learning transport operators for image manifolds. In *NIPS*, pages 423–431, 2009.

[10] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[11] Jakub M Tomczak and Max Welling. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017.

[12] Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018.

[13] Luis A. Pérez Rey, Vlado Menkovski, and Jacobus W. Portegies. Diffusion variational autoencoders. *CoRR*, abs/1901.08991, 2019.

[14] Luca Falorsi, Pim de Haan, Tim R Davidson, and Patrick Forré. Reparameterizing distributions on lie groups. *arXiv preprint arXiv:1903.02958*, 2019.

[15] Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with poincaré variational auto-encoders. In *Advances in neural information processing systems*, pages 12544–12555, 2019.

[16] Yoshihiro Nagano, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama. A wrapped normal distribution on hyperbolic space for gradient-based learning. In *International Conference on Machine Learning*, pages 4693–4702, 2019.

[17] Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. Mixed-curvature variational autoencoders. *arXiv preprint arXiv:1911.08411*, 2019.

[18] Henry Li, Ofir Lindenbaum, Xiuyuan Cheng, and Alexander Cloninger. Variational diffusion autoencoders with random walk sampling. *arXiv preprint arXiv:1905.12724*, 2019.

[19] Dimitris Kalatzis, David Eklund, Georgios Arvanitidis, and Søren Hauberg. Variational autoencoders with riemannian brownian motion priors. *arXiv preprint arXiv:2002.05227*, 2020.

[20] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*, 2017.

[21] Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick van der Smagt. Metrics for deep generative models. *arXiv preprint arXiv:1711.01204*, 2017.

[22] Hang Shao, Abhishek Kumar, and P Thomas Fletcher. The riemannian geometry of deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 315–323, 2018.

[23] Marissa Connor and Christopher Rozell. Representing closed transformation paths in encoded network latent space. *AAAI Conference on Artificial Intelligence*, 2019.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

## 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 (8), restated here for convenience:

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

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

$$\begin{aligned} \widehat{E}(x) &\equiv -\frac{1}{N_s}\sum_{s=1}^{N_s} L_x(T_\Psi(l(u_s))f_\phi(x) + \gamma\varepsilon_s) + \frac{\eta}{2}\sum_{m=1}^{M}\|\Psi_m\|_F^2 \\ &\varepsilon_s \sim \mathcal{N}(0,I) \quad u_s \sim \text{ Unif}\left(-\frac{1}{2},\frac{1}{2}\right)^M \end{aligned} \tag{9}$$

The deterministic mapping $l(u)$ 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 $\zeta_3 > 0$,

$$q(c_m) = \frac{\zeta_3}{2}\exp(-\zeta_3 |c_m|)$$

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

$$l_m(u) = -\frac{\text{sgn}(u_m)}{\zeta_3}\log(1 - 2|u_m|)$$

Next we derive expressions for each expanded term in (9). 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

$$\begin{aligned} \log q_\phi(z \mid x) &= \log\int_c q_\phi(z, c \mid x)dc \\ &\approx \max_c \log\left[q_\phi(z \mid c, x)q(c)\right] \end{aligned} \tag{10}$$

where the approximation in (10) is described in Section 2.3. 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 $\zeta_3$ are treated as a hyperparameters and $C_2 = -\frac{d}{2}\log(2\pi) - d\log\gamma + M\log\frac{\zeta_3}{2}$. Using the notation

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

we have (with hyperparameter $\zeta_q$)

$$\log q_\phi(z \mid x) \approx C_2 - \zeta_2 \left\|z - T_\Psi(c^*(z,x;\zeta_q))f_\phi(x)\right\|_2^2 - \zeta_3 \sum_{m=1}^{M} |c_m^*(z,x;\zeta_q)| \tag{11}$$

Finally, for the prior distribution (with hyperparameter $\zeta_p$) we have

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

where we use the same approximation as (10).

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

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

In practice, one may wish to construct the prior with different constants than those used for the variational posterior term (i.e., constants $\zeta_4, \zeta_5$ instead of $\zeta_2, \zeta_3$). This substitution results in

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

## B VAELLS training procedure details

In this section we provide additional details on the general training procedure for all of the experiments. 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 [11]. Our warm-up includes updates to the network weights driven by only the reconstruction loss with no Gaussian sampling in the latent space and very limited sampling of the transport operator coefficients. 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 lead the coefficient inference optimization to 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 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. This includes the reconstruction portion of the variational posterior objective term and the entire prior objective term. If this portion of the objective does not improve with a gradient step on the dictionary than we reject this step and decrease the transport operator learning rate. If 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 latent training points in (2) 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 selected to be evenly sampled 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:

$$p_\theta(z) = q_\phi(z \mid a^*), \tag{12}$$

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 $z$ using the current transport operator dictionary elements $\Psi$.

There are many hyperparameters that need to be tuned in this model. The hyperparameters that were shown to have the largest effect on training effectiveness were:

- The weight on the reconstruction term ($\zeta_1$) - use this in combination with warm-up steps to ensure reasonable reconstruction accuracy from the decoder.
- The posterior coefficient inference weight ($\zeta_q$)- this is the weight on the sparsity regularizer term used in the objective (11) 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 ($\zeta_p$) - 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 $lr_\psi$ - As mentioned above, we do vary $lr_\psi$ during transport operator training depending on whether our training steps are successful or not. If this learning rate starts too high, it can mean the training procedure takes many unsuccessful steps with no updates on the transport operator dictionaries which greatly slows down training.

**Comparison techniques** We implemented the hyperspherical VAE [12] 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 2. For MNIST experiments, we used the network architecture provided with the code for their MNIST experiments, and we dynamically binarized the MNIST inputs as they did. We implemented the VAE with VampPrior [11] 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 2. For the MNIST experiments we adapted the network in Table 5 to add a linear layer between the final convTranspose layer and the sigmoid layer. 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 2 and 5.

| 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 2: Network architecture for swiss roll experiment

| VAELLS Training |
|---|
| batch size: 30 |
| training steps: 3000 |
| latent space dimension ($z_{dim}$): 2 |
| $N_s : 1$ |
| $lr_{net} : 10^{-4}1$ |
| $lr_{anchor} : 10^{-4}$ |
| starting $lr_{\Psi} : 5 \times 10^{-5}$ |
| $\zeta_1 : 0.01$ |
| $\zeta_2 : 1$ |
| $\zeta_3 : 1$ |
| $\zeta_4 : 1$ |
| $\zeta_5 : 0.01$ |
| $\zeta_q$: $1 \times 10^{-6}$ |
| $\zeta_p$: $5 \times 10^{-5}$ |
| $\eta : 0.01$ |
| number of network and anchor update steps: 20 |
| weight on prior terms during network update steps: 0.01 |
| number of $\Psi$ update steps: 20 |
| weight on reconstruction term during network update steps: 0.001 |
| $\gamma_{post} : 0.001$ |
| warm-up steps: 0 |
| number of restarts for coefficient inference: 2 |
| $M : 1$ |
| number of anchors: 4 |

Table 3: Training parameters for swiss roll experiment

## C Swiss roll experiment

Tables 2 and 3 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 defined the prior only with respect to the anchor points that were estimated to be closest to each training point.

## D Concentric circle experiment

The concentric circle experiment uses the same network architecture as the swiss roll experiment which is specified in Table 2. Table 4 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 it to the 20-dimensional input space using a random linear mapping. We use 400 training points and randomly sample

| 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 $lr_{\Psi}$ : $4 \times 10^{-4}$ |
| $\zeta_1$ : 0.01 |
| $\zeta_2$ : 1 |
| $\zeta_3$ : 1 |
| $\zeta_4$ : 1 |
| $\zeta_5$ : 0.01 |
| $\zeta_q$: $1 \times 10^{-6}$ |
| $\zeta_p$: $5 \times 10^{-6}$ |
| $\eta$ : 0.01 |
| number of network and updates steps: N/A |
| number of $\Psi$ update steps: N/A |
| $\gamma_{post}$ : 0.001 |
| warm-up steps: 0 |
| number of restarts for coefficient inference: 1 |
| $M$ : 4 |
| number of anchors per class: 3 |

Table 4: Training parameters for concentric circle experiment

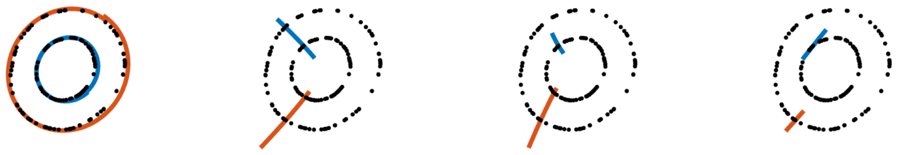


Figure 7: The orbits of each of the transport operators learned in the concentric circle test case plotted on top of encoded points.

test points.

It should be noted that, in this experiment, we did 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 point 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 contribution is computed as a sum over the variational posterior conditioned on only the anchor points on the same circle as the input point.

Fig. 7 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 it 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. 8 shows the magnitude of each of the transport operators after training the VAELLS model in the concentric circle test case.

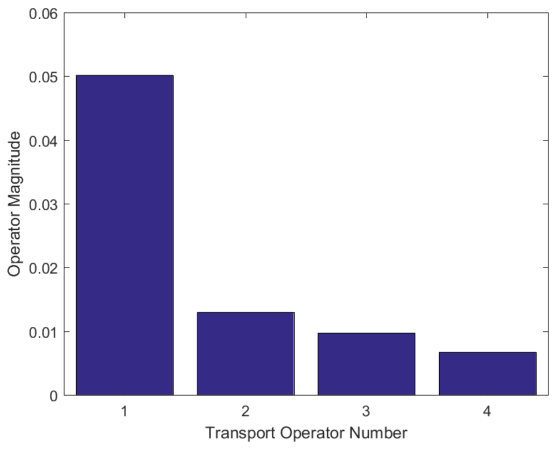
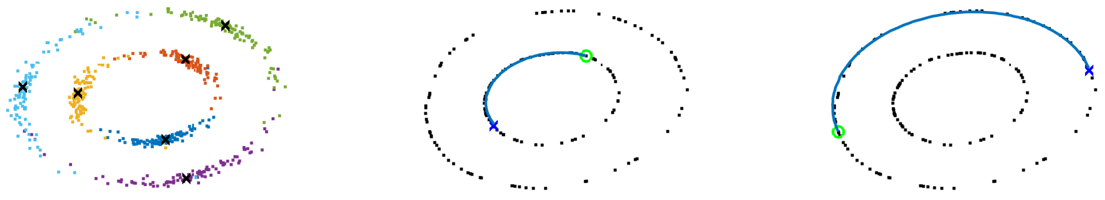


Figure 8: Magnitude of the operators after training on the 2D concentric circle experiment.



(a) (b) (c)

Figure 9: (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.

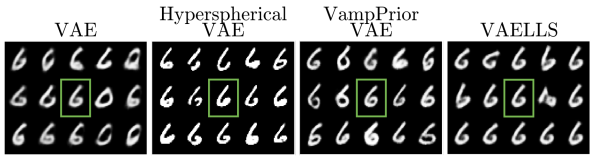
Fig. 9a shows latent points sampled from the prior using the sampling described in (5). Fig. 9(b-c) show two example inferred paths between points encoded on the concentric circle manifold.

## E Rotated MNIST experiment

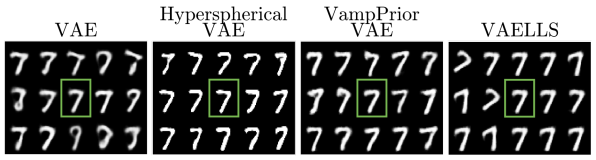
We split the MNIST dataset into training, validation, and testing sets. The training set contains 50,000 images from the traditional MNIST training set. The validation set is made up of the remaining 10,000 image from the traditional MNIST training set. We use the traditional MNIST testing set for our 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 example, they are not updated during training. Tables 5 and 6 contain the VAELLS network architecture and parameters for the rotated MNIST experiment. Fig. 10 shows more examples of images decoded from latent vectors sampled from the posterior of the models trained on rotated MNIST digits. As described in Section B, in this experiment, we defined the prior only with respect to the anchor points that were estimated to be closest to each training point.

| 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 |

Table 5: Network architecture for rotated MNIST experiment



(a)

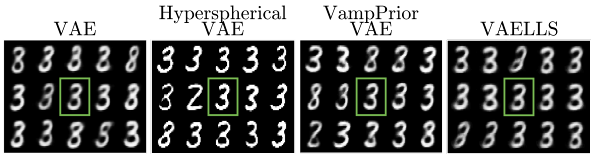


(b)

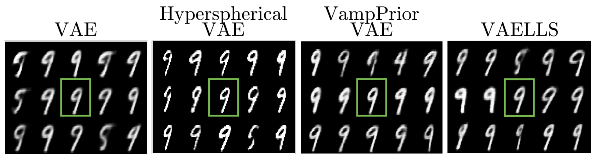
Figure 10: Examples of images decoded from latent vectors sampled from the posterior of models trained on rotated MNIST digits. In each example, the center digit (in the green box) is the encoded digit input and the surrounding digits are from the sampled latent vectors. Sampling in the VAELLS latent space results in rotations in the sampled outputs.

## F Natural MNIST experiment

This experiment uses the same training/validation/testing separation as described in the rotated MNIST experiment in Section E. 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 5). Table 7 shows the training parameters for this experiment. Fig. 11 shows more examples of images decoded from latent vectors sampled from the posterior of the models trained on MNIST digits. 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.



(a)



(b)

Figure 11: 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 encoded digit input and the surrounding digits are from the sampled latent vectors. Sampling in the VAELLS latent space results in identity-preserving transformations.

Figure 12 shows the effect that each of the four 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

| 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 $lr_{\Psi}$ : $1 \times 10^{-5}$ |
| $\zeta_1$ : 1 |
| $\zeta_2$ : 1 |
| $\zeta_3$ : 1 |
| $\zeta_4$ : 1 |
| $\zeta_5$ : 0.01 |
| $\zeta_q$: $1 \times 10^{-6}$ |
| $\zeta_p$: $1 \times 10^{-6}$ |
| $\eta$ : 0.01 |
| number of network and anchor update steps: 20 |
| weight on prior terms during network update steps: 0.0001 |
| number of $\Psi$ update steps: 60 |
| weight on reconstruction term during network update steps: 0.0001 |
| $\gamma_{post}$ : 0.001 |
| warm-up steps: 30000 |
| number of restarts for coefficient inference: 1 |
| $M$ : 1 |
| number of anchors per class: 10 |
| latent space scaling: 10 |

Table 6: Training parameters for rotated MNIST experiment

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.

| VAELLS Training - MNIST |
|---|
| batch size: 32 |
| training steps: 35000 |
| latent space dimension ($z_{dim}$): 6 |
| $N_s : 1$ |
| $lr_{net} : 10^{-4}1$ |
| $lr_{anchor} : 10^{-4}$ |
| starting $lr_{\Psi} : 1 \times 10^{-5}$ |
| $\zeta_1 : 1$ |
| $\zeta_2 : 1$ |
| $\zeta_3 : 1$ |
| $\zeta_4 : 1$ |
| $\zeta_5 : 0.01$ |
| $\zeta_q$: $1 \times 10^{-6}$ |
| $\zeta_p$: $1 \times 10^{-6}$ |
| $\eta : 0.01$ |
| number of network and anchor update steps: 20 |
| weight on prior terms during network update steps: 0.0001 |
| number of $\Psi$ update steps: 60 |
| weight on reconstruction term during network update steps: 0.0001 |
| $\gamma_{post} : 0.001$ |
| warm-up steps: 30000 |
| number of restarts for coefficient inference: 1 |
| $M : 4$ |
| number of anchors per class: 8 |
| latent space scaling: 10 |

Table 7: Training parameters for natural MNIST experiment

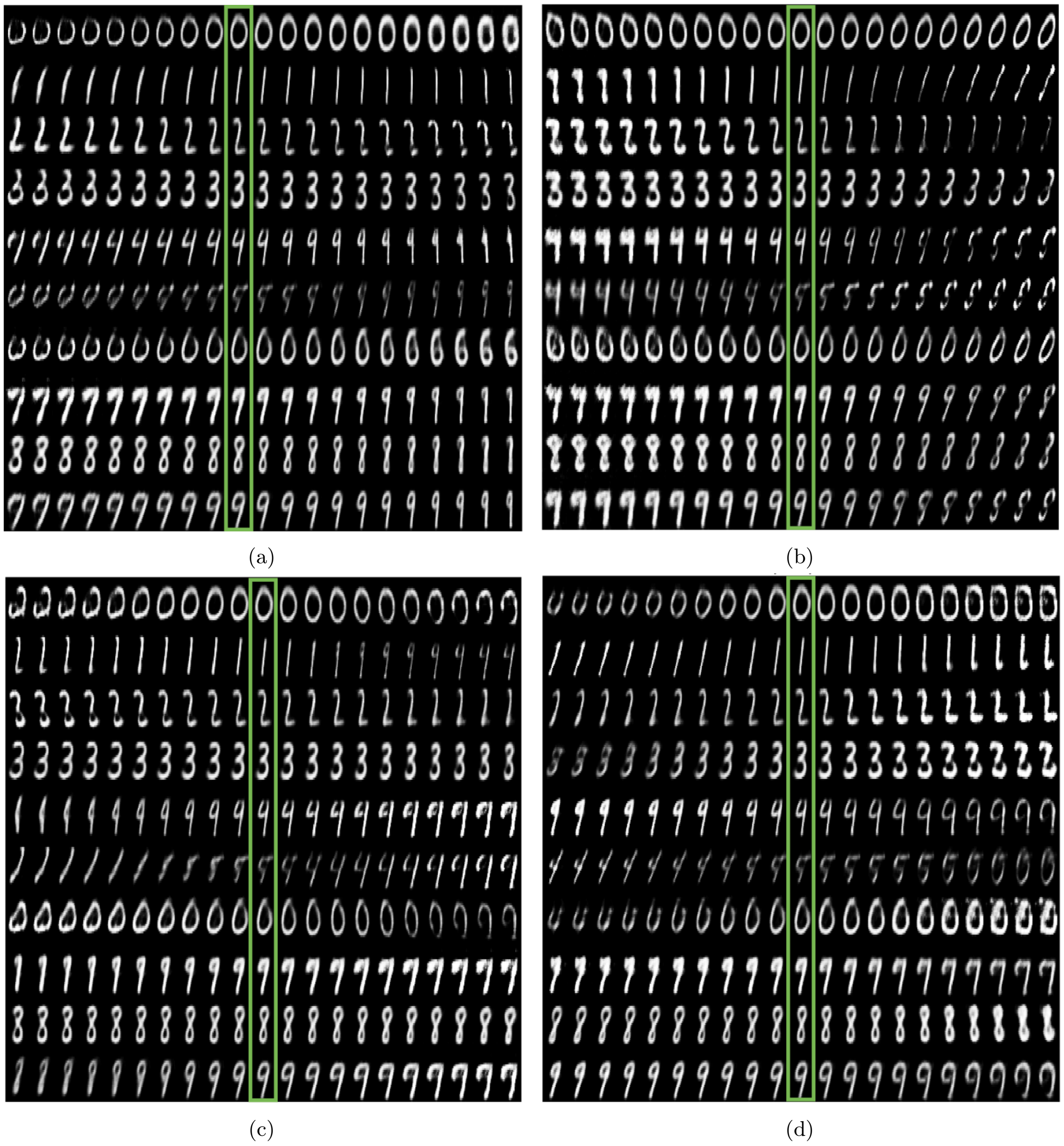


Figure 12: Extrapolated paths using each of the transport operators learned on natural MNIST digit variations.