Learning Identity-Preserving Transformations on Data Manifolds

Anonymous authors
Paper under double-blind review

Abstract

Many machine learning techniques incorporate identity-preserving transformations into their models to generalize their performance to previously unseen data. These transformations are typically selected from a set of functions that are known to maintain the identity of an input when applied (e.g., rotation, translation, flipping, and scaling). However, there are many natural variations that cannot be labeled for supervision or defined through examination of the data. As suggested by the manifold hypothesis, many of these natural variations live on or near a low-dimensional, nonlinear manifold. Several techniques represent manifold variations through a set of learned Lie group operators that define directions of motion on the manifold. However, these approaches are limited because they require transformation labels when training their models and they lack a method for determining which regions of the manifold are appropriate for applying each specific operator. We address these limitations by introducing a learning strategy that does not require transformation labels and developing a method that learns the local regions where each operator is likely to be used while preserving the identity of inputs. Experiments on MNIST and Fashion MNIST highlight our model's ability to learn identity-preserving transformations on multi-class datasets. Additionally, we train on CelebA to showcase our model's ability to learn semantically meaningful transformations on complex datasets in an unsupervised manner.

1 Introduction

A goal of many machine learning models is to accurately identify objects as they undergo natural transformations – a task that humans are adept at. According to the manifold hypothesis, natural variations in high-dimensional data lie on or near a low-dimensional, nonlinear manifold (Fefferman et al., 2016). Additionally, the manifolds representing different classes are separated by low density regions (Rifai et al., 2011a). Natural physical laws govern the possible transformations that objects can undergo and many of the identity-preserving transformations (e.g., changes in viewpoint, color, and lighting) are shared among classes of data. The sharing of transformations between classes enables increased efficiency in defining data variations – a model can represent a limited set of transformations that can describe a majority of variations in many classes. Several machine learning models incorporate specific identity-preserving transformations that are shared among a large number of classes to generalize the performance of their model to unseen data. These include equivariant models that incorporate transformations like translation and rotation into intermediate network layers (Cohen & Welling, 2016; Cohen et al., 2018) and data augmentation techniques that apply known identity-preserving variations to data while training (Cubuk et al., 2019; Ho et al., 2019; Lim et al., 2019). However, in many datasets there may be natural transformations shared among classes that are not easily prespecified from intuition, making it critical that we develop a model that can learn both 1) a representation for these transformations without explicit transformation labels and 2) the context of when each transformation is likely to be relevant.

Manifold learning strategies estimate the low-dimensional manifold structure of data and examine how points are related through natural transformations. Most common techniques learn manifold embeddings of data points without a method for generating transformed points on the manifold (Tenenbaum et al., 2000; Roweis & Saul, 2000; Belkin & Niyogi, 2003; Maaten & Hinton, 2008). Other manifold learning techniques learn to

transform points on the manifold either along linear manifold tangent planes (Dollár et al., 2007; Bengio & Monperrus, 2005; Park et al., 2015; Rifai et al., 2011c; Kumar et al., 2017) or through nonlinear Lie group operators that transverse the manifold (Rao & Ruderman, 1999; Miao & Rao, 2007; Culpepper & Olshausen, 2009; Sohl-Dickstein et al., 2010; Cohen & Welling, 2014; Hauberg et al., 2016; Connor & Rozell, 2020; Connor et al., 2021). Lie group operators represent infinitesimal transformations which can be applied to data through an exponential mapping to transform points along a manifold, and a manifold can be globally defined by a set of operators that each move in different directions along it (Hoffman, 1966; Dodwell, 1983). A Lie group operator model is well-suited for representing natural data variations because the operators can be learned from the data, applied to data points to transform them beyond their local neighborhoods, and used to estimate geodesic paths.

While the Lie group operator models have many benefits, previous approaches demonstrate two shortcomings. First, to learn Lie group operators that represent a data manifold, pairs of training points are selected which lie within a neighborhood of one another. The training objective encourages efficient paths between these nearby points, and the choice of training point pairs strongly influences the types of manifold transformations that are learned. Recent papers incorporating Lie group operators into machine learning models have either used predefined operators that represent known transformations groups (e.g., the 3D rotational group $SO(3)$ (Falorsi et al., 2019)), required transformation labels for selecting point pairs when training (Connor & Rozell, 2020), or randomly selected pairs of points from the same class (Connor et al., 2021). To learn an effective model with datasets having no labeled transformation structure, we require a strategy for selecting point pairs that automatically identifies points that are related through the transformations the model aims to learn. Second, existing Lie group operator models have lacked a method for determining which regions of the manifold are appropriate for each operator, meaning existing approaches assume that every operator is equally likely to be used at every point on the manifold. This is a flawed assumption because, while many transformations are shared between classes, there are also data variations that are unique to specific data classes. Additionally, in a dataset with several manifolds (each representing a different class), there is a limited extent to which a transformation can be applied without moving a point onto another manifold.

The main contributions of this paper are the development of methods to address, in the context of Lie group operator models, the two critical shortcomings of generative manifold models noted above. Specifically, motivated by finding perceptually similar training samples without transformation labels, we first introduce a point pair selection strategy to learn a manifold representation of natural variations shared across multiple data classes without requiring transformations labels. Second, we develop a method that uses a pretrained classifier (measuring identity-preservation of transformed samples) to learn the local regions where each operator is likely to be used while preserving the identity of transformed samples. This approach enables us to analyze the local structure of the data manifold in the context of the learned operators and to describe the invariances of the classifier. We demonstrate the efficacy of these strategies in the context of the Manifold Autoencoder (MAE) model (Connor & Rozell, 2020) to learn semantically meaningful transformations on MNIST (LeCun et al., 1998), Fashion MNIST (Xiao et al., 2017), and CelebA (Liu et al., 2015).

2 Background

Manifold Learning Traditional manifold learning models estimate the low-dimensional structure of high-dimensional data by utilizing the property that local neighborhoods on the manifold are approximately linear. Some techniques represent the manifold through a low-dimensional embedding of the data points (Tenenbaum et al., 2000; Roweis & Saul, 2000; Belkin & Niyogi, 2003; Maaten & Hinton, 2008). Others learn functions that map high-dimensional data points to linear tangent planes that represent directions of manifold motion in the local neighborhood around the points (Dollár et al., 2007; Bengio & Monperrus, 2005; Park et al., 2015). While traditional manifold learning approaches that are applied directly to data points are useful for understanding low-dimensional data structure, in many cases, the input data space is an inefficient representation of the data. For example, data in the pixel space suffers from the curse of dimensionality and cannot be smoothly interpolated while maintaining identity (Bengio et al., 2005). Many approaches have used neural networks to learn a low-dimensional latent space in which manifold models can be incorporated. The contractive autoencoder (CAE) estimates manifold tangents by minimizing the Jacobian of the encoder network, encouraging invariance of latent vectors to image space perturbations. Tangent directions can then

be estimated through top singular vectors of the encoder Jacobian (Rifai et al., 2011c;b;a; Kumar et al., 2017). Several methods estimate interpolated geodesic paths in the latent space of a trained variational autoencoder (VAE) model (Arvanitidis et al., 2018; Chen et al., 2018; Shao et al., 2018; Arvanitidis et al., 2019) and show that geodesic paths result in more natural interpolations. Some extend this approach to learn VAEs with priors that are estimated using the Riemannian metrics computed in the latent space (Arvanitidis et al., 2021; Kalatzis et al., 2020).

Lie Group Operators A Lie group is a group of continuous transformations which also defines a manifold by representing infinitesimal transformations that can be applied to input data (Hoffman, 1966; Dodwell, 1983). Several methods incorporate Lie groups into neural networks to represent data transformations that are identity-preserving within the model (Cohen & Welling, 2014; Cohen et al., 2018). A prevalent strategy is to learn a dictionary of Lie group operators that are mapped to a specific group element through the matrix exponential $\text{expm}(\cdot)$ (Rao & Ruderman, 1999; Miao & Rao, 2007; Culpepper & Olshausen, 2009; Sohl-Dickstein et al., 2010; Cohen & Welling, 2014; Connor & Rozell, 2020; Connor et al., 2021). In these models, each operator $\mathbf{\Psi}_m$, called a *transport operator*, describes a single direction along the manifold and is parameterized by a single coefficient c_m. Given an initial data point $\mathbf{z}$, the transport operators define a generative model where transformations can be derived from sampling sparse coefficients $c_m \sim \text{Laplace}\,(0, \zeta)$:

$$\widehat{\mathbf{z}} = \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m\right)\mathbf{z} + n, \tag{1}$$

where $n \sim \mathcal{N}(0, \sigma_n^2 \mathbf{I})$.

The manifold autoencoder (MAE) incorporates the transport operator model into the latent space of an autoencoder to learn a dictionary of operators that represent the global, nonlinear manifold structure in the latent space (Connor & Rozell, 2020). This model has been shown to learn operators effectively with transformation supervision, and it will provide the context for demonstrating the effectiveness of the methods developed in this paper. Specifically, we demonstrate a method for learning natural data variations in the MAE latent space using a new strategy for selecting point pairs without transformation labels, as well as a method for learning the regions of the manifold where each operator is likely to be used.

3 Methods

The MAE learns a low-dimensional latent representation of the data by defining an encoder function $f : \mathcal{X} \to \mathcal{Z}$ that maps high-dimensional data points $\mathbf{x} \in \mathbb{R}^D$ to low-dimensional latent vectors $\mathbf{z} \in \mathbb{R}^d$ and a decoder function $g : \mathcal{Z} \to \mathcal{X}$ that maps the latent vectors back into the data space (Connor & Rozell, 2020). Transport operators $\mathbf{\Psi}$ are incorporated into the latent space to learn manifold-based transformations. Before learning the transport operators, the autoencoder is pretrained to extract a latent representation of the data using the traditional autoencoder reconstruction objective.

After pretraining, the autoencoder weights are fixed and the operators are trained with the following objective, which encourages the learning of transport operators that generate efficient paths between the latent vectors $\mathbf{z}_0$ and $\mathbf{z}_1$ (coinciding with $f(\mathbf{x}_0)$ and $f(\mathbf{x}_1)$) that are nearby on the manifold:

$$\mathcal{L}_\Psi = \frac{1}{2}\left\|\mathbf{z}_1 - \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m\right)\mathbf{z}_0\right\|_2^2 + \frac{\gamma}{2}\sum_m \|\mathbf{\Psi}_m\|_F^2 + \zeta\|\mathbf{c}\|_1, \tag{2}$$

where $\gamma, \zeta > 0$.

Objective (2) is minimized via an alternating minimization scheme, as in the linear sparse coding model Olshausen & Field (1997). Specifically, at each training iteration, points pairs $\mathbf{x}_0$ and $\mathbf{x}_1$ are selected in the input space and encoded into the latent space $\mathbf{z}_0$ and $\mathbf{z}_1$. Then, coefficients $\mathbf{c}$ are inferred between the encoded latent vectors (by minimizing (2) with respect to $\mathbf{c}$) to estimate the best path between $\mathbf{z}_0$ and $\mathbf{z}_1$. After inference, the coefficients are fixed and a gradient step is taken to minimize (2) with respect to

transport operator weights. Once learned, these operators represent different types of motion that traverse the manifold, and they can be combined to generate natural paths on the manifold.

After fitting transport operators to the latent space of a fixed autoencoder network, there is a final fine-tuning training phase that updates the network weights and transport operators simultaneously using a joint objective that combines the autoencoder reconstruction loss with $\mathcal{L}_\Psi$. This fine-tuning step has been shown to dramatically improve the quality of learned transformations Connor & Rozell (2020) and addresses the potential mismatch between the data manifold and the learned latent structure by adapting the latent structure to fit the transport operators learned between the selected training point pairs. We empirically find that breaking up training into these three phases increases the stability with which the transport operators can be learned. Given the context of this MAE model, in the following sections we describe our contributions.

3.1 Unsupervised Transformation Learning

There are many possible strategies for selecting training point pairs for a Lie group operator model, and the choice of a strategy can dictate the quality of transformations that are learned. For example, point pairs may be selected as random samples from the dataset or from within the same class. These points, however, are likely to be outside of local manifold neighborhoods and may not provide representations of the natural, perceptually smooth variations in the dataset. Alternatively, point pairs can be selected as those with the lowest Euclidean distance in the pixel or autoencoder latent space. Both of these approaches are limited, however, since nearest neighbors in high dimensions has been noted as ineffective Aggarwal et al. (2001) and since unsupervised autoencoder features may not capture semantic transformations of interest through Euclidean distance. In some applications, there may be additional information about the dataset that can aid in point pair selection, such as rotation angle, semantic label, or time index in a temporal sequence. However, for most complex datasets, information about the transformations of interest is not available. This necessitates a strategy for selecting training point pairs which can learn natural transformations in complex datasets without requiring additional transformation supervision.

We generalize transport operator training to incorporate an unsupervised learning strategy that can be applied to a wide array of datasets. In particular, we select point pairs using a perceptual loss metric (Johnson et al., 2016). The perceptual loss measures the distance between feature representations of input images that are extracted from a classifier pretrained on a large dataset, like ImageNet. It has been shown that features in the penultimate layer of such pretrained classifiers serve as an effective surrogate for visual similarity (Johnson et al., 2016; Yosinski et al., 2015). Given an initial point $\mathbf{x}_0$, we define point pairs in a feature embedding $r(\mathbf{x})$ as:

$$\mathbf{x}_1 = \underset{\mathbf{x}\in\mathcal{D}}{\operatorname{argmin}} \, \|r(\mathbf{x}_0) - r(\mathbf{x})\|_2^2, \tag{3}$$

where, in practice, we randomly select $\mathbf{x}_1$ from the N nearest points to $\mathbf{x}_0$ in the classifier feature space.

With the proposed approach, we are capable of training transport operators without labels to select point pairs. Furthermore, we find that the perceptual loss results in transport operators that correspond to smooth, semantically meaningful transformations. We compare the effect of training with the proposed approach versus other point pair strategies in Appendix E.

3.2 Learning Local Transport Operator Statistics to Encourage Identity Preservation

A trained transport operator model provides a dictionary of operators that can be applied globally across the entire data space. This model has flexibility to define local manifold characteristics through the coefficients $\mathbf{c}$ which specify the combination of operators to apply to a given point. The standard prior on the transport operator coefficients is a factorial Laplace distribution parameterized by a single scale parameter ζ, which encourages transformations to be defined by a sparse set of operators:

$$p_\zeta(\mathbf{c}) = \prod_{m=1}^{M} \frac{1}{2\zeta} \exp\left(-\frac{|c_m|}{\zeta}\right). \tag{4}$$

By setting a fixed prior on coefficients across the entire manifold, there is an implicit assumption that the manifold structure is the same over the whole dataset and every operator is equally likely to be applied at every point. However, this is a flawed assumption because not all transformations in one class of data are present in all other classes and even within classes there may be regions with different manifold structure. Not capturing the local statistics of transport operator usage could result in transformed points that depart from the data manifold and change their identity.

To address this limitation, we introduce a coefficient encoder network which maps latent vectors to scale parameters that specify the Laplace distribution on the coefficients at those points in the latent space. The goal of this network is to estimate local coefficient distributions that maximize the identity-preservation of points that are transformed with operators characterized by randomly sampled coefficients:

$$\widehat{\mathbf{z}} = T_{\Psi}(\mathbf{c})\mathbf{z} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_{\epsilon}^2 I), \tag{5}$$

where $T_{\Psi}(\mathbf{c}) = \text{expm}\left(\sum_{m=1}^{M} \Psi_m c_m\right)$ is the transformation matrix corresponding to the transport operators (parameterized by coefficients $\mathbf{c}$).

Given labeled observations $(\mathbf{z}, y)$ and a pretrained classifier $r(\cdot)$, we aim to learn a network $q_{\phi}(\mathbf{c} \mid \mathbf{z})$ that outputs parameters that can be used to produce augmented samples $\widehat{\mathbf{z}}$ without changing the classifier output $r(\widehat{\mathbf{z}}) = y$.

To maximize the likelihood of obtaining the labeled classifier output for augmented samples, we adapt the concept of consistency regularization from semi-supervised learning. Consistency regularization is applied when training a classifier in a semi-supservised setting in order to ensure that known identity-preserving augmentations cause only small changes in the classifier output (Bachman et al., 2014). In our context, we use a pretrained classifier and a dictionary of transport operators and we want to find a distribution on the coefficients at individual input points that results in consistent classification outputs when $T_{\Psi}(\mathbf{c})$ is applied to inputs. The specific objective can be chosen from a variety of loss functions that encourage similarity in classifier probability outputs. We specifically minimize the KL-divegence between the label y and the label of the transformed output $r(\widehat{\mathbf{z}})$.

Unfortunately, the consistency regularization objective can be trivially minimized by setting $\mathbf{c} = 0$, resulting in an identity transformation and the same classifier output. Therefore, we want to encourage the largest coefficient values possible while maintaining the identity of our initial data point. This motivates the addition of a KL-divergence regularizer that encourages the distribution $q_{\phi}(\mathbf{c}|\mathbf{z})$ to be similar to a specified Laplace prior with a fixed scale ζ like in (4). Our final objective for training the coefficient encoder network is:

$$E = D_{KL}(y \| r(\widehat{\mathbf{z}})) + D_{KL}\left(q_{\phi}(\mathbf{c} \mid \mathbf{z}) \| p_{\zeta}(\mathbf{c})\right). \tag{6}$$

A more detailed derivation of this training objective can be found in Appendix A.

The addition of this coefficient encoder is a principled way to build identity-preservation directly into our model and to identify local manifold characteristics throughout the latent space. Since this model is trained using outputs from a pretrained classifier, the resulting encoded coefficient scale weights can be informative about classifier invariances. Specifically, we can quantify which operators are associated with large encoded coefficient scale weights in different parts of the latent space, indicating that the classifier is invariant to those transformations. Additionally, we can identify points or regions of space that have small encoded coefficient scale weights, indicating they are near class boundaries and can undergo only small transformations without changing identity.

3.3 Decreasing Computational Complexity of Coefficient Inference

To compute the gradient for the matrix exponential during training and inference, previous works have used a linear approximation (Rao & Ruderman, 1999), learned the operators in a factored form (Sohl-Dickstein et al., 2010), or used analytic gradients that are not favorable for parallel computations (Culpepper & Olshausen, 2009; Connor & Rozell, 2020; Connor et al., 2021). The main computational bottleneck for learning transport

operators is the coefficient inference step, where objective (2) is minimized with respect to $\mathbf{c}$. The ℓ_1 norm term is non-smooth, leading to slower convergence when using subgradients found via automatic differentiation (Boyd & Vandenberghe, 2004).

In this work, we employ a forward-backward splitting scheme (Beck & Teboulle, 2009) that applies automatic differentiation for the matrix exponential term, with numerical approximations that are more favorable for parallel hardware Bader et al. (2019), and proximal gradients for the ℓ_1 norm (Parikh & Boyd, 2014). We show in Appendix B (with further details on the coefficient inference algorithm) that this leads to significant speedup during training, and allows for scaling of transport operators to complex datasets that were not possible before. To the best knowledge of the authors, this is the first work to demonstrate learning of Lie group operators on datasets as large and complex as CelebA.

4 Analysis

For our experiments, we examine the ability of our proposed approaches to enable the MAE model to learn natural transformations from complex datasets where the underlying identity-preserving transformations are not easily identifiable. We also highlight the benefits of incorporating the coefficient encoder network that captures the transport operator local usage statistics by encoding coefficient scale values that best maintain the identity of latent vectors.

We work with three datasets: MNIST (LeCun et al., 1998), Fashion MNIST (Xiao et al., 2017), and CelebA (Liu et al., 2015). We select MNIST and Fashion MNIST because they contain several classes that share natural transformations but they do not have transformation labels. We select CelebA to highlight our ability learn natural transformations in a larger, more complex dataset. As a classic dataset used in papers that aim to disentangle dataset features (Higgins et al., 2017; Chen et al., 2016; Hu et al., 2018; Lin et al., 2019), CelebA contains semantically meaningful natural transformations that may be amenable to qualitative labeling.

In all experiments, we followed the general training procedure put forth previously (Connor & Rozell, 2020) by separating the network training into three phases: the autoencoder training phase, the transport operator training phase, and the fine-tuning phase. We select training point pairs that are nearest neighbors in the feature space of the final, pre-logit layer of a ResNet-18 (He et al., 2015) classifier pretrained on ImageNet (Russakovsky et al., 2015). After completely training the MAE, we fix the autoencoder network weights and transport operator weights and train the coefficient encoder network with the objective derived in Section 3.2. Additional details on the datasets, network architectures, and training procedure are available in the Appendix.

We compare against the contractive autoencoder (CAE) (Rifai et al., 2011c) and β-VAE (Higgins et al., 2017), two other autoencoder based methods that incorporate data structure into the latent space. The CAE represents another technique that learns a manifold representation in a neural network latent space. In their model, the manifold is represented by estimated manifold tangent planes at latent point locations. The β-VAE learns to disentangle factors of variation along latent dimensions through an increase in the weighted penalty on the KL-divergence term in the VAE objective. We choose these methods as they provide approaches to learn natural dataset variations in the latent space without transformation labels.

4.1 Learning Natural Data Variations

First we show how well our proposed methods enable the MAE model to learn natural data variations in the dataset when selecting point pairs using the perceptual point pair selection strategy and we highlight the usefulness of a nonlinear manifold model for generating latent space paths. Fig. 1 shows the paths generated by transport operators trained in this model. The image in the middle column in each block of images is the reconstructed version of the input image $\mathbf{x}_0$. The images to the left and right of the middle show the reconstructed images generated by an individual learned operator applied to the encoded latent vector $\mathbf{z}_0$: $\mathbf{z}_c = \text{expm}(\mathbf{\Psi}_m c)\mathbf{z}_0$, $c = -N_c, ..., N_c$. We see an individual operator generates a similar transformation across multiple inputs, and in many cases, the transformations induced by the transport operators are semantically meaningful. We also show that the perceptual point pair selection strategy is effective over a range of datasets.

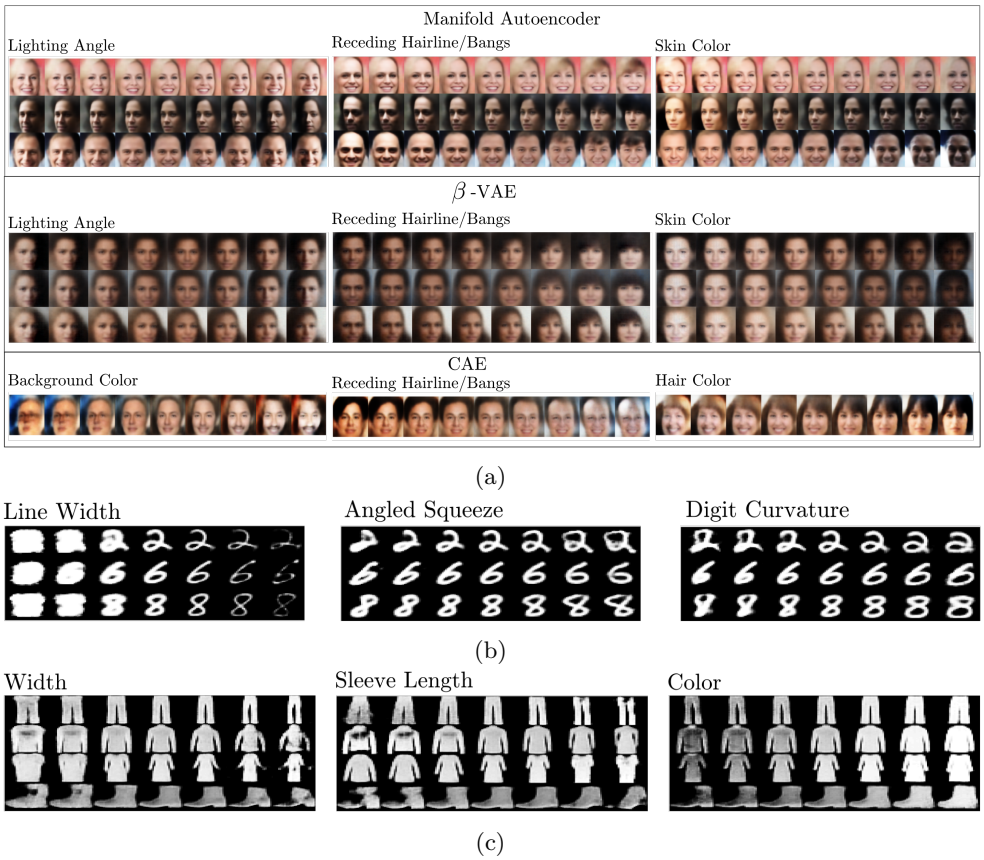


Figure 1: Paths generated by applying a subset of learned transport operators on three datasets. In each figure, images in the middle column of the image block are the reconstructed inputs and images to the right and left are images decoded from transformed latent vectors in positive and negative directions, respectively. (a) Comparing the transformations generated by three learned transport operators to transformations generated by β-VAE and CAE. The transport operators learn semantically meaningful transformations similar to the disentangled β-VAE representation, while maintaining a higher resolution in image outputs. (b-c) Transport operators learned using the perceptual point pair selection strategy generate natural transformations on both MNIST and Fashion MNIST.

In Fig. 1, we compare the transport operator paths to paths generated by the CAE and the β-VAE. The CAE-generated paths represent the directions of motion on the tangent planes estimated at individual points. The β-VAE paths are generated by varying the value of one latent dimension, while the others remain fixed. While our method and the β-VAE learn several qualitatively similar transformations, our method is capable of doing so without significantly sacrificing reconstruction performance. In the Appendix, we include more examples of the transformations generated by each learned transport operator across each dataset.

While many of the operators can be assigned a semantic label through qualitative visual inspection, the CelebA dataset has attribute labels for the images which enable a quantitative analysis of the connection between learned transport operators and dataset attributes. To classify attributes, we fine-tune a ResNet-18 pretrained on ImageNet with 16 classification heads for attributes including smile, beard, hair color, and pale skin (Mao et al., 2020). With this classifier, we are able to quantify the degree with which specific transport operators correspond to dataset attributes. Fig. 2 shows the classification outputs of the attribute classifier for specific transport operators. Our model learns operators that vary hair color, skin paleness, and several others attributes, as shown in Appendix K.

One benefit of Lie group operator models over other manifold-based models is that the learned transport operators provide a natural way for estimating nonlinear manifold paths between points in the latent space.

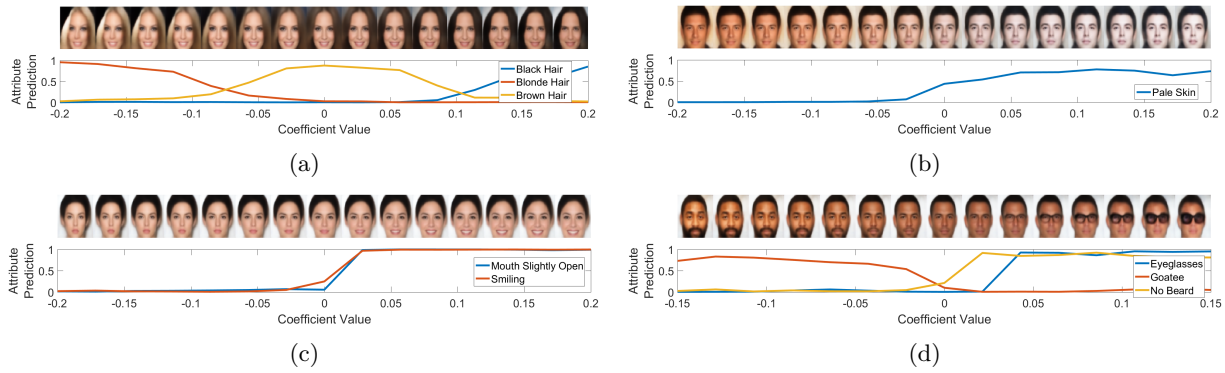


Figure 2: Paths generated by the application of two learned transport operators with the associated attribute classifier probability outputs for the transformed images. Our model learns operators that correspond to meaningful CelebA dataset attributes like (a) hair color (b) pale skin (c) smiling (d) sunglasses/beard

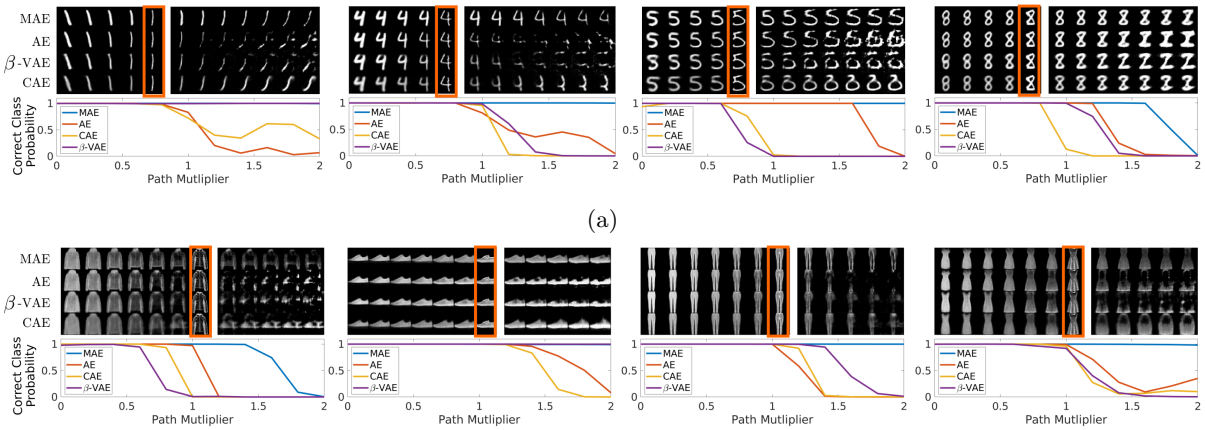
To highlight this benefit, in Fig. 3 we compare the MAE-based interpolated and extrapolated paths to those estimated using CAE and β-VAE, as well as linear paths in our original autoencoder before fine-tuning it to fit the manifold structure defined by the learned transport operators. With each method, we estimate a path between the latent vectors associated with two points $\mathbf{z}_0$ and $\mathbf{z}_1$, interpolate between the two points, and extrapolate beyond $\mathbf{z}_1$ to extend the path. For MAE, this path is estimated by inferring the set of transport operator coefficients $\mathbf{c}^*$ between the latent points and then generating the path: $\mathbf{z}_t = \text{expm}\left(\sum_{m=1}^{M} \Psi_m c_m^* t\right) \mathbf{z}_0$. When the path multiplier t is between 0 and 1 that indicates interpolation and path multipliers beyond 1 indicate extrapolation. To estimate paths in AE, CAE and β-VAE, we compute the vector difference between $\mathbf{z}_0$ and $\mathbf{z}_1$ and interpolate and extrapolated on that vector direction. In these figures, the first block of images corresponds to the interpolated path with the selected final point $\mathbf{x}_1$ surrounded in an orange box. The second block of images corresponds to the extrapolated paths.

For quantifying the identity preservation of each transformation, we input each generated image into a pretrained classifier and plot the probability of the class label associated with the inputs. All four methods perform interpolation effectively but our trained model estimates the extrapolated paths more accurately. Fig. 4 shows how that accuracy varies during extrapolation sequences for 4000 samples. The MAE is better at generating extrapolated outputs that maintain class identity. The lower classification accuracy in fashion MNIST is due to both a more challenging dataset and the lower resolution of autoencoder image outputs when compared to input images.

4.2 Learning Local Manifold Structure

After the MAE is trained, we have a dictionary of transport operators that describe manifold transformations and a network with a latent space that is adapted to the manifold. We then train the coefficient encoder from Section 3.2 to estimate the coefficient scale weights as a function of points in the latent space. To visualize how the use of the transport operators varies over the latent space, we generate an Isomap embedding (Tenenbaum et al., 2000) of latent vectors and color each point by the encoded scale parameter for coefficients associated with each of the transport operators. Fig. 5a and 5b shows these embeddings for MNIST and FMNIST, respectively. Each operator has regions of the latent space where their use is concentrated.

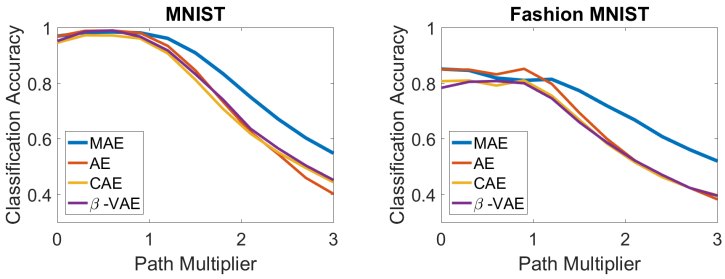
By training the coefficient encoder to maximize the similarity between the classification outputs for an input sample and for a transformed version of that sample, the network aids in identifying which transport operators can be applied to inputs in regions of the data space without changing the identity of the input. This helps significantly with data augmentation where the goal is to create new samples with in-class variations. To highlight this benefit, in Fig. 6 we show samples augmented by applying transport operators with randomly sampled coefficients to an input latent vector. In each block of images, the leftmost image (in a green box) is the input image and the images to the right are decoded augmentations. The top row shows samples



(a)

(b)

Figure 3: Identity preservation of transformed paths as quantified by a pretrained classifier output. In the figures on the top, the first block of images corresponds to the interpolated path with selected final point x_1 surrounded in an orange box. The second block of images corresponds to the extrapolated paths. Below the images are plots of the probability of the class label associated with the inputs $\mathbf{z}_0$ and $\mathbf{z}_1$. A path multiplier between 0 and 1 indicates interpolation and path multipliers beyond 1 indicate extrapolation. (a) MNIST (b) Fashion MNIST



(a) (b)

Figure 4: The average accuracy of the classifier output for images at each point along the interpolation/extrapolation sequence. When the path multiplier is between 0 and 1 that indicates interpolation and path multipliers beyond 1 indicate extrapolation. (a) MNIST (b) Fashion MNIST

augmented with transport operators controlled by coefficients sampled from Laplace distributions with encoded coefficient scale weights. The bottom row shows samples augmented with transport operators controlled by coefficients sampled from Laplace distributions with a fixed scale parameter. While both strategies generate some realistic variations of the data, using the encoded scale weights improves identity-preservation of the transformed output. The augmentations with the encoded scale weights are better at maintaining the identity of the sampled points.

Because the coefficient encoder uses a pretrained classifier to determine the coefficient distributions in the latent space, the resulting encoded scale weights can inform us about the types of manifold transformations the classifier is invariant to. Fig. 5c and 5d shows the average coefficient scale weights for each class (rows) and each transport operator (columns) for MNIST and FMNIST, respectively. From this, one can identify classes for which the classifier is both sensitive and robust to transformations, represented by small and large scale weights respectively. We can also examine which classes share the use of the same transformations. The images to the right of in Fig. 5c show transport operators being applied to samples with high encoded scale weights (in a yellow box) and samples with low encoded scale weights (in a blue box). It is interesting to examine the characteristics of transformations that are better suited to some classes than others. For instance, operator 3 in Fig. 5c increases the curve at the bottom of digits. This is a natural transformation for classes 3, 5, and 6 which all have higher coefficient scale weights for this operator, but when this is applied to a 1, that makes it look like an 8.

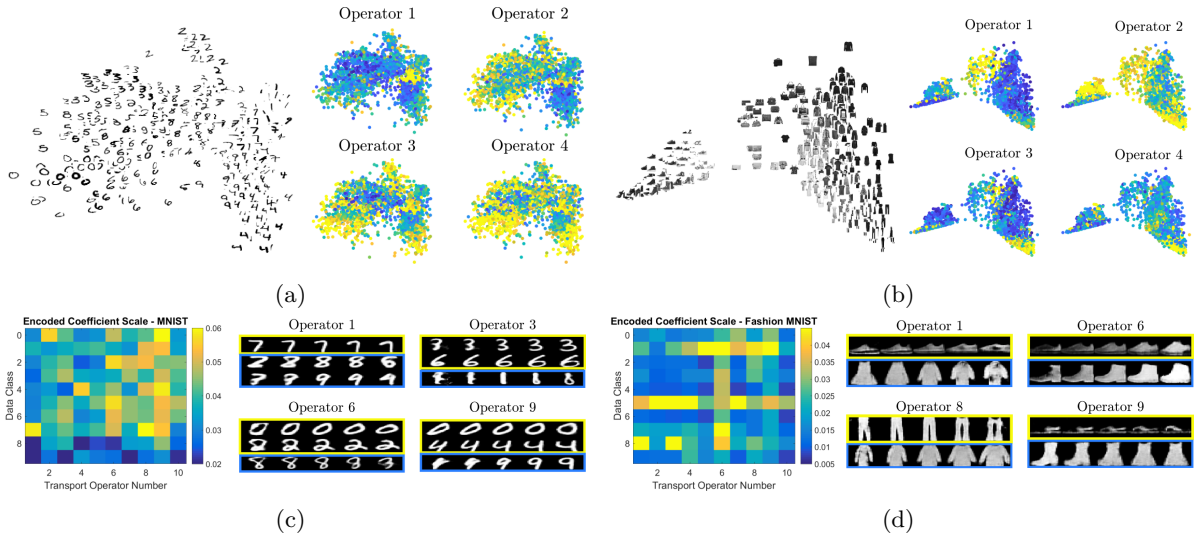


Figure 5: Visualizations of the encoded coefficient scale weights for (a, c) MNIST and (b, d) Fashion-MNIST. Figure best viewed zoomed in. (a, b): Isomap embedding of the latent vectors with input images overlaid. The scatter plots on the right show the same Isomap embedding colored by the encoded coefficient scale weights for several operators (yellow indicates large scale weights and blue small scale weights). We see operators whose use is localized in regions of the manifold space. (c, d): The average coefficient scale weights for each class on each transport operator. High scale weights for a given operator (yellow) indicate it can be applied to a given class without easily changing identity. The images on the right show examples of the operators applied to classes with high encoded scale weights (in the top yellow boxes) and classes with low encoded scale weights (in the bottom blue boxes). The examples with low coefficient scale weights change classes more easily than other examples.

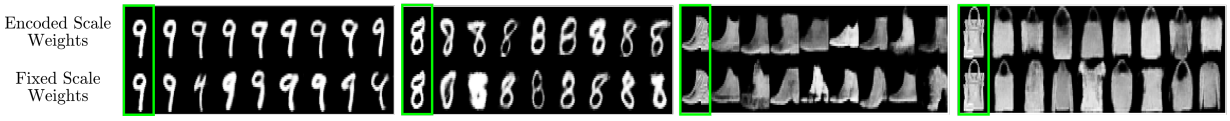


Figure 6: Examples of samples generated by transport operators using coefficients sampled with encoded scale weights (top row) and with a fixed scale weight (bottom row). Images in the green box are the input images and the remaining images in each row are transformed outputs.

5 Conclusion

In this work, we develop methods to improve the effectiveness and utility of Lie group operator models for learning manifold representations of datasets with complex transformations that cannot be labeled. We do this by introducing a perceptual point pair selection strategy for training operators and by developing a method that uses a pretrained classifier to learn local regions where operators are likely to be used while preserving the identity of transformed samples. We demonstrate the efficacy of our approach in learning natural dataset variations with the MAE. While this is a powerful model, users should be mindful of the biases that are introduced through training with specific supervision methods when drawing conclusions about learned data transformations. This work presents a promising technique for learning representations of natural data variations that can improve model robustness. In future work, we can address some limitations by expanding this work to consider methods for localizing manifold structure with limited or no class labels and improving the reconstruction fidelity by applying the model within more complex generative model embeddings.

References

Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu (eds.), *Database Theory — ICDT 2001*, pp. 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44503-6.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations*, 2018.

Georgios Arvanitidis, Soren Hauberg, Philipp Hennig, and Michael Schober. Fast and robust shortest paths on manifolds learned from data. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1506–1515. PMLR, 2019.

Georgios Arvanitidis, Bogdan Georgiev, and Bernhard Schölkopf. A prior-based approximate latent riemannian metric. *arXiv preprint arXiv:2103.05290*, 2021.

Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. *arXiv preprint arXiv:1412.4864*, 2014.

Philipp Bader, Sergio Blanes, and Fernando Casas. Computing the matrix exponential with an optimized taylor polynomial approximation. *Mathematics*, 7(12), 2019. ISSN 2227-7390. doi: 10.3390/math7121174. URL `https://www.mdpi.com/2227-7390/7/12/1174`.

Amir Beck and Marc Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, January 2009. ISSN 1936-4954. doi: 10.1137/080716542. URL `http://epubs.siam.org/doi/10.1137/080716542`.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

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

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. The curse of dimensionality for local kernel machines. *Techn. Rep*, 1258:12, 2005.

Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, UK ; New York, 2004. ISBN 978-0-521-83378-3.

Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1550. PMLR, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets, 2016.

Taco Cohen and Max Welling. Learning the irreducible representations of commutative lie groups. In *International Conference on Machine Learning*, pp. 1755–1763. PMLR, 2014.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *International Conference on Learning Representations*, 2018.

Marissa Connor and Christopher Rozell. Representing closed transformation paths in encoded network latent space. In *AAAI Conference on Artificial Intelligence*, volume 34, pp. 3666–3675, 2020.

Marissa Connor, Gregory Canal, and Christopher Rozell. Variational autoencoder with learned latent structure. In *International Conference on Artificial Intelligence and Statistics*, pp. 2359–2367. PMLR, 2021.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Benjamin J Culpepper and Bruno A Olshausen. Learning transport operators for image manifolds. In *Advances in Neural Information Processing Systems*, pp. 423–431, 2009.

Peter C Dodwell. The lie transformation group model of visual perception. *Perception & Psychophysics*, 34 (1):1–16, 1983.

Piotr Dollár, Vincent Rabaud, and Serge J Belongie. Learning to traverse image manifolds. In *Advances in Neural Information Processing Systems*, pp. 361–368, 2007.

Luca Falorsi, Pim de Haan, Tim R Davidson, and Patrick Forré. Reparameterizing distributions on lie groups. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3244–3253. PMLR, 2019.

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

Manuel Gil. *On Rényi divergence measures for continuous alphabet sources*. PhD thesis, Citeseer, 2011.

Søren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John Fisher, and Lars Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Artificial Intelligence and Statistics*, pp. 342–350. PMLR, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pp. 2731–2741. PMLR, 2019.

William C Hoffman. The lie algebra of visual perception. *Journal of mathematical Psychology*, 3(1):65–98, 1966.

Qiyang Hu, Attila Szabó, Tiziano Portenier, Matthias Zwicker, and Paolo Favaro. Disentangling factors of variation by mixing them, 2018.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016.

Dimitris Kalatzis, David Eklund, Georgios Arvanitidis, and Søren Hauberg. Variational autoencoders with riemannian brownian motion priors. In *International Conference on Machine Learning*, 2020.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2013.

Abhishek Kumar, Prasanna Sattigeri, and P Thomas Fletcher. Semi-supervised learning with gans: Manifold invariance with improved inference. *Advances in Neural Information Processing Systems*, 2017.

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.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 2019.

Jianxin Lin, Zhibo Chen, Yingce Xia, Sen Liu, Tao Qin, and Jiebo Luo. Exploring explicit domain supervision for latent space disentanglement in unpaired image-to-image translation, 2019.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 1:1–48, 2008. URL `http://www.cs.toronto.edu/~hinton/absps/tsne.pdf`.

Longbiao Mao, Yan Yan, Jing-Hao Xue, and Hanzi Wang. Deep multi-task multi-label cnn for effective facial attribute classification, 2020.

Xu Miao and Rajesh PN Rao. Learning the lie groups of visual invariance. *Neural computation*, 19(10): 2665–2693, 2007.

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.

PACE. *Partnership for an Advanced Computing Environment (PACE)*, 2017. URL `http://www.pace.gatech.edu`.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014. ISSN 2167-3888. doi: 10.1561/2400000003. URL `https://doi.org/10.1561/2400000003`.

Mijung Park, Wittawat Jitkrittum, Ahmad Qamar, Zoltán Szabó, Lars Buesing, and Maneesh Sahani. Bayesian manifold learning: The locally linear latent variable model (ll-lvm). In *Advances in Neural Information Processing Systems*, pp. 154–162, 2015.

Rajesh PN Rao and Daniel L Ruderman. Learning lie groups for invariant visual perception. In *Advances in neural information processing systems*, pp. 810–816, 1999.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286. PMLR, 2014.

Salah Rifai, Yann N Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. *Advances in Neural Information Processing Systems*, 24:2294–2302, 2011a.

Salah Rifai, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin, and Xavier Glorot. Higher order contractive auto-encoder. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 645–660. Springer, 2011b.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning*, 2011c.

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

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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*, pp. 315–323, 2018.

Jascha Sohl-Dickstein, Ching Ming Wang, and Bruno A Olshausen. An unsupervised algorithm for learning lie group transformations. *arXiv preprint arXiv:1001.1027*, 2010.

Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2000.

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

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

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

A Coefficient Encoder Details

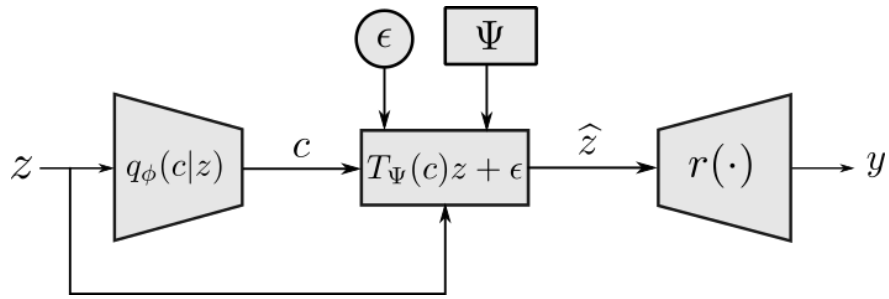


Figure 7: System diagram for learning local coefficient statistics.

Our motivation in training an encoder to learn the coefficient statistics is to understand the regions of the manifold in which specific transport operators are applicable and the invariances of a pretrained classifier to the learned transport operator-induced transformations. In this section, we will outline a derivation of our coefficient encoder training objective from a variational inference perspective by learning a variational posterior for the coefficients using a deep neural network (Kingma & Welling, 2013; Rezende et al., 2014). Given observations $(\mathbf{z}_i, y_i)$ for $i = 1, \ldots, N$, and a pre-trained classifier $r(\cdot)$, we aim to learn a distribution $q_\phi(\mathbf{c}|\mathbf{z})$ from which we can sample to define the transformation $T_\Psi(\mathbf{c})$ that can be applied to $\mathbf{z}$ while maintaining class identity. In other words, we want the augmented $\widehat{\mathbf{z}} = T_\Psi(\mathbf{c})\mathbf{z} + \epsilon$ to result in $r(\widehat{\mathbf{z}}) = y$.

To encourage identity-preservation of vectors transformed with transport operators that are controlled by sampled coefficients, we maximize the likelihood of a class output for an observation under the system model diagrammed in Figure 7. This system connects the observed latent vector $\mathbf{z}$ to an output y through a transformation defined by $\mathbf{c}$. We follow the derivation used in Rezende & Mohamed (Rezende & Mohamed, 2016) which introduces the variational posterior to estimate the log likelihood of the data.

Consider a parameterized distribution for the conditional likelihood of our observations:

$$\begin{aligned}
\log p_\theta(y \mid \mathbf{z}) &= \log \mathbb{E}_{p_\zeta(\mathbf{c})}\left[p_\theta(y \mid \mathbf{c}, \mathbf{z})\right] && (7)\\
&= \log \int_{\mathbf{c}} p_\zeta(\mathbf{c}) p_\theta(y \mid \mathbf{c}, \mathbf{z}) d\mathbf{c} && (8)\\
&= \log \int_{\mathbf{c}} q_\phi(\mathbf{c} \mid \mathbf{z}) \frac{p_\zeta(\mathbf{c})}{q_\phi(\mathbf{c} \mid \mathbf{z})} p_\theta(y \mid \mathbf{c}, \mathbf{z}) d\mathbf{c} && (9)\\
&\geq \int_{\mathbf{c}} q_\phi(\mathbf{c} \mid \mathbf{z}) \log \left[\frac{p_\zeta(\mathbf{c})}{q_\phi(\mathbf{c} \mid \mathbf{z})} p_\theta(y \mid \mathbf{c}, \mathbf{z})\right] d\mathbf{c} && (10)\\
&= E_{q_\phi}\left[\log p_\theta(y \mid \mathbf{c}, \mathbf{z})\right] - D_{KL}\left(q_\phi(\mathbf{c} \mid \mathbf{z}) | p_\zeta(\mathbf{c})\right) && (11)\\
&= E_{q_\phi}\left[\log E_\epsilon p_\theta(y \mid \mathbf{c}, \mathbf{z}, \epsilon, \widehat{\mathbf{z}})\right] - D_{KL}\left(q_\phi(\mathbf{c} \mid \mathbf{z}) | p_\zeta(\mathbf{c})\right). && (12)
\end{aligned}$$

In (7) and (8), we marginalize over the coefficients $\mathbf{c}$ as required under our system model in Figure 7. In (10), we lower bound the conditional likelihood with a variational lower bound derived from Jensen's inequality. Finally, we use the reparameterization trick to define $\widehat{\mathbf{z}}$ as a deterministic function of $\mathbf{c}$, $\mathbf{z}$, and the parameter-free random variable ϵ in (12).

In order to sample from $q_\phi(\mathbf{c} \mid \mathbf{z})$ when computing the expectations, we first use the reparameterization trick (Connor et al., 2021; Kingma & Welling, 2013; Rezende et al., 2014) to define the sampled coefficients $\widehat{\mathbf{c}}$ as a function of a uniform random variable $\mathbf{u} \sim \text{ Unif}\left(-\frac{1}{2}, \frac{1}{2}\right)^M$:

$$\widehat{\mathbf{c}} = l_\phi(\mathbf{u}, \mathbf{z}) = -h_\phi(\mathbf{z}) \operatorname{sgn}(\mathbf{u}) \log(1 - 2\,|\mathbf{u}|), \tag{13}$$

where l_ϕ is a defined mapping from a uniform distribution to a Laplace distribution and the Laplace scale parameters are defined by the output of the coefficient encoder h_ϕ that we aim to learn.

Using the reparameterization of $\widehat{\mathbf{c}}$ and $\widehat{\mathbf{z}}$, we can define the expectation:

$$E_{q_\phi}\left[\log E_\epsilon p_\theta(y \mid \mathbf{c}, \mathbf{z}, \epsilon, \widehat{\mathbf{z}})\right] = E_{\mathbf{u}}\left[\log E_\epsilon p_\theta(y \mid \widehat{\mathbf{c}} = l_\phi(\mathbf{u}, \mathbf{z}), \epsilon, \widehat{\mathbf{z}})\right] \tag{14}$$

$$\approx \frac{1}{JK}\sum_{j=1}^{J}\log\sum_{k=1}^{K} p_\theta\left(y|\widehat{\mathbf{c}}^{(j)} = l_\phi(\mathbf{u}^{(j)}, \mathbf{z}), \epsilon^{(k)}, \widehat{\mathbf{z}}\right). \tag{15}$$

Our pre-trained classifier defines p_θ which allows us to specify final objective using of $r(\cdot)$. Using a KL-divergence as the likelihood function between the ground truth labels y_i and the classifier output for the augmented inputs $r(\widehat{\mathbf{z}}_i^{(j)})$ and simplifying the model by setting ϵ to 0, we get:

$$\log p_\theta(y \mid \mathbf{z}) \geq -\frac{1}{J}\sum_{j=1}^{J} D_{KL}\left(y_i \mid r(\widehat{\mathbf{z}}_i^{(j)})\right) - D_{KL}\left(q_\phi(\mathbf{c} \mid \mathbf{z}) | p_\zeta(\mathbf{c})\right), \tag{16}$$

where in practice we use a single sample $J = 1$, resulting in the objective in (6) which we want to minimize. The KL divergence between the $q_\phi(\mathbf{c} \mid \mathbf{z})$ and $p_\zeta(\mathbf{c})$ has a closed form expression (Gil, 2011):

$$D_{KL}\left(q_\phi(\mathbf{c} \mid \mathbf{z}) \| p_\zeta(\mathbf{c})\right) = \log(h_\phi(\mathbf{z})) - \log(\zeta) + \frac{\zeta}{h_\phi(\mathbf{z})} - 1 \tag{17}$$

B Coefficient Inference Details

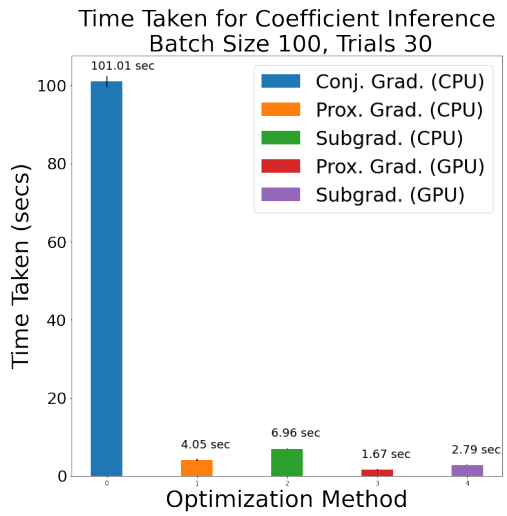


Figure 8: Comparison in time taken for various coefficient inference algorithms on fully trained transport operators. Each inference trial is performed over random batches of 100 point pairs drawn from the CelebA dataset (with a latent dimension of 32). 30 trials are run with the same $\mathbf{c}_0$ and tolerance. We used the proximal gradient (GPU) method.

In order to perform coefficient inference, previous Lie group operator methods (Connor & Rozell, 2020; Connor et al., 2021) use the analytic gradient proposed in (Culpepper & Olshausen, 2009) with a conjugate gradient descent solver. This method requires an eigenvalue decomposition for computing the gradient which is not favorable for parallel computations. Alternatively, we use the PyTorch implementation of the matrix exponential that allows for automatic differentiation. Furthermore, to handle the non-smooth ℓ_1 norm in objective (2), we apply a proximal gradient step in a forward-backward splitting scheme (Beck & Teboulle,

2009). For the ℓ_1 norm, the proximal gradient has a known, closed-form solution in the soft-thresholding function (Parikh & Boyd, 2014):

$$\mathcal{T}_\lambda(\mathbf{c}) = \text{sign}(\mathbf{c}) * \max\Big(|\mathbf{c}| - \lambda, 0\Big) \tag{18}$$

Let $\nabla_{\mathbf{c}} \frac{1}{2} \left\| \mathbf{z}_1 - \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m\right) \mathbf{z}_0 \right\|_2^2 \approx \nabla \widetilde{f}(\mathbf{c})$ be a numerical approximation to the gradient of the ℓ_2 term, found through automatic differentiation. Given initial coefficient values $\mathbf{c}_0$ drawn from an isotropic Gaussian with variance 4×10^{-4}, our gradient descent step is:

$$\mathbf{c}_{k+1} = \mathcal{T}_{\zeta\alpha_k}\left(\mathbf{c}_k - \alpha_k \nabla \widetilde{f}(\mathbf{c}_k)\right) \tag{19}$$

These steps are iterated upon until either a max iteration count is reached, or the change in coefficients, $\|\mathbf{c}_{k+1} - \mathbf{c}_k\|_2$, falls below some threshold. For all experiments we use a max iteration count of 800 and a tolerance of 1×10^{-5}. For our step-size, we use $\alpha_k = (0.985)^k \alpha_0$ with $\alpha_0 = 1 \times 10^{-2}$. We experimented with different acceleration methods (Kingma & Ba, 2015; Beck & Teboulle, 2009), and found that in many cases they resulted in worse performance.

The main parameter to select for inference is the sparsity-inducing parameter on the coefficient prior, ζ. This value is sensitive to the dataset, latent dimension, and γ hyper-parameter value. In practice, we fix a γ value and pick ζ to be as high as possible, inducing sparsity in the inferred coefficients, while maintaining good reconstruction performance of $\widehat{\mathbf{z}}_1$. If ζ is too large, that will result in all coefficients going to zero, preventing proper reconstructive performance. On the other hand, setting ζ to be too small will result in all operators being used to represent the transformation between each point pair, preventing any meaningful structure from being learned during training. Parameters used in specific training runs are included in their respective Appendix sections.

When compared against the coefficient inference implementation from (Connor & Rozell, 2020), we perform inference for a batch of 100 samples in an average of 1.67 seconds over 100 trials whereas the inference from (Connor & Rozell, 2020) took an average of 101.01 seconds. We also compare the speed-up over a baseline that strictly uses automatic differentiation (subgradients) and compare the benefits from moving to GPU hardware in Figure 8. We use the proximal gradient (GPU) method from this figure. Experiments were run on a machine with an Intel i7-6700 CPU with 4.00 GHz and a Nvidia TITAN RTX.

C Training Strategy

In all experiments, we follow the general training procedure put forth previously (Connor & Rozell, 2020). We train the MAE with three training phases: the autoencoder training phase, the transport operator training phase, and the fine-tuning phase. During the autoencoder training phase, the network weights are updated using a reconstruction loss objective: $E_{\text{AE}} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$.

During the transport operator training phase, the network weights are fixed and the transport operators are trained between pairs of points using the objective (2). Pairs of images x_0, x_1 are chosen using the perceptual point pair selection strategy described in Section 3.1. The images are then encoded into the latent space z_0, z_1. For each batch, the first step is to infer the coefficients between all pairs of latent vectors. Coefficient inference is best performed when the entries of the latent vectors are close to the range $[-1, 1]$. Because of this, we define a scale factor that can be applied to encoded latent vectors to reduce the magnitude of their entries prior to performing coefficient inference. In practice, we inspect the latent vector magnitudes after the autoencoder training phase and choose a scale that will adjust the magnitudes of the latent vector entries to be in the range $[-1, 1]$. This does not have to be a precise range for the latent vector magnitudes but instead is a practical guideline. Coefficient inference is performed on the scaled latent vectors as described in Sections 3.3 and Appendix B. After the coefficients are inferred for a batch, the weights on the dictionary elements are updated. This phase of training is performed until the loss values reach a plateau and the dictionary magnitudes plateau.

The fine-tuning training phase begins after the transport operator training phase is complete. In this phase, both the network weights and the transport operator weights are updated using the joint objective:

$$E = \lambda \left(\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|_2^2 + \|\mathbf{x}_1 - \hat{\mathbf{x}}_1\|_2^2 \right) + (1 - \lambda) E_\Psi, \tag{20}$$

where E_Ψ is defined in (2). Using this objective, we alternate between taking steps on the transport operator weights while the network weights are fixed and taking steps on the network weights while the transport operator weights are fixed. Additionally, during the fine-tuning phase we incorporate occasional steps in which we update the network weights using only the reconstruction loss to ensure effective image reconstruction.

In most cases, it is necessary to reduce the γ parameter in front of the Frobenius norm dictionary regularizer prior to fine-tuning or the dictionary magnitudes will reduce to zero. We report the γ we use for transport operator training and fine-tuning in the experimental details sections below. It may also be necessary to decrease the network learning rate during fine-tuning.

The coefficient encoder training requires a network that is trained to classify data from our selected dataset. We train this classifier using training data from a given dataset. For datasets with worse autoencoder reconstruction quality, we train the classifier in the latent space. Otherwise, we train the classifier on images in the data space. With the MAE and classifier trained, we train the coefficient encoder network following the strategy described in Section 3.2 and Appendix A.

In order to train the CAE, we use the same autoencoder architecture used with the MAE with the addition of a Frobenius norm regularizer on the encoder Jacobian, weighted by a selected λ value (different from the λ in (20)). The Jacobian is computed using PyTorch automatic differentiation. We find the Jacobian norm decreases to the same value irrespective of our choice of λ, leading us to choose $\lambda = 1$. For the β-VAE we use the same architectures outlined in (Higgins et al., 2017) with $\beta = 10$ for CelebA and $\beta = 5$ for MNIST and FMNIST. We find that setting β any higher results in poor reconstructive performance. Scripts for training both comparison methods are included in the code repository.

Hyper-parameter tuning for all experiments was performed on the Georgia Tech Partnership for Advanced Computing Environment (PACE) clusters (PACE, 2017). Experiments were performed using a Nvidia Quadro RTX 6000. Runs training the CAE, and β-VAE on CelebA were all run on a separate machine with a Nvidia TITAN RTX.

D Parameter Selection

The MAE model has several hyperparameters that must be tuned and we will provide guidance to determining ideal parameter values for our experiments and future experiments. First, we will describe some signs to look out for to identify if a run is succeeding or failing. One indicator that we compute is the transport operator difference which is:

$$E_{\Psi\text{diff}} = \frac{1}{2} \left\| \mathbf{z}_1 - \operatorname{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m \right) \mathbf{z}_0 \right\|_2^2 - \frac{1}{2} \left\| \mathbf{z}_1 - \operatorname{expm}\left(\sum_{m=1}^{M} \widehat{\mathbf{\Psi}}_m c_m \right) \mathbf{z}_0 \right\|_2^2, \tag{21}$$

where $\widehat{\mathbf{\Psi}}_m$ is the dictionary after the gradient step is taken. A gradient step should decrease the transport operator objective meaning the $E_{\Psi\text{diff}}$ value should be positive for an effective step. If there are many gradient steps that result in negative $E_{\Psi\text{diff}}$ values, that indicates that the parameters are not optimal or that the learning rate is too large. This is an important metric to observe during fine-tuning to determine whether to select a smaller γ or smaller network learning rate.

Signs of failure of a training run with selected training parameters:

- All operator magnitudes reduce towards zero.
- All inferred coefficients between point pairs are zero.

- Most of the operators generate transformation paths with latent values that increase quickly to infinity.
- Many steps have negative values for $E_{\Psi\text{diff}}$.
- The operator magnitudes increase which results in unstable training steps with NaN values in the computed objective.

Dictionary regularizer parameter The dictionary regularizer parameter γ is the weight on the Frobenius norm term in (2). This objective term serves two purposes. First, it balances the effect of the coefficient sparsity regularizer with the parameter ζ. If ζ is large, the sparsity regularizer encourages small coefficient magnitudes and one way to achieve that while still effectively inferring paths is to increase the magnitude of the operators. The Frobenius norm term must have a large enough influence to counterbalance this force or the operators will increase in magnitude to the point of being unstable. If a run is becoming unstable, we recommend decreasing ζ or increasing γ.

The second purpose of the dictionary regularizer term is to identify which operators are necessary to represent the transformations on the data manifold. If an operator is not being used to represent a transformation between $\mathbf{z}_0$ and $\mathbf{z}_1$ then the dictionary regularizer reduces its magnitude to zero. Therefore, during training we are able to estimate the model order based on how many dictionary elements remain non-zero. In our tests we vary γ between 2×10^{-8} and 2×10^{-4}. If γ is too small, it will not counterbalance the coefficient sparsity term and the dictionaries will grow to unstable magnitudes. If γ is too large it will reduce the magnitude of all operators to zero. During the fine-tuning steps, we have often found it necessary to reduce the γ because, with a larger γ, both the operator magnitudes and the latent vector magnitudes can decrease substantially, which leads to an ineffective manifold model.

Coefficient sparsity parameter The coefficient sparsity parameter ζ in (2) controls the sparsity of the coefficients that are used to estimate paths between $\mathbf{z}_0$ and $\mathbf{z}_1$. We run tests with values of ζ between 0.005 and 2. From dataset to dataset the ideal value varies. If ζ values are too small then all the operators are used to represent all the paths between point pairs. The ζ should be increased so fewer than M coefficients are used for each inferred path. When ζ is too large, all the coefficients go to zero during inference. This means there is no path inferred between $\mathbf{z}_0$ and $\mathbf{z}_1$ because of overweighting the sparsity constraint.

Number of dictionary elements As mentioned above, the dictionary regularizer acts as a model order selection tool so our strategy for selecting number of dictionary elements M is to increase the number of dictionary elements until some of their magnitudes begin reducing to zero during training. This indicates that some of the operators are not necessary for representing transformations.

Latent dimension The latent dimension of the autoencoder is selected to ensure quality reconstructed image outputs.

Relative weight of reconstruction and transport operator objectives The parameter λ determines the weight of the reconstruction term relative to the transport operator term. We observe good performance of the model for λ between 0.5 and 0.75.

E Comparing Point Pair Selection Strategies

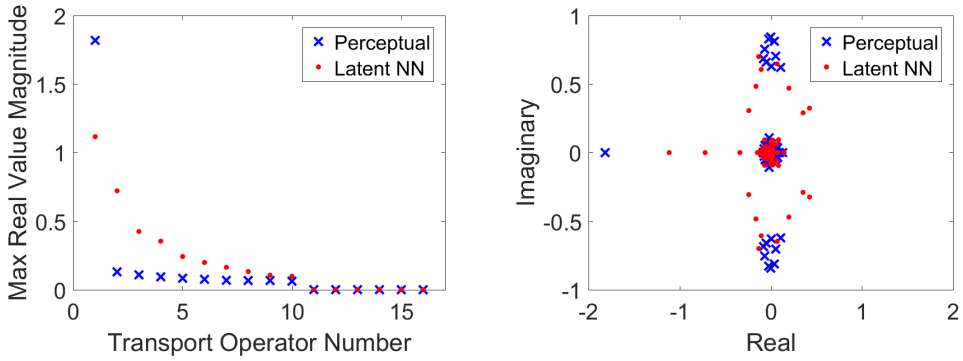
In the past, transport operators have been trained using point pairs that were selected randomly from the same class (Connor et al., 2021) or with some knowledge of transformation labels (Connor & Rozell, 2020). In this work, we establish a method that learns natural transformations without requiring transformation labels using a perceptual point pair selection strategy. The perceptual loss metric enables us to select point pairs that may share semantically meaningful transformations without being exactly the same. Fig. 9 shows examples of nearest neighbors selected using pixel similarity, similarity in the latent space, and the perceptual loss metric. This highlights how the perceptual loss metric can be useful for identifying inputs with similar qualitative characteristics. For instance, on the left, the perceptual loss metric identifies another bald man as

a nearest neighbor which has the similar characteristics of the initial image even though the exact image looks quite different.

Figure 9: Examples of nearest neighbors identified through pixel similarity, latent similarity, and the perceptual loss metric. The images in the green boxes are the initial reference images and the images to the right of those show three selected nearest neighbors. The perceptual loss metric identifies neighbors that share similar characteristics like hair style without being exactly the same.

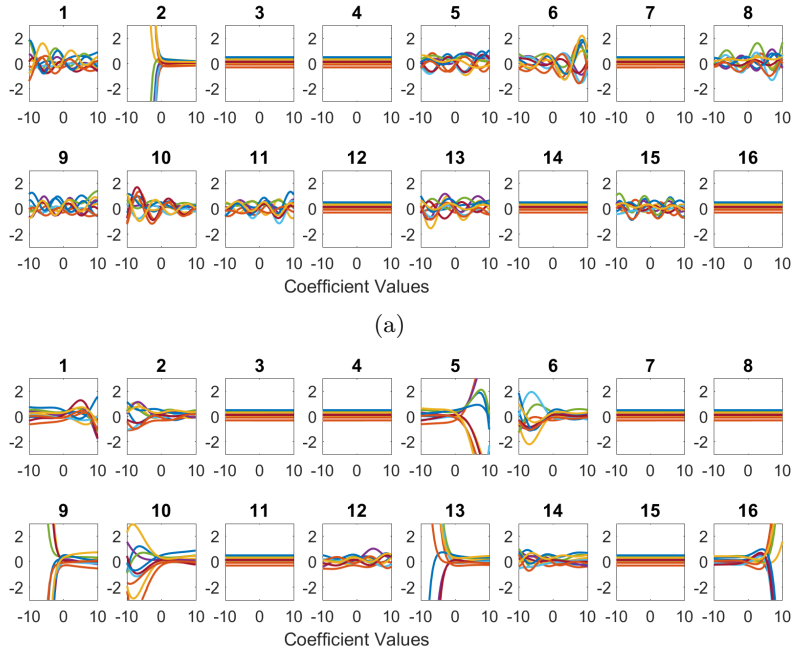
When examining the success of learned operators, one characteristic we care about is how well the operators maintain stability of generated paths. We have defined this as a useful characteristic for identity-preservation of applied operators – if operators expand latent vectors to infinity, that will very likely lead to a change in identity. Each of the operators can be viewed as the dynamics matrix of a linear dynamical systems model and we can analyze their stability by observing their eigenvalues (Strogatz, 2000). In our model, because the transport operators relate to natural data transformations without a temporal component, it does not matter if the transport operators are applied with positive or negative coefficients. Therefore, general stability or instability of the dynamical system associated with a given transport operator has the same impact of causing the latent vectors to increase to infinity because our coefficients can be either positive- or negative-valued. In this setting, we define the stability of our transport operator system in the context of marginal stability of a dynamical system. Marginally stable dynamical systems generate cyclic transformations that never increase or decrease the magnitude of the inputs. A marginal stable system has only imaginary eigenvalues with no real parts (Strogatz, 2000). Therefore, we can identify transport operators approaching marginal stability by investigating the magnitude of the real parts of their eigenvalues and specifically the eigenvalue with the maximum magnitude of its real part. When the maximum magnitude of the real part for all eigenvectors associated with an operator is close to zero, that indicates that the operator is closer to marginal stability. Therefore, we quantify the stability of transport operator paths by looking at the maximum magnitude of real parts of eigenvalues associated with each transport operator. Fig 10a. shows the sorted maximum magnitude of the real parts of eigenvalues in each of the 16 operators learned in the MNIST experiment when using the perceptual point pair selection strategy and when using a simple strategy of selecting point pairs as nearest neighbors in the latent space. Except for one operator, all the operators trained using latent space similarity for supervision have larger maximum magnitudes of real eigenvalue components than the operators trained using the perceptual loss metric. This indicates that the operators trained using the latent space similarity to select training point pairs are farther from marginal stability and can be seen as less stable by our definition of transport operator stability.

To view the effect of these operators more intuitively, we plot the values of the latent dimensions as individual operators are applied in Fig. 11. The paths of the lines in each of these plots show a lot about how the operators influences each latent dimension. Fig 11a shows the paths generated by transport operators trained with the perceptual point pair selection strategy. The plots with straight lines show the effect of operators whose magnitudes are reduced to 0 during training. Most of the operators learned using the perceptual point pair selection strategy are close to cyclic except for operator 2 (the operator with the large real component magnitude in Fig. 10a). In contrast, several of the operators trained with point pairs selected as nearest neighbors in the latent space extend to infinity (Fig 11b). This can explain the larger maximum real value magnitudes in Fig. 10a. This analysis leads us to conclude that the perceptual point pair selection strategy is more likely to yield identity-preserving transport operators.



(a) (b)

Figure 10: Analysis of eigenvalues of operators learned in the MNIST experiment. (a) The maximum absolute value of the real parts of eigenvalues computed from each learned operator. (b) Plot of the real and imaginary parts of the eigenvalues for each operator.



(a)

(b)

Figure 11: Visualizations of the effect of each learned transport operator on each dimension of an encoded latent vector. In each plot, each line represents a single latent dimension and the coefficient magnitude of the transformation varies on the x-axis. (a) Paths from operators learned with perceptual point pair supervision. (b) Paths from operators learned with point pairs selected as nearest neighbors in the latent space.

F MNIST Experiment Details

The MNIST dataset is made available under the terms of the Creative Commons Attribution-Share Alike 3.0 license. 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 images. The traditional MNIST testing set is used for our testing set. The input images are normalized so their pixel values are between 0 and 1. The network architecture used for the autoencoder is shown in

Table 1. The training parameters for the transport operator training phase and the fine-tuning phase are shown in Tables 2 and 3.

Prior to training the coefficient encoder for the MNIST dataset, we train a classifier on the labeled MNIST image data which we use to encourage identity-preservation during coefficient encoder training. The training parameters for the coefficient encoder are shown in Table 4. The image classifier we use is based on the simple LeNet architecture with two convolutional layers and three fully connected layers (LeCun et al., 1998).

Table 1: Network Architecture for MNIST and Fashion 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
BatchNorm: feat: 64	ReLU
ReLU	convTranpose: chan: 64, kern: 4, stride: 1, pad: 1
conv: chan: 64, kern: 4, stride: 2, pad: 1	BatchNorm: feat: 64
BatchNorm: feat: 64	ReLU
ReLU	convTranpose: chann: 64, kern: 4, stride: 2, pad: 2
conv: chan: 64, kern: 4, stride: 1, pad: 0	BatchNorm: feat: 64
BatchNorm: feat: 64	ReLU
ReLU	convTranpose: chan: 1, kernel: 4, stride: 2, pad: 1
Linear: 2 Units	Sigmoid

Table 2: Training parameters for the transport operator training phase of the MNIST experiment

MNIST Transport Operator Training Parameters
batch size: 250
autoencoder training epochs: 300
transport operator training epochs: 50
latent space dimension (z_{dim}): 10
M : 16
lr_{net} : 10^{-4}
lr_{Ψ} : 10^{-3}
ζ : 0.1
γ : 2×10^{-6}
initialization variance for Ψ: 0.05
number of restarts for coefficient inference: 1
nearest neighbor count: 5
latent scale: 30

Table 3: Training parameters for the fine-tuning phase of the MNIST experiment

MNIST Fine-tuning Parameters
batch size: 250
transport operator training epochs: 100
lr_{net} : 10^{-4}
lr_{Ψ} : 10^{-3}
ζ : 0.1
γ : 2×10^{-6}
λ: 0.75
number of network update steps: 50
number of Ψ update steps: 50

Table 4: Training parameters for the MNIST Coefficient Encoder

MNIST Coefficient Encoder Parameters
batch size: 250
training epochs: 300
lr: 10^{-3}
ζ_{prior} : 0.1
λ_{kl}: 0.5
coefficient spread scale: 0.1
classifier domain: image

G MNIST Experiment Additional Results

Here we show additional experimental details and results for the MNIST experiment. Fig. 12 shows the magnitude of all 16 operators after the fine-tuning phase. Six of the operators have their magnitudes reduced to zero. Fig. 13 shows the paths generated by transport operators trained on MNIST data.

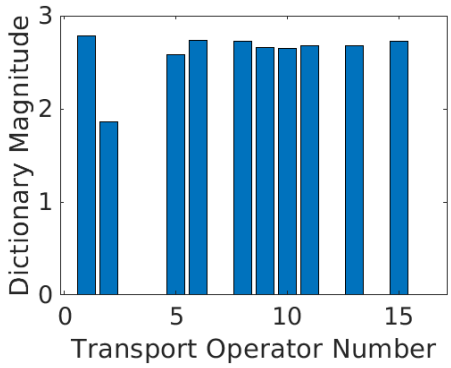


Figure 12: The magnitudes of the learned operators after fine-tuning.

H Fashion-MNIST Experiment Details

The Fashion-MNIST dataset is made available under the terms of the MIT license. We split the Fashion MNIST dataset into training, validation, and testing sets. The training set contains 50,000 images from the Fashion-MNIST training set. The validation set is made up of the remaining 10,000 images. The traditional Fashion-MNIST testing set is used for our testing set. The input images are normalized so their pixel values are between 0 and 1. The network architecture used for the autoencoder is the same as in the MNIST experiment and it is shown in Table 1. The training parameters for the transport operator training phase and the fine-tuning phase are shown in Tables 5 and 6.

Prior to training the coefficient encoder for the Fashion-MNIST dataset, we train a classifier the latent vectors associated with labeled Fashion-MNIST data which we use to encourage identity-preservation during coefficient encoder training. The training parameters for the coefficient encoder are shown in Table 7. The latent vector classifier has a simple architecture of Linear(512), ReLU, Linear(10), Softmax.

Table 5: Training parameters for the transport operator training phase of the Fashion-MNIST experiment

Fashion-MNIST Transport Operator Training Parameters
batch size: 200
autoencoder training epochs: 300
transport operator training epochs: 50
latent space dimension (z_{dim}): 10
M : 16
lr_{net} : 10^{-4}
lr_{Ψ} : 10^{-3}
ζ : 0.5
γ : 2×10^{-5}
initialization variance for Ψ: 0.05
number of restarts for coefficient inference: 1
nearest neighbor count: 5
latent scale: 30

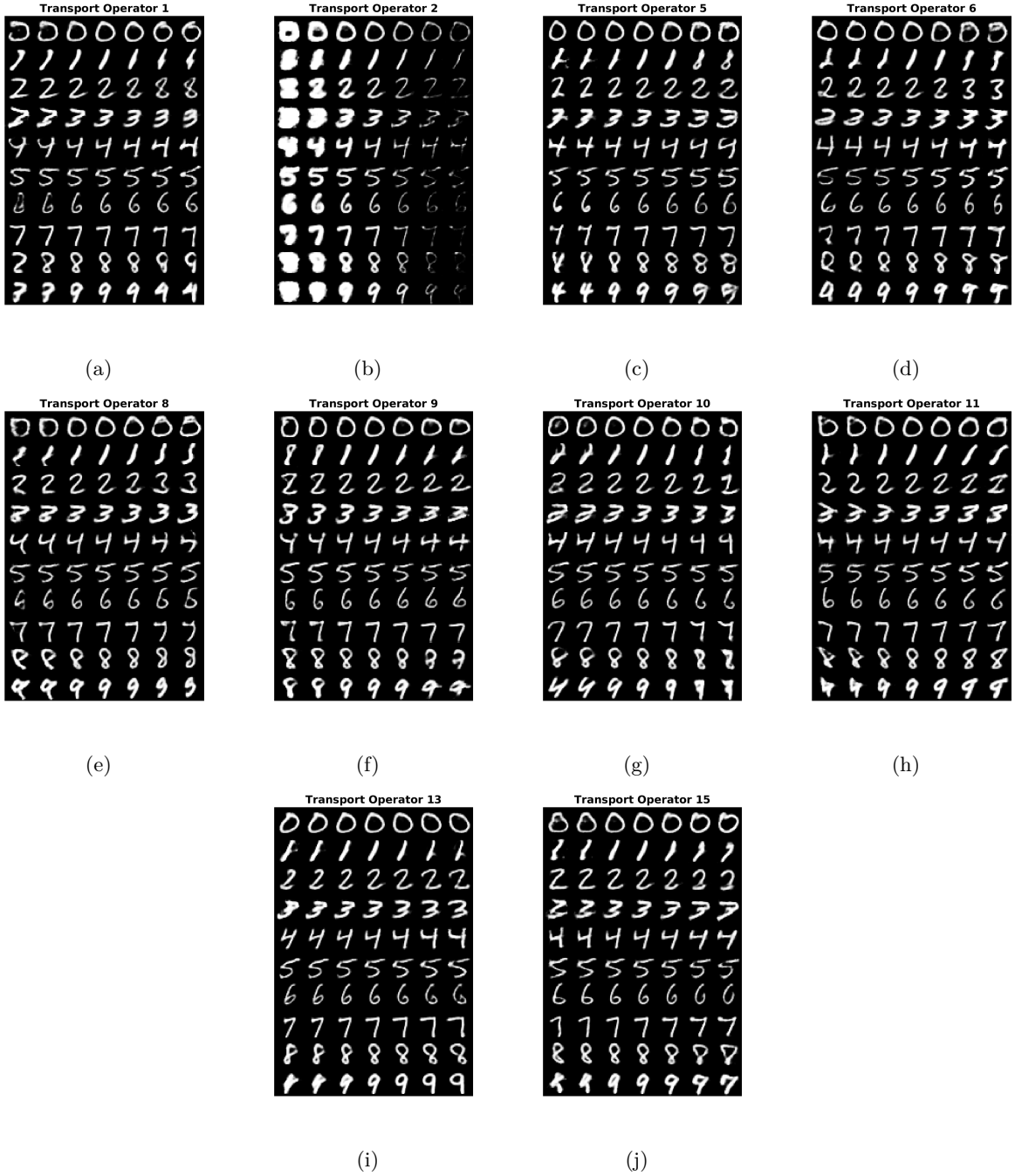


Figure 13: Paths generated by all non-zero transport operators trained on the MNIST dataset. Images in the middle column of the image block are the reconstructed inputs and images to the right and left are images decoded from transformed latent vectors in positive and negative directions, respectively

I Fashion MNIST Experiment Additional Results

Here we show additional experimental details and results for the Fashion-MNIST experiment. Fig. 14 shows the magnitude of all 16 operators after the fine-tuning phase. Six of the operators had their magnitudes reduced to zero.

Table 6: Training parameters for the fine-tuning phase of the Fashion-MNIST experiment

Fashion-MNIST Fine-tuning Parameters
batch size: 200
transport operator training epochs: 150
lr_{net} : 10^{-4}
lr_{Ψ} : 10^{-3}
ζ : 0.5
γ : 2×10^{-6}
λ: 0.75
number of network update steps: 50
number of Ψ update steps: 50

Table 7: Training parameters for the Fashion-MNIST Coefficient Encoder

Fashion-MNIST Coefficient Encoder Parameters
batch size: 200
training epochs: 300
lr: 10^{-3}
ζ_{prior} : 0.5
λ_{kl}: 0.5
coefficient spread scale: 0.1
classifier domain: latent

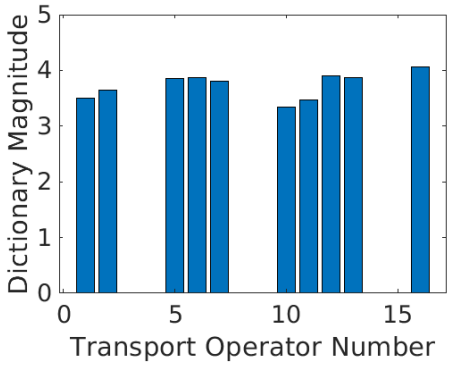


Figure 14: The resulting magnitude of the learned operators after fine-tuning the MAE.

To visualize how the use of the transport operators varies over the latent space, we generate an Isomap embedding (Tenenbaum et al., 2000) of latent vectors and color each point by the encoded scale parameter for coefficients associated with each of the transport operators. Fig. 5b shows these embeddings for Fashion-MNIST data. Each operator has regions of the latent space where their use is concentrated. Fig. 5d shows the average coefficient scale weights for each class (rows) and each transport operator (columns) for Fashion-MNIST. There are some classes like trouser and sandal (classes 1 and 5) which have large encoded coefficient scale weights for most of the transport operators. This means they are robust to many natural transformations. Other classes like coat and shirt (classes 4 and 6) have smaller encoded coefficient scale weights which means they are sensitive to many transformations. The images to the right in Fig. 5d show transport operators being applied to samples with high encoded scale weights (in a yellow box) and samples with low encoded scale weights (in a blue box). Fig. 15 shows the paths generated by non-zero transport operators trained on Fashion-MNIST data.

J CelebA Experiment Details

The CelebA dataset is publicly available for non-commercial research purposes. We split the CelebA dataset into training and testing sets. The training set contains the first 150,000 images accompanied with the entire test set. The input images are normalized so their pixel values are between 0 and 1. The network architecture used for the autoencoder is shown in Table 8. The training parameters for the transport operator training phase and the fine-tuning phase are shown in Table 9.

The attribute classifier is a ResNet-18 model adapted from `https://github.com/d-li14/face-attribute-prediction` with 16 classifier heads after the layer with 512 hidden units. Each classifier head corresponds to a single attribute and is modeled as a fully-connected linear layer with 256 hidden units, followed by batch

normalization, dropout, and ReLU layers. Afterwards, two logits are output for a 0/1 prediction for each classifier output. The training procedure, including the dynamic loss weighting, follows (Mao et al., 2020).

Table 8: Network Architecture for CelebA Experiments

Encoder Network	Decoder Network
Input $\in \mathbb{R}^{64\times 64}$	Input $\in \mathbb{R}^{32}$
conv: chan: 32 , kern: 4, stride: 2, pad: 1	Linear: 80,000 Units
BatchNorm: feat: 32	ReLU
ReLU	convTranpose: chan: 256, kern: 3, stride: 1, pad: 0
conv: chan: 64, kern: 4, stride: 2, pad: 1	BatchNorm: feat: 256
BatchNorm: feat: 64	ReLU
ReLU	convTranpose: chann: 256, kern: 3, stride: 1, pad: 0
conv: chan: 128, kern: 3, stride: 2, pad: 1	BatchNorm: feat: 256
BatchNorm: feat: 128	ReLU
ReLU	convTranpose: chan: 256, kernel: 3, stride: 1, pad: 1
conv: chan: 256, kern: 3, stride: 1, pad: 1	BatchNorm: feat: 256
BatchNorm: feat: 128	ReLU
ReLU	convTranpose: chan: 128, kernel: 3, stride: 1, pad: 1
conv: chan: 256, kern: 4, stride: 2, pad: 1	BatchNorm: feat: 128
BatchNorm: feat: 256	ReLU
ReLU	convTranpose: chan: 128, kernel: 3, stride: 1, pad: 0
conv: chan: 128, kern: 4, stride: 2, pad: 1	BatchNorm: feat: 128
BatchNorm: feat: 128	ReLU
ReLU	convTranpose: chan: 3, kernel: 4, stride: 2, pad: 0
Linear: 32 Units	Sigmoid

Table 9: Training parameters for the CelebA experiment

CelebA Transport Operator Training Parameters
batch size: 500
autoencoder training epochs: 300
transport operator training epochs: 50
latent space dimension (z_{dim}): 32
M : 40
lr_{net} : 10^{-4}
lr_{Ψ} : 10^{-3}
ζ : 1.5
γ : 1×10^{-5}
initialization variance for Ψ: 0.05
number of restarts for coefficient inference: 1
nearest neighbor count: 5
latent scale: 2

CelebA Fine-tuning Parameters
batch size: 500
fine-tuning epochs: 10
lr_{net} : 10^{-4}
lr_{Ψ} : 10^{-3}
ζ : 0.8
γ : 5×10^{-7}
λ: 0.75
number of network update steps: 50
number of Ψ update steps: 50

K CelebA Experiment Additional Results

Here we show additional experimental results for the CelebA experiment. Fig. 16 shows the paths generated by the 40 transport operators trained on celebA data. Each row represents a different operator acting on the same input image. Images in the middle column of the image block are the reconstructed inputs and images to the right and left are images decoded from transformed latent vectors in positive and negative directions, respectively. Fig. **??** shows the classification outputs of the attribute classifier for example transport operators. These two operators that vary smiling, beard, and sunglasses attributes.

Fig. 17 shows an interesting feature in our learned operators – many of them generate cyclic paths that begin and end at nearly the same point. Also, in these cyclic sequences, the transformations seem to lead to a change in gender. The image sequence in Fig. 17a shows the path generated by a single operator. The image in the middle is the reconstructed input image and the images to the left and right are the paths generated by negative and positive coefficients respectively. This operator changes the hairline and quantity of bangs. As we apply the transport operator with a negative coefficient (to the left of center), the woman gains bangs and then becomes a man with a moustache on the far left of the image. As we apply the transport operator with a positive coefficient (to the right of center), the woman's forehead gets higher and then she becomes a man with a high forehead and eventually, on the far right the woman transforms into a man with bangs, similar to the man on the far left. The similarity between the generated images on the far left and far right is notable because this indicates the transformation path is nearly closed. Fig. 17b shows the change in five of the 32 latent dimensions over the generated path. These paths have a nearly cyclic structure.

This is particularly interesting because in (Connor & Rozell, 2020) they learn closed transformation paths by selecting point pairs on known cyclic transformation paths and highight the benefit of the transport operator model for representing these types of paths. In this case, we learn this cyclic path with only perceptual point pair supervision. Additionally, this identifies a benefit of the transport operator approach over other models of the manifold in a neural network latent space. We can learn nonlinear paths that keep the generated points in the same neighborhood in the latent space without extending to infinity which is inevitable when linear paths that are used to represent natural transformations.

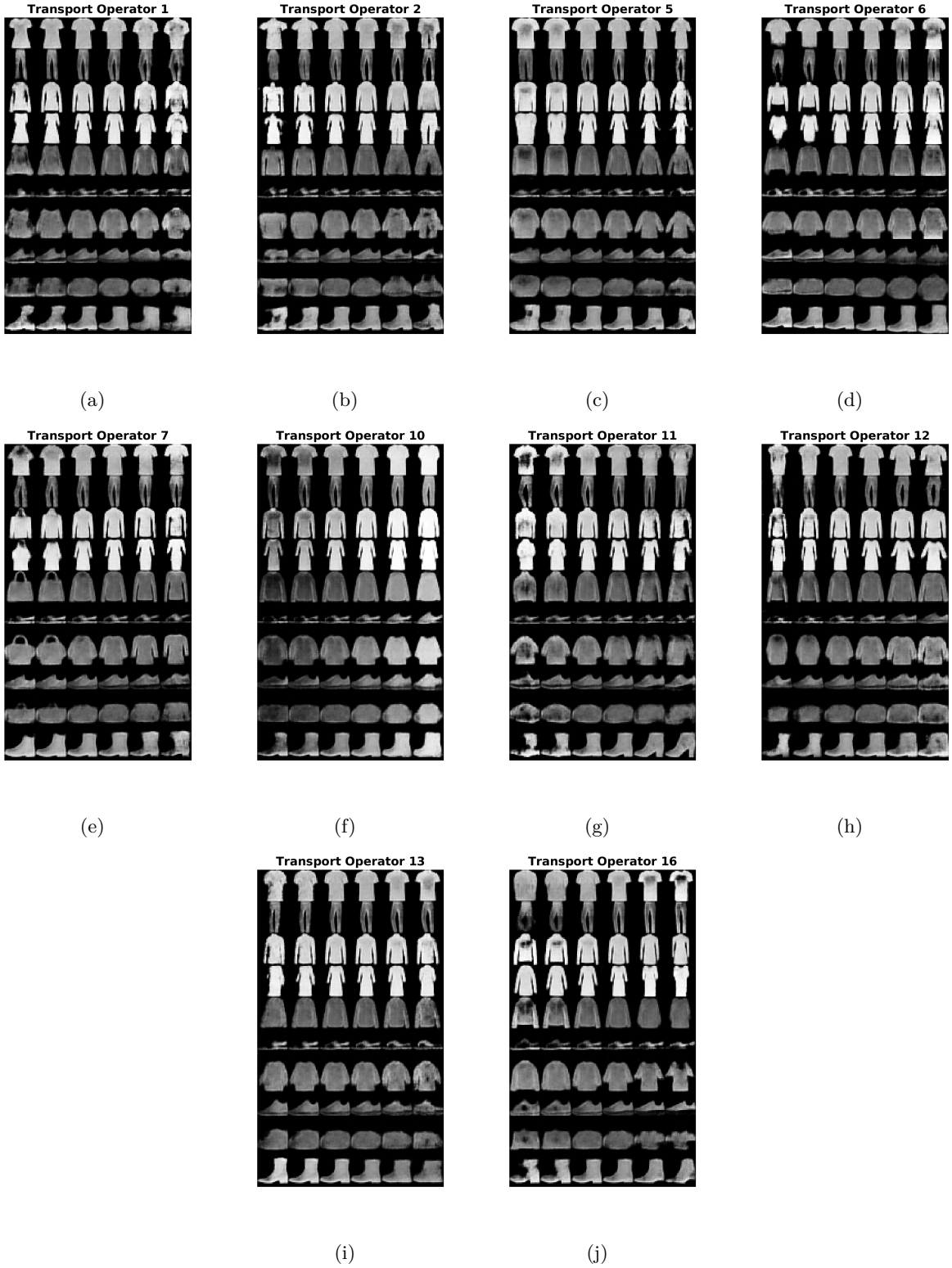
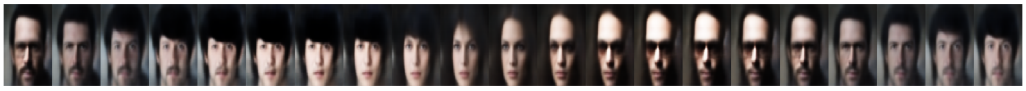


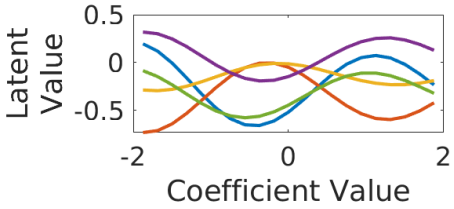
Figure 15: Paths generated by all non-zero transport operators trained on the Fashion-MNIST dataset. Images in the middle column of the image block are the reconstructed inputs and images to the right and left are images decoded from transformed latent vectors in positive and negative directions, respectively

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

Figure 16: Paths generated by all 40 transport operators trained on the celebA dataset. Images in the middle column of the image block are the reconstructed inputs and images to the right and left are images decoded from transformed latent vectors in positive and negative directions, respectively



(a)



(b)

Figure 17: An example of an operator that generates a nearly cyclic path in the latent space (a) Image outputs along a path generated by a single learned operator. The images on the far left and far right look similar which indicates that this operator generates a nearly closed path that begins and ends at the same point. (b) Paths of five of the 32 latent dimensions as the learned operator is applied to them. This again highlights the cyclic nature of the transport operator generated paths.