STRUCTURED LEARNING WITH MANIFOLD REPRESENTATIONS OF NATURAL DATA VARIATIONS

A Dissertation
Presented to
The Academic Faculty

By

Marissa Connor

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2021

STRUCTURED LEARNING WITH MANIFOLD REPRESENTATIONS OF NATURAL DATA VARIATIONS

Thesis committee:

Dr. Christopher Rozell
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mark Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Irfan Essa
School of Interactive Computing
Georgia Institute of Technology

Dr. Bruno Olshausen
Helen Wills Neuroscience Institute and School of Optometry
University of California, Berkeley

Dr. David Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: August 10, 2021

To my daughters, Juliette and Camille. You are my sunshine. I hope that this will inspire you to work hard in pursuit of your dreams and to be limitless in your ambitions.

ACKNOWLEDGMENTS

I have been incredibly lucky to have had a wonderful community of supporters while at Georgia Tech. First I would like to thank my advisor, Chris Rozell. You are an inspiration in so many ways - your perspective on life outside of research, your interdisciplinary and wide-ranging engagement with the broader community, and your commitment to true mentorship of students. You recognized the value of my hard work and dedication in the face of technical challenges and frustrations, and you were an incredible source of support for which I will always be grateful. I would like to thank Bruno Olshausen for giving generously of your time for both expansive and focused technical discussions. It was an honor to collaborate with you and to experience your thoughtfulness and natural curiosity. Thank you to Mark Davenport for your technical feedback and your support of the greater Georgia Tech ECE community. Finally, thank you to Professor David Anderson and Professor Irfan Essa for serving on my committee and providing me with technical guidance.

I want to thank Jack Culpepper for your technical mentorship and guidance throughout the years. It was a pleasure being able to discuss details of a model that we both can appreciate the benefits and challenges of. I was extremely fortunate to spend my time at Georgia Tech with the SIPsters: Aurele, Abbie, Mengchen, Adam Charles, Nick, John, Ayse, Greg, Adam Willats, Matt, Pavel, Stefano, Sankar, Kyle, Kion, Belén, Siva, Saurabh, and Alex. What a fun and engaging group! Acknowledgements would not be complete without a shout out to the MLCV subgroup. It was a great collaborative mini community for technical discussion. I want to give individual thanks to Adam Charles who is a excellent example of what a researcher can be - friendly, helpful, well-rounded, motivated, and bright. I am very grateful for the support and camaraderie of Greg, Matt, and Adam Willats. A Ph.D. can sometimes feel like an isolating environment in which you're working alone towards an unknown goal. The community of support, communication, and encouragement that you three provided was immensely helpful for pushing through roadblocks to achieve success. You would all drop anything to discuss a technical questions, read a draft, or create a new emoji.

I would like to thank several members of the larger Georgia Tech community. First, the Children of the Norm - it was a pleasure to form relationships with so many smart and inspiring people. A special thank you to Michael Moore - my experience at Georgia Tech would not have been the same without your (extremely) consistent presence. I thoroughly enjoyed our discussions (especially the bachelor recaps). To my fellow members of Lean In, my experience at Georgia Tech would truly not have been the same without you. I have so much pride when thinking about our fabulous community of women (and Sebastian!) supporting other women. Kaitlin Fair - you came into my life at Georgia Tech at a time when I was figuring out my voice and power as an independent researcher and you helped me establish that. You have always pushed me outside of my comfort zone to reach for my goals and I will always be appreciative of that.

A big thank you to my incredible family who shaped me into the woman I am today. Thank you to my mom for being such a constant source of comfort and encouragement. You have been a listening ear through endless phone calls and an unbelievable help with all that life has thrown our way. On top of all that, you are such an inspiration to me as a woman with a successful technical career who always seemed capable of doing it all. Thank you to my dad without whom I surely would not have been on this path. You taught me to reach for the stars. I aspire to achieve your zest for life and your unquenchable curiosity. And thank you Charlie for always being my bud.

Most importantly, a gigantic thank you to my husband Jeff. We have lived and grown so much throughout my Ph.D. and you have been my constant companion and my number one supporter. You have celebrated with me through the peaks and supported me through the many valleys of this process. This achievement feels like one that we can both joyously share.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Both humans and machines rely on object classification to perform vital tasks. Classification is challenging in part because natural variations can greatly alter objects' appearances while maintaining their identity. An effective classifier must be invariant to these identity-preserving transformations while also successfully discriminating between samples from different classes. State-of-the-art machine learning algorithms achieve impressive transformation-invariant classification results by making use of large labeled datasets. This results in black box models that lose information about the attributes that make samples unique. Insights about the structure of data can be used to develop models of how natural data varies which can help create more robust and interpretable classification techniques.

According to the manifold hypothesis, natural variations in high-dimensional data lie on or near a low-dimensional, nonlinear manifold. Additionally, many identity-preserving transformations are shared among classes of data which can allow for an efficient representation of data variations: a limited set of transformations can describe a majority of variations in many classes. Therefore, a machine learning model can improve its classification performance with limited data and its interpretability by directly incorporating manifold models of natural data variations that are shared among classes.

This work demonstrates the learning of generative models of identity-preserving transformations on data manifolds in order to analyze, generate, and exploit the natural variations in data for machine learning tasks. The introduced transformation representations are incorporated into several novel models to highlight the ability to generate realistic samples of semantically meaningful transformations, generalize transformations beyond their source domain, and estimate transformations between data samples. We first develop a model for learning 3D manifold-based transformations from 2D projected inputs. This work highlights how the manifold transformation structure can be learned with impoverished data and used to perform depth inference from 2D moving inputs. We then confirm that our generative model of transformations can be generalized across classes by defining two transfer learning tasks that map transformations learned from a rich dataset to previously unseen data. Next, we develop the manifold autoencoder, which learns low-dimensional manifold structure from complex data in the latent space of an autoencoder and adapts the latent space to accommodate this structure [1, 2]. Finally, we introduce the Variational Autoencoder with Learned Latent Structure (VAELLS) which incorporates a learnable manifold model into the fully probabilistic generative framework of a variational autoencoder [3].

CHAPTER 1
INTRODUCTION

Both humans and machines rely on object classification to perform vital tasks. They may need to recognize a pedestrian in an intersection, discern the identity of a person waving to them on the street, or determine whether an x-ray indicates that a patient has cancer. The task of classification, while often second nature to humans, is complex. Natural transformations, such as changes in lighting, pose, size, or viewpoint, can greatly alter an object's appearance while maintaining its identity. Therefore, an effective classifier must be invariant to identity-preserving transformations while also effectively discriminating between samples from different classes.

State-of-the-art machine learning algorithms such as convolutional neural networks [4, 5] and transformers [6] have achieved impressive classification results – reaching nearly 90% classification accuracy on the benchmark ImageNet dataset [7]. Improvements in machine learning performance can be attributed to the use of large labeled datasets like ImageNet [7] (21k classes, 14 million images), JFT-300M [8] (18k classes, 303 million images), and JFT-3B (30k classes, 3 billion images) [9]. These datasets contain many classes of objects and numerous examples of each class that comprehensively capture in-class variations. Therefore, the classifier's invariance to natural transformations is achieved through the sheer diversity of the training samples presented rather than through a deeper understanding of the natural data structure.

The need for large labeled datasets presents practical challenges for generalizing the high-performance classification models to real world settings. First, in many applications, such as medical, environmental, and defense imaging, it is not possible to collect large labeled training sets with numerous examples for every class. With limited class examples, neural network generation and classification performance decreases greatly [10, 11, 12].

Second, these network models gain invariance to natural transformations at the cost of losing information about the attributes that make each sample unique, preventing them from generalizing to novel images and classes. Finally, neural networks are inherently black boxes which extract relevant information from samples without providing evidence for how or why they arrive at a certain output. These challenges can be addressed by utilizing insights about the structure of data and how humans may exploit that structure to achieve robust classification performance.

According to the manifold hypothesis, natural variations in high-dimensional data lie on or near a low-dimensional, nonlinear manifold [13]. Additionally, the manifolds representing different classes are separated by low density regions [14]. 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 allows data variations to be represented efficiently: a limited set of transformations can describe a majority of variations in many classes.

Work in the neuroscience literature suggests that humans build internal models of natural transformations when processing data. In particular, experiments suggest that humans create an analog spatial representation in which the stages of internal mental transformation processes, such as rotation, correspond directly to the stages of transformation processes observed in the world [15, 16]. This implies that humans internally imagine transformations when asked to perform tasks like identifying rotated versions of reference objects [17, 18, 19]. Neuroscience research has also shown that the brain may exploit the manifold nature of object transformations by using hierarchical processing stages to untangle the representations of different objects undergoing the same physical transformations (e.g., changes in pose and position) [20, 21]. Finally, there is experimental evidence that humans generalize transformation representations across different objects such that a transformation identified for one object can be used to model the same transformation applied to another object [22]. These themes of directly incorporating manifold models of natural data variations into visual processing and classification can be leveraged to resolve the practical challenges identified in modern machine learning systems.

Specifically, if machine learning systems directly modeled natural data variations that are shared among classes, those variations could be learned on a small set of richly sampled classes and then new samples could be identified as transformed versions of samples that were already presented to the system. This would reduce the requirement for large numbers of labeled examples from each training class, instead allowing for classification using a global dictionary of known transformations and just a few training samples per class. Additionally, incorporating transformations into classification systems would maintain information about the attributes that make samples unique and would boost the explanatory power of these systems because training samples could be related to test samples through the known transformations. The cornerstone for such a system is a representation of natural transformations.

The goal of this thesis is to demonstrate the learning of generative models of identity-preserving transformations on data manifolds in order to analyze, generate, and exploit the natural variations in data for machine learning tasks. In order to accommodate the human-inspired classification setting presented, this transformation model has several requirements. In particular the transformations should be

- Learned from the data with limited supervision. This will provide the flexibility of identifying dataset-specific transformations that may not be known a priori.
- Constrained to the structure of an estimated data manifold. This ensures that the transformations represent natural data variations.
- Transferable between classes. This will allow transformations to be learned on one or a few classes and used on other classes.
- Capable of generating new samples. This enables the system to imagine transformed

samples.

- Able to infer relationships between samples. This will allow for the comparison between novel samples and previous training samples to determine if they are related by known transformations.

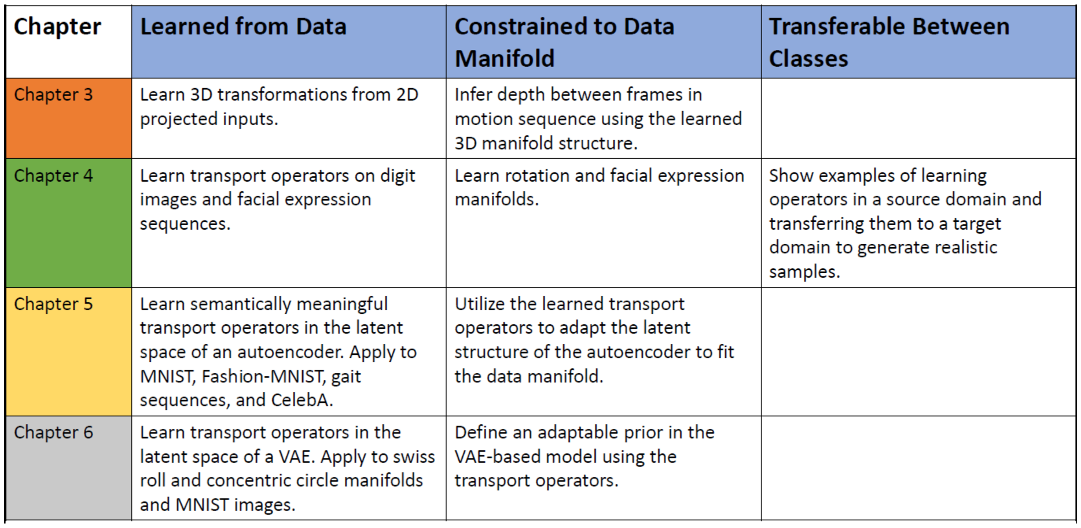
In this thesis, transformation learning is incorporated into several novel models, each of which highlights how the transformation representation satisfies a subset of the requirements listed above and together they show the model satisfies all of the requirements. Experiments are performed with a variety of data types (3D points, engineered features, and features learned by neural networks) using datasets with and without explicitly labeled natural variations. In these experiments, it is shown that the developed transformation model can be used to generate realistic samples of semantically meaningful transformations, generalize transformations beyond their source domains, and estimate transformations between data samples. This model establishes a foundation from which to build a human-inspired classification model that directly incorporates natural transformations. This novel classification model could be of interest in fields like medicine, defense, and robotics which utilize label-limited datasets with the goals of understanding natural identity-preserving variations and performing classification that is invariant to those variations. Additionally it may be used to develop insight into how humans learn and adapt internal models of transformations that contribute to their robust understanding of the world.

In the model presented in this thesis, identity-preserving transformations are learned using the transport operator manifold learning strategy [23]. Manifold learning models estimate the low-dimensional manifold structure of data and examine how points are related through natural transformations. The transport operator technique is a specific manifold learning model that learns to transform points through nonlinear Lie group operators that transverse a manifold [24, 25, 23, 26, 27, 11, 1, 3]. Lie group operators represent infinitesimal transformations which can be applied to data through an exponential mapping to transform points along a manifold. This manifold can be globally defined by a set of operators that each move in different directions along it [28, 29]. While previous work with the transport operator manifold model presented simple, proof-of-concept examples, it is well-suited for representing complex natural data variations because the operators can be learned from the data, applied to data points to generate new samples beyond their local neighborhoods, and used to estimate geodesic paths. In this thesis, through a series of engineering advancements, this model is extended to increase its capability to learn complex transformations. It is then incorporated into a diverse set of applications to highlight the power and flexibility of this model for representing transformations in different training environments.

1.1 Organization of Work

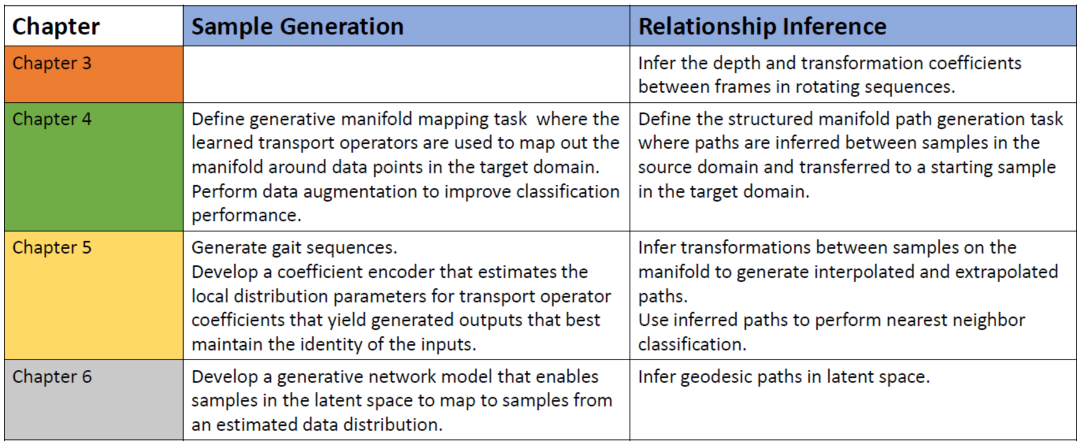
Chapter 2 provides background on techniques and context required for understanding the work in later chapters. Specifically this chapter discusses manifold models for learning natural variations and current state-of-the-art methods for incorporating transformations directly into machine learning models.

Figure 1.1 provides a high level view of the chapter organization and how the requirements for a desired transformation model are addressed by the chapters. In Chapter 3, a model is developed for learning transport operators that represent 3D rotational transformations from 2D projected inputs. This work highlights how the transport operator transformation model can be learned with impoverished data and how the constraint of the learned manifold model helps enable depth inference from two-dimensional inputs. While rotation is a well understood transformation, there is no computational model for how humans develop and adapt their internal model of rotation, which may be needed to perform mental spatial transformations, depth inference, and third person perspective taking. This work provides a possible explanation for how humans may incorporate an internal model of 3D transformations to aid in vision tasks with 2D projected inputs on their retina. The model is also used to draw conclusions about the possible sensory input requirements for



Chapter	Learned from Data	Constrained to Data Manifold	Transferable Between Classes
Chapter 3	Learn 3D transformations from 2D projected inputs.	Infer depth between frames in motion sequence using the learned 3D manifold structure.	
Chapter 4	Learn transport operators on digit images and facial expression sequences.	Learn rotation and facial expression manifolds.	Show examples of learning operators in a source domain and transferring them to a target domain to generate realistic samples.
Chapter 5	Learn semantically meaningful transport operators in the latent space of an autoencoder. Apply to MNIST, Fashion-MNIST, gait sequences, and CelebA.	Utilize the learned transport operators to adapt the latent structure of the autoencoder to fit the data manifold.	
Chapter 6	Learn transport operators in the latent space of a VAE. Apply to swiss roll and concentric circle manifolds and MNIST images.	Define an adaptable prior in the VAE-based model using the transport operators.	

(a)



Chapter	Sample Generation	Relationship Inference
Chapter 3		Infer the depth and transformation coefficients between frames in rotating sequences.
Chapter 4	Define generative manifold mapping task where the learned transport operators are used to map out the manifold around data points in the target domain. Perform data augmentation to improve classification performance.	Define the structured manifold path generation task where paths are inferred between samples in the source domain and transferred to a starting sample in the target domain.
Chapter 5	Generate gait sequences. Develop a coefficient encoder that estimates the local distribution parameters for transport operator coefficients that yield generated outputs that best maintain the identity of the inputs.	Infer transformations between samples on the manifold to generate interpolated and extrapolated paths. Use inferred paths to perform nearest neighbor classification.
Chapter 6	Develop a generative network model that enables samples in the latent space to map to samples from an estimated data distribution.	Infer geodesic paths in latent space.

(b)

Figure 1.1: High level view of how the chapters each address the stated requirements for the desired transformation model.

learning rotation and to perform random dot kinematogram tasks where 3D structure is inferred from motion.

In Chapter 4, the transport operator manifold model is established as a robust and accurate manifold representation which can be generalized to define the manifold structure outside of the original training domain. This confirms that the transformations learned using transport operators can be transferred between classes and used in tasks with limited training samples and/or training labels. The capability to generate new samples is confirmed through the development of two transfer learning tasks that map the transformations learned from a rich dataset to previously unseen data. These transfer tasks are shown to increase classifier robustness, reduce required diversity in the training data, and enable the generation of novel examples.

In Chapter 5, the transport operator model is incorporated into the latent space of an autoencoder to develop the manifold autoencoder which learns the low-dimensional manifold structure from training data and adapts the latent space to accommodate this structure. By integrating the transport operator model into an autoencoder latent space, this model can learn complex data transformations which can be visualized by decoding the transformed latent vector outputs. This learned manifold model is used to identify unlabeled transformations in the data, perform data augmentation, and enhance classification performance. It is first applied to data that has closed transformation paths which extend from a starting point and return to nearly the same point. Through experiments on data with natural closed transformation paths, it is shown that this model introduces the ability to learn the latent dynamics of complex systems, generate transformation paths, and classify samples that belong on the same transformation path. The model is then extended to learn representations of natural variations without requiring transformation labels during training. Finally, a method is developed for learning local regions where each transport operator is likely to be used while preserving the identity of inputs. This improves data augmentation performance of the model and introduces the capability of analyzing the transformational invariances of pretrained classifiers.

In Chapter 6, the Variational Autoencoder with Learned Latent Structure (VAELLS) is introduced which incorporates a learnable manifold model into the latent space of a Variational Autoencoder (VAE) [30, 31] . In contrast to the manifold autoencoder, which uses a deterministic autoencoder mapping, the VAELLS model has a fully probabilistic generative framework. This enables the model to learn the nonlinear manifold structure from the data and use that structure to define a prior distribution in the latent space which can be sampled from to generate transformed outputs. The integration of a latent manifold model ensures that the learned prior is well-matched to the data, leading to a more accurate generative model of the data than other VAE approaches with fixed priors in the latent space. The latent manifold model also permits defining generative transformation paths in the latent space and describing class manifolds with transformations stemming from examples of each class.

Finally, in Chapter 7 we will summarize the contributions of this thesis and discuss future directions for this work.

CHAPTER 2
BACKGROUND

2.1 Manifold Models

Envision the following observations that you might have in your everyday life – rotating your head while viewing your shifting environment, imagining how your friend would look if their smile transformed into a frown, or watching a basketball bouncing on a basketball court. Each of these examples represent a complex, smoothly-varying transformation that humans are accustomed to understanding and interpreting. Natural laws constrain transformations in the world to be smooth - as an object is transformed from one state to another, it follows a continuous path with each intermediate step also producing a transformed version of that object. These smoothly varying natural transformations support the formation of data manifolds containing transformed samples from individual classes. A manifold is a topological space that locally resembles a Euclidean space with a dimension n [32], where n is less than the dimension of the input space. An intuitive example of a manifold is a sphere in three dimensions. From every point on a sphere the local neighborhood appears two-dimensional but the overall nonlinear structure is three-dimensional.

The manifold hypothesis describes high-dimensional data as lying on or near a low-dimensional, nonlinear manifold [13]. The existence of a nonlinear manifold structure has been experimentally verified in many domains including natural images [33], neural recordings [34, 35], hyperspectral imagery [36], and medical data [37]. The implications of the manifold hypothesis are particularly important for machine learning tasks that utilize high dimensional data because, rather than attempting to extract useful information from individual pixels or input features, the data can be distilled to a lower dimensional space in which the relevant characteristics can be analyzed and utilized.

Dimensionality reduction techniques have long been employed to capture the low-dimensional structure in data. When the data variations are within a linear subspace, classic dimensionaity reduction techniques such as Principle Component Analysis (PCA) [38, 39] and Independent Component Analysis (ICA) [40] can be employed. PCA decomposes the data into a linear combination of orthogonal basis vectors that maximize the variance of the data projection and ICA extends this to non-Gaussian signals by exploiting the statistical independence between the basis vectors rather than orthogonality. When the manifold structure is nonlinear, as in many applications, manifold learning algorithms can be used to discover this structure from the data.

The most common manifold learning approaches (e.g., Isomap [41], Locally-Linear Embedding (LLE) [42], Maximum Variance Unfolding (MVU) [43], Laplacian Eigenmaps [44], t-Stochastic Neighbor Embedding (t-SNE) [45], and Uniform Manifold Approximation and Projection (UMAP) [46]) represent the manifold through a low-dimensional embedding of the data points without a method for generating transformed points on the manifold. There have been several techniques introduced (e.g., Locally Smooth Manifold Learning (LSML) [47, 48] and Non-Local Manifold Tangent Learning (NLMT) [49]) that use the training points to learn a function that maps high dimensional points to linear tangent planes that represent the manifold structure. The Locally Linear Latent Variable Model (LL-LVM) estimates both the embedded points and the mapping between the high-dimensional and low-dimensional data using a probabilistic model which attempts to maximize the likelihood of the observations [50]. These methods, which we refer to as tangent plane estimation techniques, can admit representations that have some generative capabilities on the estimated linear tangent planes. However, the linear approximations limit the magnitude of transformations that can be applied while remaining on the manifold.

Other methods define manifolds through nonlinear Lie group operators that transverse the manifold [24, 25, 23, 26, 27, 11, 1, 3]. To characterize manifold transformations, a transformation operator, $\mathbf{T} \in \mathbb{R}^{D\times D}$, needs to be defined that can generate the transformations between two data points $\mathbf{x_0}, \mathbf{x_1} \in \mathbb{R}^D$: $\mathbf{x}_1 = \mathbf{T}\mathbf{x}_0$. This transformation belongs to a family of operators $\mathcal{T}$ (i.e., $\mathbf{T} \in \mathcal{T}$) which is endowed with group structure. Specifically, i) there exists a group operation $\circ$ to combine transformations; ii) if $\mathbf{T}_0, \mathbf{T}_1 \in \mathcal{T}$ then $\mathbf{T}_0 \circ \mathbf{T}_1 \in \mathcal{T}$; iii) the operators are associative; iv) there exists an identity operator; and v) there exists an inverse element for each operator $\mathbf{T} \in \mathcal{T}$. This continuous transformation group is termed a Lie group [32].

A Lie group of transformation operators defines transformations that describe the natural variations in the data space $\mathcal{X}$. Thus these operators can be used in conjunction with an input data point $\mathbf{x}_0$ to compactly represent the manifold surface, $\mathcal{M} \subset \mathcal{X}$ stemming from $\mathbf{x}_0$: $\mathcal{M} = \{\mathbf{X} \in \mathcal{X} | \mathbf{x} = \mathbf{T}\mathbf{x}_0, \mathbf{T} \in \mathcal{T}\}$.

Each of these transformation operators represents a continuous transformation and can be formulated as the infinitesimal transformation: $\mathbf{T} = (\mathbf{I} + \boldsymbol{\Psi}\Delta c)$ where $\mathbf{T}$ is a small offset from the identity matrix. The real number $c \in \mathbb{R}$ parameterizes $\mathbf{T}$ with a matrix $\boldsymbol{\Psi} \in \mathbb{R}^{D \times D}$ which is an operator for the transformation group. Transformed data points are defined with the infinitesimal transformation as $\mathbf{x}_{\Delta c} = \mathbf{T}\mathbf{x}_0 = (\mathbf{I} + \boldsymbol{\Psi}\Delta c)\mathbf{x}_0$. As $\Delta c \to 0$, the previous equation becomes $\frac{\delta \mathbf{x}}{\delta c} = \boldsymbol{\Psi}\mathbf{x}$. This dynamical system has the well-known solution $\mathbf{x}_c = \mathrm{expm}(\boldsymbol{\Psi} c)\mathbf{x}_0$ where expm is a matrix exponential [24, 25].

It is assumed that the manifold is defined by a finite set of transformation operators $\{\mathbf{T}_1, \mathbf{T}_2, ...\mathbf{T}_M\}$, each parameterized by its own coefficient, c_m, which specifies the contribution of the individual operator to the given transformation. Therefore a transformation $\mathbf{T}$ can be represented by a weighted combination of transformation dictionary elements, $\boldsymbol{\Psi}_m$

$$\mathbf{T} = \mathrm{expm}\left(\sum_{m=1}^{M} \boldsymbol{\Psi}_m c_m\right). \tag{2.1}$$

This defines a generative transformation model that characterizes the manifold $\mathcal{M}$ by a set of transformation operators $\boldsymbol{\Psi}_m$. Following previous conventions [23], these transformation dictionary elements $\boldsymbol{\Psi}_m$ are called *transport operators*. The relationship between points on the manifold is defined with the following generative model:

$$\mathbf{x}_1 = \operatorname{expm}\left(\sum_{m=1}^{M} \boldsymbol{\Psi}_m c_m\right) \mathbf{x}_0 + \mathbf{n}, \qquad \mathbf{n} \sim \mathcal{N}(0, I) \quad c_m \sim \text{Laplace}\left(0, b\right). \tag{2.2}$$

This formulation allows for different geometrical characteristics at various points on the manifold and it provides the flexibility of defining a different transformation matrix between every pair of points. The specific local structure of the transformations between a pair of points is governed by the coefficients $\mathbf{c} \in \mathbb{R}^M$. The Laplace prior on the coefficients encourages the manifold geometry between any given pair of points to be represented by only a small subset of dictionary elements. Throughout this document we define $\mathbf{T}(\mathbf{c}) = \operatorname{expm}\left(\sum_{m=1}^{M} \boldsymbol{\Psi}_m c_m\right)$ and refer to the dynamics matrix $\mathbf{A} = \sum_{m=1}^{M} \boldsymbol{\Psi}_m c_m$.

This is an ideal manifold model to incorporate into machine learning techniques to represent natural variations for a few reasons. First, it defines the manifold through a set of learned operators that can be used at any point on the manifold. As we discuss in more detail in Chapter 4, this manifold representation can also generalize beyond the original training domain better than the tangent plane estimation techniques. Second, the operators themselves represent a global set of nonlinear transformations that describe natural variations in the data. These operators can be used to interpret dataset characteristics, infer relationships between points, and generate long paths along the manifold.

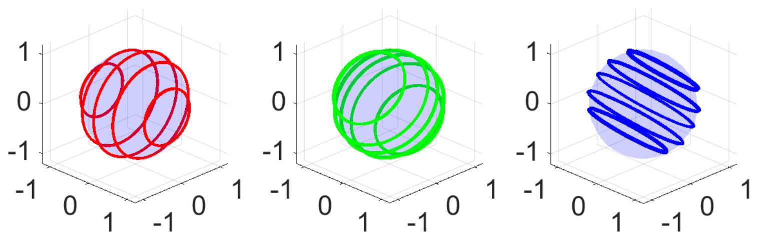
Using the relationship between points in Equation 2.2, a probabilistic generative model can be written that allows efficient inference. This model assumes a Gaussian noise model and a Gaussian prior on the transport operators $\boldsymbol{\Psi}$ as well as the sparsity inducing prior on the coefficients $\mathbf{c}$. The resulting negative log posterior for the model is given by

$$E_\Psi = \frac{1}{2}\left\| \mathbf{x}_1 - \operatorname{expm}\left(\sum_{m=1}^{M} \boldsymbol{\Psi}_m c_m\right) \mathbf{x}_0 \right\|_2^2 + \frac{\gamma}{2}\sum_m \|\boldsymbol{\Psi}_m\|_F^2 + \zeta\|\mathbf{c}\|_1, \tag{2.3}$$

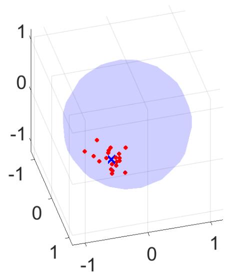
where $\|\cdot\|_F$ is the Frobenius norm and $\gamma, \zeta \geq 0$. The Frobenius norm regularizer on the dictionaries serves two purposes. First, it balances the effect of the coefficient sparsity regularizer. The ℓ_1 norm term on the coefficients encourages coefficient sparsity but also encourages small coefficient magnitudes. One way to achieve small coefficient magnitudes while continuing to effectively estimate the path between $\mathbf{x}_0$ and $\mathbf{x}_1$ is to increase the magnitudes of the operators. The Frobenius norm regularizer constrains the growth in magnitude of the operators. 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 $\mathbf{x}_0$ and $\mathbf{x}_1$ then the dictionary regularizer reduces its magnitude to zero. Therefore, during training, the model order can be estimated based on how many dictionary elements remain non-zero.

Using an established unsupervised algorithm from Culpepper and Olshausen [23], the transport operator dictionary can be learned from pairs of neighboring points by performing alternating minimization with the objective in Equation 2.3. Specifically, for a given batch of point pairs, the algorithm alternates between inferring the coefficients given a fixed dictionary of transport operators and taking a step in the gradient direction of the dictionary elements. After training the transport operators, the manifold structure can be represented by a combination of these learned transport operators and samples that are known to exist on the manifold. Figure 2.1a shows an example of the dictionary of transport operators learned on point pairs on a sphere. Each plot uses a few example starting points on the sphere (i.e., on the manifold) $\mathbf{x}_i(0)$, $i = 1, ...N$ and visualizes the effect of a single transport operator by applying a single dictionary element $\mathbf{\Psi}_m$ to each point in time steps $\mathbf{x}_i(t) = \exp(\mathbf{\Psi}_m t)\mathbf{x}_i(0)$, $t = 0, ..., T$.

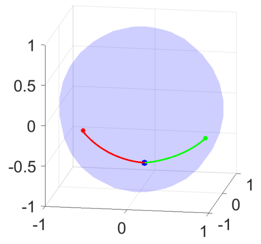
These operators can be used to generate new samples and infer relationships between samples – these are two of the requirements set forth in the introduction for a human-inspired classification system. Generation is accomplished by applying the learned operators with a set of coefficients that are randomly sampled or representative of a particular



(a)



(b)



(c)

Figure 2.1: Examples of transport operator learning, sampling, and inference on 3D spherical manifold. (a) Orbits generated from transport operators trained on a spherical manifold. (b) Points generated from sampling transport operator coefficients. The blue 'x' represents the original point on the manifold and the red dots represent the generated points. (c) Examples of inferred paths between points on the manifold. The blue dot represents the starting point and the red and green dots represent the ending points. The lines show the interpolation paths.

transformation.

$$\mathbf{x}_i = \text{expm}(\sum_m \mathbf{\Psi}_m w_{i,m})\mathbf{x}_0, \tag{2.4}$$

$$w_{i,m} \sim \text{Laplace}(0, a_m). \tag{2.5}$$

Figure 2.1b shows samples generated with the rotational transport operators shown in Figure 2.1a.

Inference of transformations between samples is a critical step in learning the transformations and can be used for analyzing relationships between points and interpolating manifold paths. After inferring a set of transport operator coefficients $\mathbf{c}^*$ between points on the manifold, a path can be generated: $\mathbf{z}_t = \operatorname{expm}\left(\sum_{m=1}^{M} \Psi_m c_m^* t\right) \mathbf{z}_0$. When the path multiplier t is between 0 and 1 that indicates interpolation and path multipliers beyond 1 indicate extrapolation. Figure 2.1c shows interpolated paths between samples on the sphere using inferred coefficients.

2.2 Incorporating Transformations into Neural Networks

In the introduction, we discussed the benefits of a human-inspired classifier that directly incorporates natural transformations into its processing chain and put forth requirements for a transformation model that could form the basis for this classifier. Namely the transformation model should learn the natural variations from the data, constrain the transformation structure to the data manifold, enable transfer of the transformations between classes, have the capability of generating new samples, and have the ability to infer relationships between samples. We review state-of-the-at methods for incorporating transformations into machine learning models. First we review models that incorporate transformations into the classification pipeline to increase classifier robustness. Next we examine models that incorporate estimates of transformations and manifold structure into generative neural networks. These models are of interest because they are generate realistic image samples to estimate transformation effects and augment training data.

As we will discuss below, the state-of-the-art techniques do not meet the requirements we set forth for a transformation model. In particular, most of these models require the use of predefined transformations. While effective in general cases for incorporating widely known transformations like rotation and scaling, there are many identity-preserving transformations in complex datasets that are difficult to identify.

2.2.1 Transformations in Classification Pipeline

As mentioned in the introduction, many state-of-the-art classification methods that achieve the high classification accuracy do so by incorporating training data that includes examples illustrating a wide variety of transformations. Without a dataset that includes this abundance of transformation variability, the resulting accuracy of trained classifiers will suffer. To avoid this performance degradation and increase the robustness of classifier performance, there are several approaches that incorporate transformations directly into the classification pipeline. Some approaches augment input data with transformations and other incorporate invariance or equivariance to transformations into classifier networks.

The first approach incorporating transformations into the classification pipeline is data augmentation. Data augmentation applies transformations to training data samples, either in the input space or in a latent space, in order to present a classifier with a more diverse set of samples. The core component of a successful data augmentation technique is a method for selecting which transformations to apply in order generate new, informative samples without contaminating the training data with samples that are not reflective of natural data. The simplest data augmentation strategy manually incorporates transformations like flipping, cropping, and changing colors into the training set [51]. This has been proven effective for enhancing image classification performance [52]. There are some techniques that incorporate these known, identity-preserving transformations in an intelligent manner by modeling human augmentation approaches [53] or identifying an augmentation policy that leads to the best results on subsets of the training and validation data [54, 55, 56]. If there are accurate models for generating data in the training domain, new examples can be synthetically generated to incorporate variability that may be lacking in real-world data [57]. All of these methods require prior knowledge of the image processing functions that represent identity-preserving transformations.

Rather that defining the image processing functions a priori, representations of data variability can also be identified through examining relationships between pairs of points in a dataset. In some cases, pixel-wise image deformations between examples from the same class can be gathered from a training set and applied to new classes. The transformations between pairs of images in the same class have been represented as simple affine transformations [58] as well as manifold transformations of image velocity fields [11]. There are several methods that use a strategy called mixup to generate new samples that are combinations of two existing images in the dataset [59]. While these methods estimate transformations from the datasets that can be used across classes and have the capability of generating samples, they define data variations through linear paths which may not reflect the data manifold structure and they do not have methods for estimating the transformational relationship between samples.

Some data augmentation techniques are applied to few-shot learning problems in which knowledge of variations in a rich source domain can be used to enable robust classification of examples in new target classes from few training samples [60]. These techniques are able to learn generalizable transformation representations and to leverage the shared transformations between classes to enrich sparsely sampled target classes. Some methods use image analogies to learn a function that can generate a transformed version of an input sample [61, 62]. The Δ-autoencoder model conditions both the encoder and decoder of an autoencoder on additional examples from the same class as the training inputs [63]. By limiting the latent dimension, this model results in latent vectors that represent the variations between the two examples from the same class. The learned decoder can be used to augment inputs by decoding sampled latent vectors.

The next set of approaches attempts to incorporate transformation invariance into convolutional neural networks (CNNs). The spatial transformer network (STN) accomplishes this by defining a network layer that estimates warping to apply to input features to increase classification accuracy [64, 65]. The initial STN model defines specific transformations like translation, scale, and rotation but it has been extended to incorporate more complex transformations like 3D perspective transformations [66, 67] and diffeomorphic image transformations [68] as well as combinations of transformations [69]. In a similar vein, deformable convolutional networks vary the grid of sampling locations for convolutions to mimic data transformations from one layer to the next [70]. By enforcing invariance to natural, identity-preserving transformations, these methods improve classification performance while losing information about what makes data samples unique and why a classifier arrives at a specific output.

Finally, there is a set of work focused on building equivariance to specific transformations into convolutional networks. To achieve equivariance, a change in the input should lead to a change in the output which means that information about transformations is preserved. In baseline CNNs, the convolutional filters applied on a sampling grid across images and feature inputs achieve translation equivariance because translations in the inputs result in translations in the convolved outputs. More recently there have been many models that incorporate rotational equivariance into convolutional networks by designing specific convolutional layers [71, 72, 73, 74]. These models have also been expanded to allow for equivariance to other transformations such as 3D rotation [75, 76] and scale [77]. As with the data augmentation approaches, most current equivariant approaches are limited to a simple set of prespecified transformations which constrains their effectiveness on datasets with more complex shared transformations.

2.2.2 Transformation Structure in Generative Networks

There are several network models which learn functions that map points from a low-dimensional latent space to a high-dimensional data space including autoencoders [78, 79], variational autoencoders (VAEs) [30, 31] and generative adversarial networks (GANs) [80]. Autoencoders are deterministic networks which are made up of encoders that map from the data space to the latent space and decoders that map latent vectors back to the date space. Autoencoders are trained in an unsupervised manner using a reconstruction loss between image inputs and reconstructed image outputs. Generative models (VAEs and GANs) represent complex data distributions through the learned generator functions that map latent vectors to data outputs. In particular, GANs and VAEs sample from a latent space with a specified prior distribution in order to generate new, realistic samples. VAEs are trained with an objective that maximizes a lower bound on the log likelihood of the data. This objective contains two terms - the first term contains a reconstruction error between the input image and the reconstructed output and the second term encourages the variational posterior of the latent vectors to resemble a prespecified prior distribution.

Autoencoders and generative network models provide the ability to imagine new data samples and to visualize the transformations resulting from motion in the low-dimensional latent space. In many instances, the latent space is described as a manifold representation. However, there is often no objective during training that ensures that the structure in the latent space corresponds to the true data manifold.

VAE-based models can incorporate structure in the latent space through the use of specific prior distributions on latent vectors. In the traditional VAE, the latent space prior is defined as a Gaussian distribution for model simplicity [30]. Other models incorporate more complex latent structures such as hyperspheres [81], tori [82, 83], and hyperboloids [84, 85, 86]. Recently the Lie VAE model was developed which encodes the data into latent variables that are elements in a Lie group which represent transformations of a reference object [83]. These models have demonstrated their suitability for certain datasets by choosing a prior that is the best match out of a predefined set of candidates. A predefined latent structure may not be a good fit for the many datasets which have a complex data distribution. A mismatch between the specified latent space prior and the true data manifold will result in a less accurate model for the data.

There are several methods that incorporate a manifold model directly into the latent space of a neural network in order to learn the manifold structure of the data in a principled way. The contractive autoencoder (CAE) estimates manifold tangents by minimizing the 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 [87, 88, 14, 89]. Similar to the tangent plane estimation manifold learning techniques mentioned in section 2.1, the CAE has a limited ability to represent long paths on the manifold because the tangent plane estimate is only applicable in small neighborhoods around points. Several methods estimate interpolated geodesic paths in the latent space of a trained variational autoencoder (VAE) model [90, 91, 92, 93, 94] 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 [95, 96].

CHAPTER 3
LEARNING 3D MANIFOLD STRUCTURE FROM 2D PROJECTIONS

3.1 Introduction

Humans interact with a 3D world[1] but only perceive visual inputs as 2D projections on their retinas. To effectively interact with 3D objects and environments, humans must have the ability to estimate 3D structure from their limited projected inputs. On its own, the problem of recovering 3D structure is underconstrained – there are an infinite number of possible depths for each projected point. To resolve this challenge, humans rely on a variety of cues to infer depth including motion parallax, binocular disparity, texture, occlusions, shadows, size, blur, and shading [97]. One of the cues that strongly elicits the perception of 3D structure is the motion cue.

As an object transforms in 3D, the temporal consistency of this motion provides a continuous precept that can be used to resolve the ambiguity of depth for points on an object's surface. Motion has been shown to enable depth inference in the absence of other cues [98, 99, 100, 101, 102, 103]. Knowledge of the transformations that objects could be undergoing can constrain the possible 3D structure of point stimuli on those objects, leading to accurate depth perception.

Mathematically we have precise definitions of geometric transformations, like rotations and translations, and we know how to employ them to successfully compute point depths from multiple viewpoints or frames. Many computer vision algorithms utilize mutliview geometry to develop robust models for estimating depths of image keypoints [104, 105, 106, 107, 108, 109]. Additionally, the knowledge of the geometric relationship between

[1]This chapter is in collaboration with Dr. Bruno Olshausen and Dr. Christopher Rozell. MC developed the model and performed the experiments. BO provided valuable technical discussions and feedback and CR supervised the project. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1148903 and DGE-1650044

scenes can be used to develop machine learning algorithms that can estimate depths from single frames and multiple frames [110, 111, 112, 113]. While all of these methods achieve the objective of estimating depth from multiple viewpoints, we aim to understand how humans perceive 3D transformations and how they use their knowledge of transformations to aid in discerning structure from motion.

It is arguable that humans utilize internal models of 3D transformations based on a variety of mental transformation experiments. Experiments suggest that humans create an analog spatial representation in which the stages of internal mental transformation processes, such as rotation, correspond directly to the stages of transformation processes observed in the world [15, 16]. This, along with descriptions from subjects performing mental transformation tasks [114], suggests that humans internally imagine transformations when asked to perform tasks like identifying rotated versions of reference objects [17, 18, 19].

A question remains of how these internal models are formed. The knowledge to form models of transformations must be accumulated either on a developmental [115] or evolutionary time scale [116, 117]. In either instance there must be a mechanism for learning the transformation model from data and adapting the model to altered environments and vision systems. There is evidence that humans are able to learn about data variations in an unsupervised manner [118] and adapt their representations of data invariances based on experiences in an altered visual world [119, 120].

Generative manifold models present a possible mechanism for learning and representing internal models of natural transformations. The manifold hypothesis states that natural variations in high-dimensional data, like retinal projections of the 3D world, lie on or near a low-dimensional, nonlinear manifold [13]. Neuroscience research has shown that the brain explicitly exploits the manifold nature of object transformations by using hierarchical processing stages to untangle the representations of different objects undergoing the same physical transformations (e.g., changes in pose and position) [20, 21]. Additionally, we and others have shown that generative manifold models are able to learn representations of natural data variations [23, 26, 1, 3, 2].

In this work, we develop a proof of concept for the viability of learning 3D transformation representations from 2D projected inputs using a generative manifold model. Focusing on rotational motion, which is used in many structure from motion tasks [98, 101, 103], we develop a manifold-based method for inferring depth from moving 2D projected points and learning 3D rotational transformation models from 2D training stimuli. Finally, we apply the learned transformation model to structure from motion tasks and compare to human performance on psychophysical experiments.

3.2 Background

3.2.1 Structure from Motion

The ability for humans to perceive depth from moving points and objects has long been studied. Wallach and O'Connell first introduced the term kinetic depth effect to describe this phenomenon [121]. Since then, perception of 3D depth from motion cues has been investigated through a wide array of physchophysical experiments which aim to isolate the effect of the motion cue for depth perception [98, 99, 100, 101, 102, 103]. Stimuli for these experiments include 2D shadow visualizations of 3D objects and random dots on the surface of invisible rotating shapes. The experimental results suggest that humans can generate stable precepts of 3D structures in a variety of conditions.

Several computational models have aimed to describe this phenomenon. Ullman proposed a model called the incremental rigidity scheme which builds up a model of the 3D object structure as it receives new inputs [122]. Many other approaches utilize multiview geometry to form a 3D structure from multiple views of an object or scene. One of the early works in this space utilized the understanding that different frames of the same object are related through rotation and a translation operations to build a computational model for this process [104]. Many have built off of this initial model to estimate the relative relationships between views of objects using geometric transformations between points [123, 124, 125, 126].

Since these early papers, structure from motion has become a rich area of research in computer vision. Factorization methods estimate the camera orientation and 3D point locations over continuous frames of a rotating object [106, 127]. Other techniques expand on the geometric approaches by introducing robust methods for computing point correspondences between multiple views in order to estimate the structure of the full scene [105, 107, 108, 109, 128].

Finally, with the introduction of neural networks, there has been growth in neural-network-based methods for estimating the depth of objects and scenes. These models use 3D depth labels [129, 130, 131], knowledge of camera viewpoints [110, 111, 112], and temporal consistency [113] to supervise learning. Several methods use an image reconstruction objective by estimating depth in one image, transforming it, and projecting it to compare against a second image [110, 111, 112].

All of these methods, while very successful at estimating camera motion and depth, either rely on a mathematical understanding of rotational and translational motion and how it is applied or require ground truth depth labels. In this work we learn a representation of the 3D transformations without using ground truth knowledge of depth.

3.2.2 Mental Transformations

The evidence that humans create internal representations of 3D transformations comes from experiments on mental transformation tasks. There are two common mental transformation tasks. The first is the third-person-perspective (3PP) taking task where subjects must determine what a scene looks like from different perspectives. Tests of 3PP include the object perspective test [132], the Money standardized test of direction sense [133, 134], and the pictures test [135] which all ask users to imagine themselves at a certain location with a specific viewpoint in order to determine the placement of objects in an imaginary scene from the specified viewpoint. These tasks are examples of imagining egocentric motion (or motion of the observer). The second grouping of tasks are mental rotation tasks where subjects must determine if input objects are rotated versions of comparison objects [17, 136]. Mental rotation tasks include 2D planar rotation tasks with images of polygons [137], flags [138], pictures of human bodies [134], and alphanumeric characters [136] as well as 3D rotation tasks with wire frame objects [17, 139] and visualizations of human bodies [140]. These tasks are examples of allocentric motion (or motion of an object in the environment). In both of these cases, many subjects describe the process of completing the tasks as visualizing the full transformation in their head [114]. That is, they develop an analog spatial representation in which the task performed in their head corresponds to the imagined transformation happening in the 3D world.

There is quantitative evidence that supports the model that humans perform 3D mental rotation to compare objects. Early work by Shepard & Metzler asked subjects to determine if two wire frame images were rotated versions of one another [17]. The key finding was that the subject response time increased monotonically with the rotation angle which supports the model that humans perform internal mental rotation on this task. Since then, this linear relationship between response time and rotation angle on mental rotation tasks has been proven with a variety of visual inputs [18, 141, 142, 15]. Another set of research that supports the mental transformation hypothesis notes the impact of external motion on performance on mental transformation tasks [143, 144, 145, 146], suggesting the mental rotation tasks use similar brain mechanisms as those used during physical motion.

It should be noted that the true mechanism for performing mental spatial transformations is still open for debate. While we have presented evidence that humans develop an internal 3D structural model that is mentally transformed, there is another theory of internalizing representations of objects which suggests that the mechanism is more akin to language models which encode specific representations of object and environmental views [147, 148]. The question of which representation is correct has long been debated [149, 150, 151, 152, 153, 154, 148, 155, 156, 157]. Additionally, we have presented mental rotation and perspective-taking tasks as using similar mechanisms of envisioning transformations but there is evidence that the performance on allocentric spatial visualization tasks (like mental rotation of objects) differs from performance on egocentric spatial orientation tasks (like third-person-perspective taking) [135]. For instance, perspective taking tasks result in faster response times compared to mental rotation tasks [158] and, in some cases, lose the linear relationship between response time and rotation angle [159, 134, 160]. However, the mechanism that individual subjects use on each of these tasks may vary. For instance, subjects have reported solving perspective-taking tasks in an allocentric way by imagining object motion rather than self motion [161, 162].

With these caveats in mind, we focus on the development of a model that can learn transformations that may be used to model internal mental rotation. This model can be naturally applied to both egocentric and allocentric motion tasks. While there have been computational models introduced for mental rotation [163, 164, 165, 166], they have both assumed prior knowledge of the rotational transformations and been focused on modeling specific brain areas that are involved in this process rather than focusing on the representation of the transformation model itself as we do.

3.3 Model Description

We aim to develop a model for learning 3D rotational transformations from 2D projected inputs and use that model to describe how humans may employ motion cues to recover the 3D structure of objects in their environment. This perceptual setting is visualized in Figure 3.1 where an object is transforming in 3D but the visual inputs are in the form of 2D projections on the retina. In particular, each object is represented as a combination of 3D key points $\mathbf{x}^{(i)} \in \mathbb{R}^3, \ i = \{1, ..., N_P\}$ that are projected to 2D point locations $\mathbf{y}^{(i)} \in \mathbb{R}^2$. Assuming rigid body motion and incorporating a transformation model can constrain the possible 3D motion between different transformed viewpoints. This provides the structure necessary to infer depth from points on a moving object. We develop a method that uses a generative manifold model as a representation of transformations, and we show that we can learn rotational transformation operators and use them to accurately infer point depth from rotating points and scenes. We build up to the learnable model of natural transformations in two steps. In the first step, we assume the 3D rotational tarnsformation model is known and we develop a method for inferring the depth of 2D projections of rotating points. In the second step, we utilize the depth inference approach from the first step to develop a learning model that can adapt the transformation representation to ensure it corresponds to the real world transformations. We will preface the descriptions of each of these tasks with an overview of the transport operator model, a learnable generative manifold model, which was detailed in section 2.1.

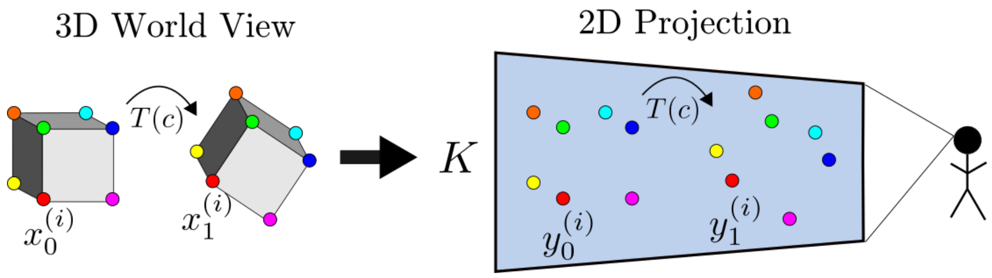


Figure 3.1: Visualization of the 3D depth inference problem. Three dimensional points $\mathbf{x}^{(i)}$ on an object are jointly transformed in the 3D world view and the visual inputs are in the form of 2D projected points $\mathbf{y}^{(i)}$. The projection matrix $\mathbf{K}$ projects all 3D points onto the 2D projection plane.

3.3.1 Transport Operator Model

As discussed in section 2.1, the transport operator manifold model learns a dictionary of M Lie group operators $\mathbf{\Psi_m}$ which represent transformations. These operators are effective for representing an internal transformation model for a few reasons. First, once learned, the transport operators are stored as a representation of possible transformations that may be experienced or observed. This means they can be reused in the future when the same type of transformation is visualized. Second, the transport operator model is a generative manifold model meaning it can interpolate and extrapolate new views of points undergoing a learned transformation. This provides a way of creating an internal visualization of how an object transforms similar to the analog spatial representation that humans describe when performing mental transformation tasks [114]. Finally, transport operators can be used to infer the relationship between points in two separate viewpoints and define the 3D transformations between them.

With the transport operator model, the relationship between two individual points $\mathbf{x}_0^{(i)}$ and $\mathbf{x}_1^{(i)}$ is defined as follows:

$$\begin{gathered}\mathbf{x}_0^{(i)} = \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m\right)\mathbf{x}_1^{(i)} + \mathbf{n}, \\ \mathbf{n} \sim \mathcal{N}(0, I) \quad c_m \sim \text{Laplace}\left(0, \frac{1}{\zeta}\right),\end{gathered} \tag{3.1}$$

where $\mathbf{c} \in \mathbb{R}^M$ is the set of coefficients that specifies the local structure of transformations between $\mathbf{x}_0^{(i)}$ and $\mathbf{x}_1^{(i)}$. Note in this chapter it is always assumed that $\mathbf{x} \in \mathbb{R}^3$, and in order to make depth inference causal, we model $\mathbf{x}_0$ as a transformation of $\mathbf{x}_1$, unlike the transport operator formulation in section 2.1. Given this relationship between points, the original work from Culpepper & Olshausen [23] defines the negative log posterior of the model as:

$$E_\Psi = \frac{1}{2}\left\|\mathbf{x}_0^{(i)} - \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m\right)\mathbf{x}_1^{(i)}\right\|_2^2 + \frac{\gamma}{2}\sum_m \|\mathbf{\Psi}_m\|_F^2 + \zeta\|\mathbf{c}\|_1, \tag{3.2}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\gamma, \zeta \geq 0$. The first term is a data fidelity term that specifies how well $\mathbf{x}_0^{(i)}$ can be represented as a transformed version of $\mathbf{x}_1^{(i)}$ when the transformations are constrained by the current dictionary of operators $\mathbf{\Psi}$. The data fidelity objective term is an indication of how well the transport operators fit the data manifold. The second term is a Frobenius norm regularizer on the dictionary elements which constrains the growth of the dictionary magnitudes and helps identify how many operators are necessary for representing transformations on the data manifold. The third term is the sparsity regularizer which encourages each transformation between point pairs to be represented with a sparse set of coefficients.

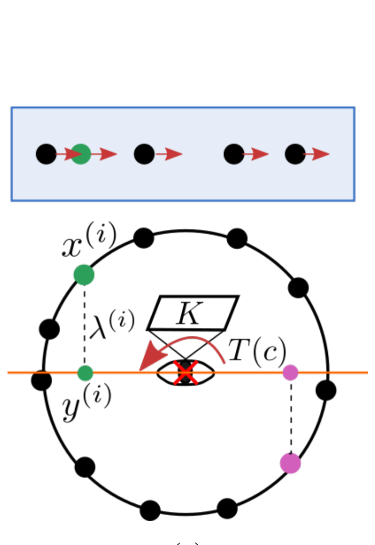
Given a dictionary of operators $\boldsymbol{\Psi}$, the 3D transformation between $\mathbf{x}_0^{(i)}$ and $\mathbf{x}_1^{(i)}$ can be estimated by inferring a set of transport operator coefficients $\mathbf{c}$. This inference is performed by minimizing E_Ψ when $\gamma = 0$. If the operators need to be learned or adapted, E_Ψ is used as an objective for transport operator training as well. Training proceeds by alternating between coefficient inference between point pairs and gradient steps on the transport operators.

We adapt the transport operator model to use time-varying views of transforming points in a 2D projection plane to learn a generative motion model. We begin by developing an inference procedure that enables joint depth estimation and coefficient inference from pairs of 2D inputs points in different viewing frames.

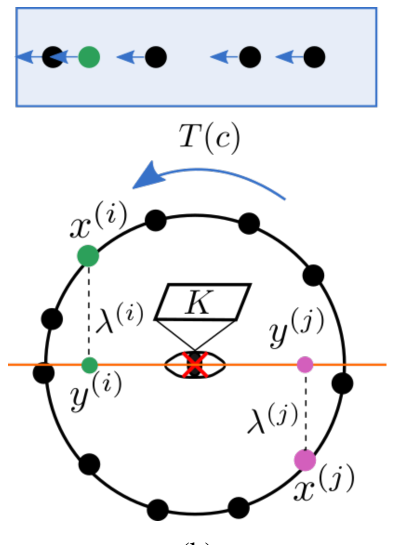
3.3.2 Depth Inference with Projected Inputs

In this section, we will assume that the transport operators are either known a priori or already learned. We describe the training procedure in subsection 3.3.4. We begin defining the 3D points $\mathbf{x}^{(i)} \in \mathbb{R}^3$ and their 2D projections $\mathbf{y}^{(i)} \in \mathbb{R}^2$. We assume that the viewer is located at the origin and that the 3D points are orthographically projected onto the viewing plane. Figure 3.2 shows a top-down view of the setup of this problem. The eye located at the red 'x' in the center represents the viewer. Each 3D point $\mathbf{x}^{(i)}$ is projected onto the viewing plane to a corresponding 2D point $\mathbf{y}^{(i)}$ with an associated depth $\lambda^{(i)}$: $\mathbf{y}^{(i)} = \mathbf{K}\mathbf{x}^{(i)}$. The matrix $\mathbf{K}$ is the orthographic projection matrix which is defined as $\mathbf{K} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ in all of our experiments. This projection matrix corresponds to setting the viewing plane to the xy-plane and defining the unknown depth as the z-coordinate of the 3D input points. It is assumed that the $\mathbf{K}$ is known during processing. We observe N_P points transforming jointly on a rigid object and concatenate all N_P points into a matrix: $\mathbf{X}_0 = \begin{bmatrix} \mathbf{x}_0^{(1)} \dots . \mathbf{x}_0^{(N_P)} \end{bmatrix}$.

In order to test the inference of structure from motion, we generate sequences with two



(a)



(b)

Figure 3.2: Top-down views of the depth inference problem setup. The 3D points $\mathbf{x}^{(i)}$ have an associated depth $\lambda^{(i)}$. Each point is projected onto the orange viewing plane using the projection matrix $\mathbf{K}$. This results in the 2D projected points $\mathbf{y}^{(i)}$. The 3D points are rotating counter-clockwise around the axis and the points in the blue shaded box on top indicate the direction of motion of the projected points. (a) Egocentric framework where the observer rotates. (b) Allocentric framework where the object rotates around the observer.

or more frames showing the projections of N_P points undergoing rotation. Unless otherwise stated, we assume ground truth knowledge of point correspondences between frames while acknowledging that identification of accurate point correspondences is an important part of structure from motion systems that should be addressed in future work. Our model uses a sequence of several frames of transforming points to reverse the projection procedure on points in the final frame in the sequence. This allows for a causal estimate of the point depths. We refer to this sequence of frames as the inference window. The relationship between points in frames at $t = 0$ and $t = 1$ is defined as:

$$\mathbf{Y}_0 = \mathbf{K}\mathbf{T}(\mathbf{c})\widehat{\mathbf{X}}_1(\lambda) + \mathbf{W}, \qquad (3.3)$$

where $\mathbf{W}$ is a Gaussian noise matrix, $\widehat{\mathbf{X}}_1$ is a matrix of estimated 3D point locations associated with $\mathbf{Y}_1$ and $\mathbf{T}(\mathbf{c})$ is the matrix exponential of a weighted combination of transport operators which can each represent a different type of motion:

$$\mathbf{T}(\mathbf{c}) = \operatorname{expm}\left(\sum_{m=1}^{M} \boldsymbol{\Psi}_m c_m\right). \tag{3.4}$$

To compute $\widehat{\mathbf{X}}_1$, we reverse the process of the projection matrix in two steps. First, we concatenate the $\mathbf{Y}_1$ with a vector of zeros in the z-coordinate position that was lost during projection:

$$\bar{\mathbf{X}}_1 = \begin{bmatrix} \mathbf{y}_1^{(1)} & \cdots & \mathbf{y}_1^{(N_P)} \\ 0 & \cdots & 0 \end{bmatrix}. \tag{3.5}$$

The second step for reversing the projection process is adding the depth to the newly introduced dimension. To do this, we compute the outer product between the standard basis vector associated with the axis lost during projection $\mathbf{e}_z$ and the depth vector $\lambda \in \mathbb{R}^{N_P}$:

$$\widehat{\mathbf{X}}_1(\lambda) = \bar{\mathbf{X}}_1 + \mathbf{e}_z \lambda^\top. \tag{3.6}$$

This model for incorporating the depth can be integrated into the data fidelity term of the objective function in Equation 3.2 and used for inferring the depth λ and coefficients $\mathbf{c}$ between point pairs.

$$\begin{aligned} L_{\text{df}} &= \frac{1}{2}\sum_{i=1}^{N_P} \|\mathbf{y}_0^{(i)} - \mathbf{K}\mathbf{T}(\mathbf{c})\widehat{\mathbf{x}}_1^{(i)}\left(\lambda^{(i)}\right)\|_2^2 \qquad (3.7)\\ &= \frac{1}{2}\text{trace}\left(\left(\mathbf{Y}_0 - \mathbf{K}\mathbf{T}(\mathbf{c})\widehat{\mathbf{X}}_1(\lambda)\right)^\top \left(\mathbf{Y}_0 - \mathbf{K}\mathbf{T}(\mathbf{c})\widehat{\mathbf{X}}_1(\lambda)\right)\right). \qquad (3.8) \end{aligned}$$

We add two additional constraints to this model to improve the consistency of accurate depth estimation. First, we incorporate a Gaussian prior on the depths which constrains them to magnitudes consistent with the size of the rotating objects. Second, we group several consecutive views of the transforming points together into our inference window to infer a fixed set of coefficients that represents the transformations between each consecutive view. This assumes a fixed transformation speed over multiple frames. This is consistent with the slowness principle which posits that natural variations are persistent in time [167]. Figure 3.3 shows this setting where the same coefficients $\mathbf{c}$ are inferred between points in each neighboring frame and the depth is inferred for the final point in the sequence. Using more than two motion frames for depth inference provides additional information that can be used to resolve depth ambiguities. To model this setting, we generalize Equation 3.3 for N_T viewpoints:

$$\mathbf{Y}_{N_T-n} = \mathbf{K}\mathbf{T}^n(\mathbf{c})\widehat{\mathbf{X}}_{N_T}(\lambda) = \mathbf{K}\mathbf{T}(n\mathbf{c})\widehat{\mathbf{X}}_{N_T}(\lambda)\,, \quad n = \{1, ..., N_T\}, \tag{3.9}$$

where the change from $\mathbf{T}^n(\mathbf{c})$ to $\mathbf{T}(n\mathbf{c})$ is possible because the coefficients $\mathbf{c}$ are weights for terms in a summation in a matrix exponential and raising an exponent to the power of n is the same as multiplying its exponential term by n.

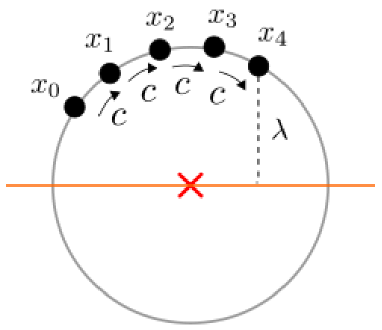


Figure 3.3: Visualization of the inference window sequence for a single point. The inference window is made up of several frames of transformed points. We assume that the transformation speed is constant between the frames, resulting a constant coefficient value representing the transformation from one frame to the next. The depth λ is inferred for the final frame in the sequence.

We define an objective that leverages multiple views and a depth regularizer:

$$L = \frac{1}{2N_T}\sum_{n=1}^{N_T}\sum_{i=1}^{N_P}\left[\|\mathbf{y}_{N_T-n}^{(i)} - \mathbf{K}\mathbf{T}(n\mathbf{c})\widehat{\mathbf{x}}_{N_T}^{(i)}(\lambda)\|_2^2\right] + \zeta\|\mathbf{c}\|_1 + \frac{\beta}{2}\|\lambda\|_2^2 + \frac{\gamma}{2}\sum_m \|\mathbf{\Psi}_m\|_F^2. \tag{3.10}$$

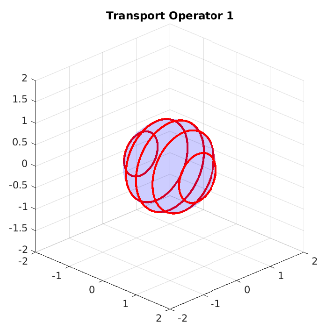
With this objective, the depth λ and the coefficients $\mathbf{c}$ can be jointly inferred for sequences of transforming points. See section A.1 for more details on the inference process.

It should be noted that this model can be applied to two viewing frameworks – the allocentric framework in which points rotate around the observer and the egocentric framework in which the observer rotates with respect to the surrounding world. When the motion is centered around the origin (i.e., when the origin of the observer coordinate system and the world coordinate system is the same), the allocentric and egocentric learning frameworks utilize the exact same model. As shown in Figure 3.2, the only difference between the allocentric and egocentric frameworks when motion is centered at the origin is the direction in which the projected points move with respect to the rotational motion. When the viewer rotates in a counter-clockwise direction in the egocentric framework, the projected points with positive depth (i.e. points in front of the viewer) move to the right in the viewing plane. On the other hand, when the points rotate in a counter-clockwise direction in the allocentric framework, the projected points with positive depth move to the left in the viewing plane. Therefore, going forward, the model and experiments can correspond interchangeably to the egocentric or allocentric frameworks.

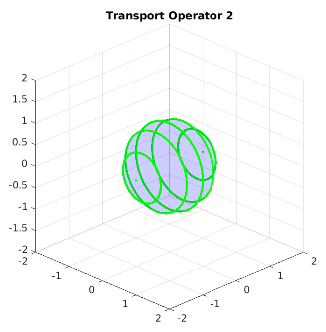
To highlight the effectiveness of this inference model, we will examine how accurately it can infer depths and transformations using ground truth rotational operators and explore the requirements for the inputs that lead to robust depth estimation.

3.3.3 Depth Inference Experiments

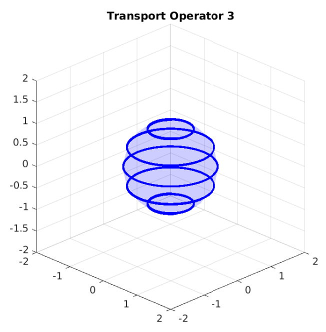
Three-dimensional rotational matrices can be defined as elements of the 3D rotational group SO(3) and ground truth rotational transport operators can be derived from elements of the $\mathfrak{so}(3)$ Lie algebra [168]. Figure 3.4 shows the trajectories of these ground truth 3D rotational operators, each rotating around one of the principal axes. These plots are generated by selecting a few example starting points on a sphere and applying individual dictionary elements $\mathbf{\Psi_m}$ to each point as they evolve over time: $\mathbf{x}_t^{(i)} = \mathrm{expm}(\mathbf{\Psi}_m \frac{t}{T})\mathbf{x}_0^{(i)}$,



(a)



(b)



(c)

Figure 3.4: Trajectories generated by ground truth rotational transport operators. Each line represents the trajectory of an individual transport operator dictionary element applied to one of several example starting points selected on the sphere. These three operators generate rotation around each of the three principle axes.

$t = 0, ..., T$.

Using Equation 3.10 and the fixed $\mathbf{\Psi}$ representing ground truth rotational operators, we jointly infer the coefficients and depths in sequences of transforming points. Figure 3.3 shows a visualization of this inference setting for a single point. Given N_T views of rotating points, we infer the depth λ for the projected points in the last viewpoint $\mathbf{Y}_{N_T}$ as well as the coefficients $\mathbf{c}$ that correspond to the shared transformation between every pair of consecutive views in the sequence. This ensures that the depth as time N_T is inferred by samples preceding it in the motion sequence, making this a causal system. Figure 3.5 shows examples of depth inference for points on the surfaces of three different shapes. The plots in the first column show the visual input of projected points in the final viewing plane of the sequence. The plots in the second column show a side view of the point stimuli where the ground truth depth locations are shown on the x-axis of the plot. In these plots the points are colored by the ground truth depths. The plots in the third column show a side view with the estimated depth locations for each of the points. The points are again colored by the ground truth depths. This shows that the estimated depths correspond with the ground truth depths for a variety of shapes.

We quantitatively evaluate the accuracy of the inferred depths for many trials to analyze

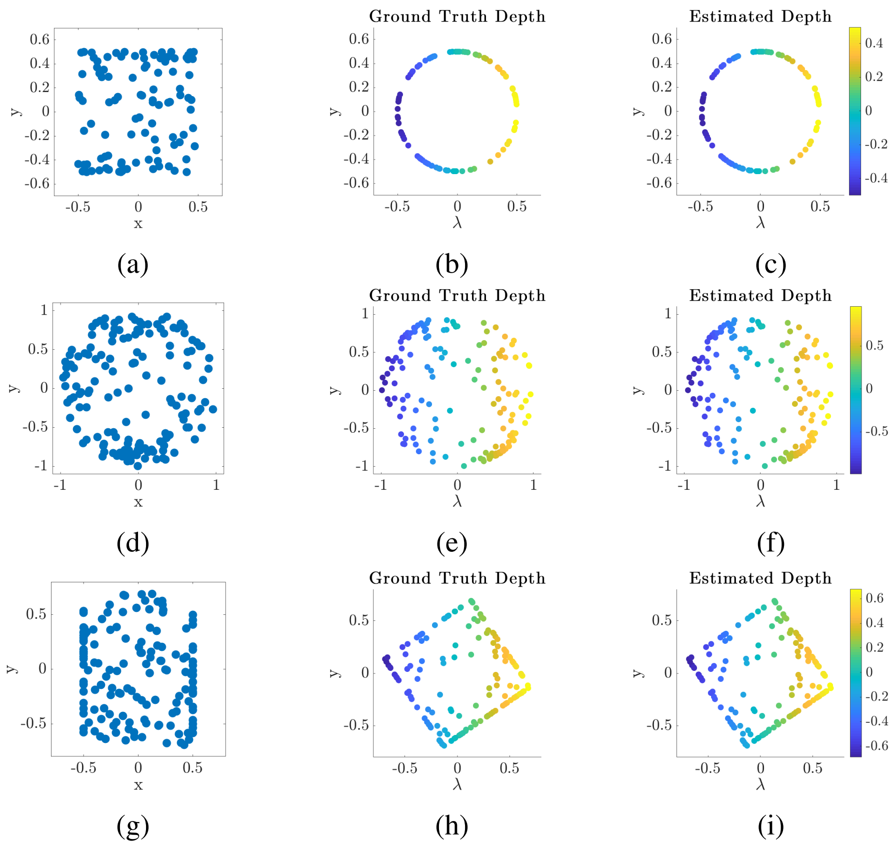


Figure 3.5: Examples of inferred depths for points on the surface of different shapes. Plots in the first column show the 2D point projections in the final viewing plane. Plots in the second column show a side view of the ground truth 3D point stimuli where the x-axis in the plots is the depth axis. The points are colored by the ground truth depth. Plots in the third column show a side view of the estimated depths for the projected points. The points are colored by the ground truth depth. (a-c) Points on a cylinder (d-f) Points on a sphere (g-i) Points on a cube.

the impact of various model parameters. There are three parameters of particular interest during inference. The first is the number of jointly transforming points N_P. This indicates the amount of coherent rotational motion viewed in the input stimulus. Psychophysical experiments have been run that indicate a greater number of coherently moving points leads to a more robust depth percept [100, 101, 102, 99]. Next we are interested in the impact of the perceptual extent of rotation viewed in a sequence of frames. The perceptual extent of rotation is a combination of two parameters, the number of frames in a rotation sequence N_T and the ground truth rotation angle between each frame in the rotation sequence θ. The full angular extent of rotation viewed is $\theta_{\text{path}} = N_T \theta$.

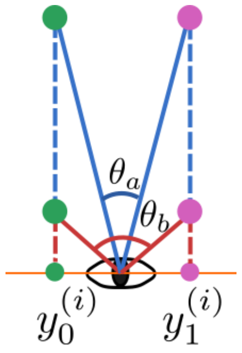
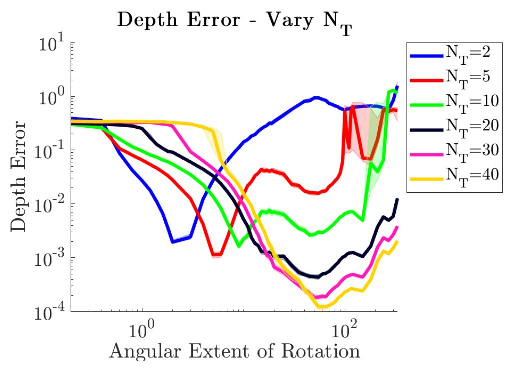
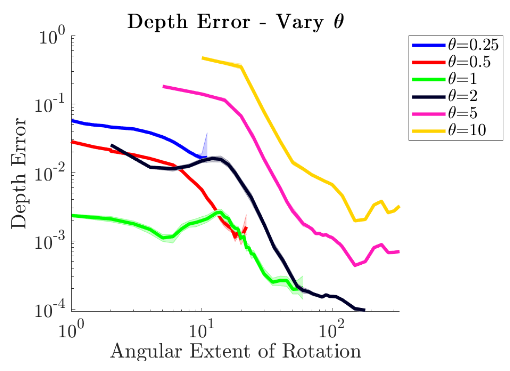


Figure 3.6: Example of depth-angle ambiguity. Points $\mathbf{y}_0^{(i)}$ and $\mathbf{y}_1^{(i)}$ are projections of a 3D point that is transformed from view 0 (green points) to view 1 (pink points). These points could result from a small rotation angle θ_a on points with large depth magnitudes or a larger rotation angle θ_b with smaller depth magnitudes.

In order to quantitatively evaluate the success of inferred depth, we use two metrics. The first is the mean squared error between the estimated depths and the ground truth depths for the N_P rotating points. Ideally depth inference would result in low MSE between the estimated depth and the ground truth depth. However, with a rotational transformation, as we are working with here, there exists a depth-angle ambiguity. Namely, when viewing projected points $\mathbf{y}_0^{(i)}$ and $\mathbf{y}_1^{(i)}$ from two separate views, they could be either projections of points with large depths that undergo rotation with a smaller angle θ_a or points with small depths that undergo rotation with a larger angle θ_b. These two possibilities are visualized in Figure 3.6. While it is not ideal for the depth to be off by a scaling factor, the inferred structure can still be accurate. Additionally, this depth ambiguity is observed in experiments with human subjects [100]. In psychophysical experiments, one metric for determining accuracy of a percept is comparing the estimated ordering of point depths to the ground truth ordering of point depths [169]. To analyze the accuracy of the inferred structure in the presence of potential scaling in depth, we compare the ordering of the inferred depths to that of the ground truth depths using the Kendall's Tau rank correlation coefficient [170]. We compare the Kendall's Tau between all N_P points as well as between five randomly selected points. We choose to compare the ordering of five randomly selected



(a)

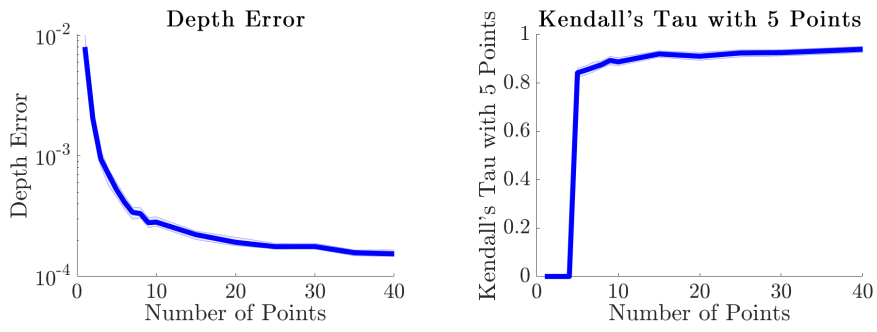


(b)

Figure 3.7: Quantitative metrics for depth inference when varying angular extent of rotation. (a) Median depth error as angular extent increases. Each line is generated with different numbers of frames in the inference window N_T. The optimal performance occurs for angular extents in the range of N_T to $2N_T$. (b) Median depth error as angular extent increases. Each line is generated with different angles of rotation between sequence frames θ. A rotation angle of $\theta = 2°$ results in the lowest depth error at $180°$ of rotation.

points in order to define a metric that can be used to fairly compare the performance as the number of points increases. With greater N_P, even if depths are accurate within some error range, there is a greater chance of incorrectly ordering a few points because there is a greater point density. Therefore, comparing five randomly sampled points should provide a consistent metric as we vary N_P.

Figure 3.7 shows the median depth error as we vary the angular extent of rotation θ_{path}. In Figure 3.7a, each line represents a different number of frames N_T. The error bars in all plots in this chapter represent the bootstrap confidence interval. For each line in Figure 3.7a, because the values of N_T are fixed, moving along the x-axis corresponds to increasing the angle between frames. Each of these lines has a clear minimum and this minimum occurs at an angular extent in the range of N_T to $2N_T$. This corresponds to an angle θ between frames of $1° - 2°$. Up to this minimum value, the depth error decreases as the rotational extent increases, indicating the benefit of a more complete view of the rotational sequence. Figure 3.7b breaks down the performance for individual values of θ. As θ increases up to $\theta = 2°$, the depth error decreases with increasing θ. For $\theta \geq 5°$, the performance starts to degrade. This is consistent with the patterns in Figure 3.7a, and it indicates that rotation



(a) (b)

Figure 3.8: Quantitative metrics for depth inference when varying the number of coherently transforming points N_P. (a) Median depth error as N_P increases. The depth error decreases as the number of points increases and there is a large decrease in depth error from $N_P = 1$ to $N_P = 10$. (b) Mean Kendall's Tau for 5 randomly selected points as N_P increases. Values of this metric for $N_P < 5$ are set to zero because there are not enough points to compare five randomly selected points. Kendall's Tau increased as N_P increases.

angles between frames that are too large (corresponding to fast rotational motion) result in less accurate depth inference[2]. In the remaining tests of inference performance with ground truth operators, unless otherwise stated, we set $N_T = 30$ and $\theta = 2°$. See section A.1 for more details on model parameters.

Figure 3.8 shows the median depth error and the mean Kendall's Tau as we vary N_P. This shows that the depth estimation improves as more points are added with a large performance improvement from $N_P = 1$ to $N_P = 10$. Going forward, we use $N_P = 20$. Research in structure from motion has shown that increasing the number of transforming points improves the general depth percept [100, 101, 102] but it may not increase the accuracy of the inferred depths [99].

The final quantity we analyze in this controlled setting with ground truth rotational operators is the effect of adding Gaussian noise to the operators. Noisy operators depart from the ground truth rotational transformations and analyzing the performance with noisy operators can indicate the influence of accurate rotational transformation models on effec-

[2]We should note that optimization experiences instability as greater number of frames are used in the inference window, leading to large increases in depth error for $N_T > 50$ in many settings. Therefore, in Figure 3.7b, we only display lines until an angular extent of rotation for which instability impacts the optimization. The angular extent where this occurs is smaller for the smaller values of θ because they require larger values of N_T to achieve the same angular extent of rotation.

tive depth inference. To provide an intuitive understanding of the effect of noise on the operators, Figure 3.9 shows the trajectories for operators with increased noise standard deviation. Figure 3.10 shows the median depth error and mean Kendall's Tau metrics as noise is added to the ground truth operators. Both metrics indicate that the depth inference is robust to noise with a standard deviation of around 10^{-3} - 10^{-2} but performance decreases sharply with noise larger than that. This shows that the model can perform effectively with some transformation inaccuracy but performance decreases with increasingly inaccurate transport operators. This highlights the necessity of accurate rotational operators and inspires the learning and adaptation procedure introduced and analyzed in the next section.

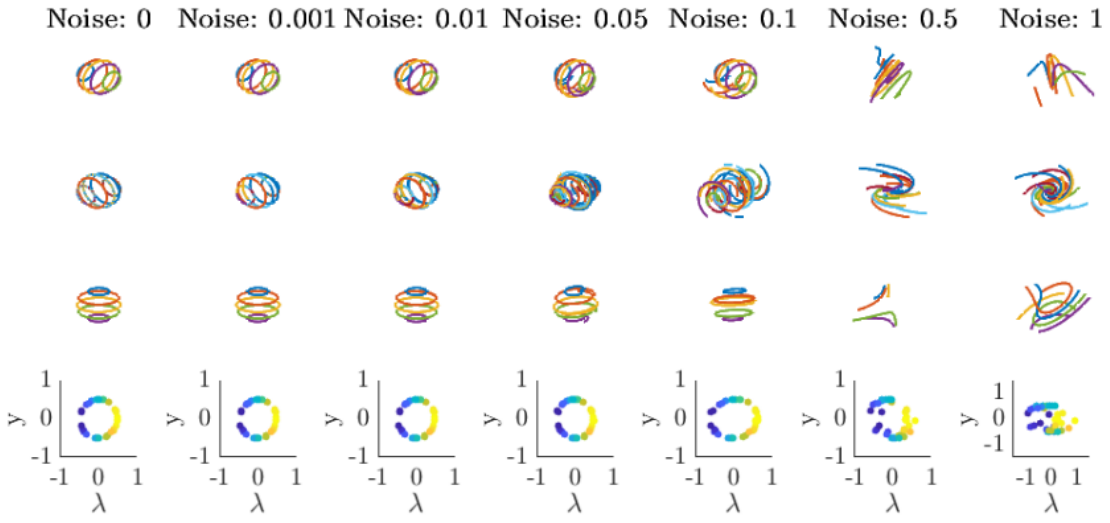
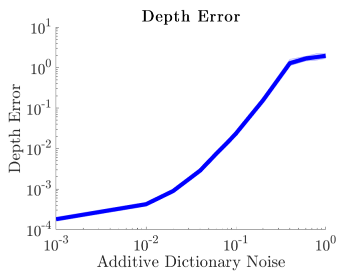


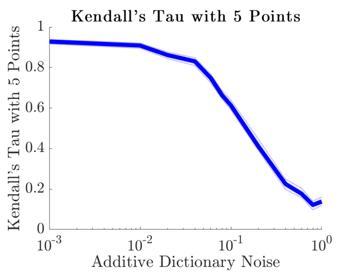
Figure 3.9: Examples of noisy operator trajectories. Each column shows examples of the ground truth rotational operators with additive Gaussian noise with increasing standard deviation. Rows 1-3 show the trajectories for the three rotational operators. Row 4 shows depth inferred for points projected from a rotating cylinder using the three operators in each column. The operators do not vary much in appearance from the ground truth operators (first column) with noise standard deviations of 0.01 or less. For noise standard deviations larger than that, the operators diverge from rotational operators and the point depths no longer look like a cylinder.

3.3.4 Learning 3D Transport Operators from 2D Projected Inputs

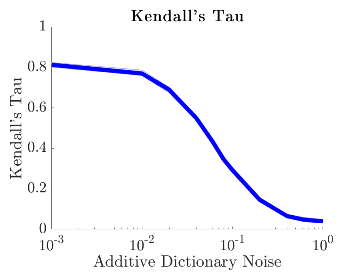
The stated goal of this work is to develop a model that can learn 3D transformational representations from rotating 2D projected input points. The learning procedure is a straight-



(a)



(b)



(c)

Figure 3.10: Quantitative metrics for depth inference when varying the standard deviation of Gaussian noise added to the ground truth operators. (a) Median depth error as the standard deviation of dictionary noise increases. The depth error remains relatively low until a noise standard deviation of around 10^{-2} after which the depth error increases significantly. (b-c) Mean Kendall's Tau for 5 randomly selected points and all 20 stimulus points respectively. The Kendall's Tau values remains high until noise standard deviations greater than 10^{-2}.

forward extension of the coefficient and depth inference model from the previous sections. Training of the transport operator dictionary elements is performed using gradient descent. For each training step, a sequence of projected rotated points $\mathbf{Y}_n,\ \ n = \{1, ..., N_T\}$ is generated. First the dictionary weights are fixed and the depth and coefficients are inferred. Then, fixing the depth and coefficients, the gradient on the dictionary elements is computed using the objective in Equation 3.10 with $\zeta = \beta = 0$. If this gradient step improves the objective, then it is accepted. Otherwise, the step is rejected and the learning rate is decreased. See section A.2 for more details on the training procedure.

With this training procedure, we are able to learn rotational transport operators from randomly initialized operators. Figure 3.11 shows the trajectories of the operators during one training run in which the number of dictionary elements M is set to 3. At the beginning of training, the trajectories do not correspond to common geometric transformations but they quickly adapt to represent near-rotational operators with trajectories similar to the ground truth operators shown in Figure 3.4. In the section A.2 we show an example of learning rotational operators from a dictionary with six operators.

We quantitatively compare these operators to the ground truth operators using the same depth MSE and Kendall's Tau metrics employed to analyze inference success. We can

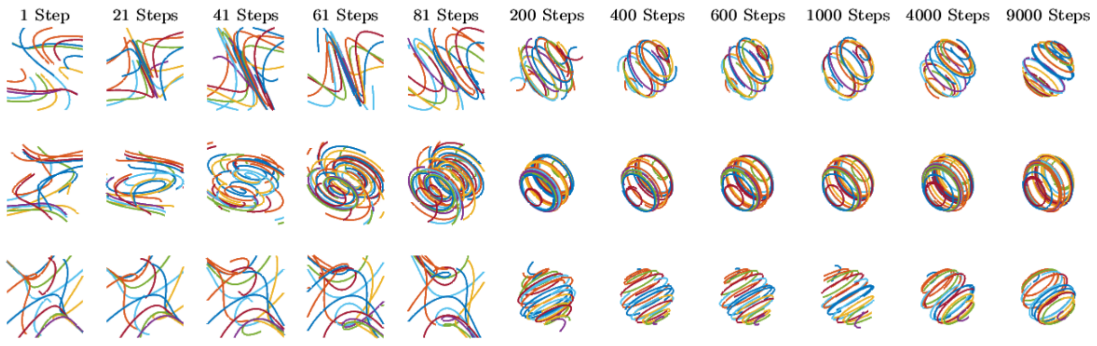
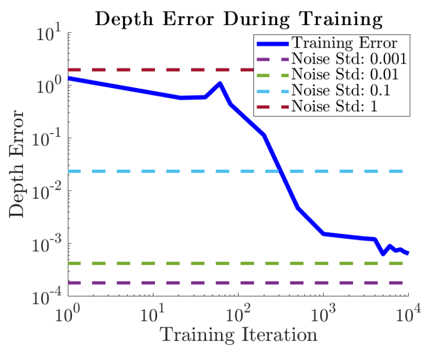
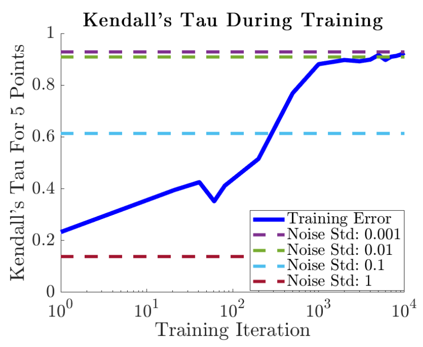


Figure 3.11: Transport operator trajectories during training. Each row represents one of the three learned operators. Each column shows the trajectories at a different training step. The operators begin with random initializations at step 1 and quickly reach a rotation structure around 200 steps. From 200 steps to 9000 steps, the operators vary relatively slowly, resulting in operators with clear rotational structure at the end of training.

compute the depth error and Kendall's Tau metrics for inferred depths using operators at various points during training and compare them to the metric values resulting from depth inference using the ground truth operators with noise added. This gives us a proxy for estimating the deviation between the learned operators and ground truth rotational operators. In Figure 3.12, we show the depth error and Kendall's Tau for depths inferred using transport operators at different points in the training procedure. For reference, we also plot straight lines which correspond to the values for these metrics in Figure 3.10 which are computed using ground truth rotational operators with added noise with standard deviations of 10^{-3}, 10^{-2}, 10^{-1}, and 1. This shows that our method learns operators that are close in structure to the ground truth operators and the performance they achieve is similar to ground truth operators with additive Gaussian noise with a standard deviation of 10^{-2}. As we show in Figure 3.10, transport operators with Gaussian noise with a standard deviation of 10^{-2} achieve low depth errors and large Kendall's Tau values. Additionally we see that the depth inference performance with learned operators improves significantly over the first 100-200 training steps but requires fine-tuning for many steps after that to achieve optimal performance.



(a)



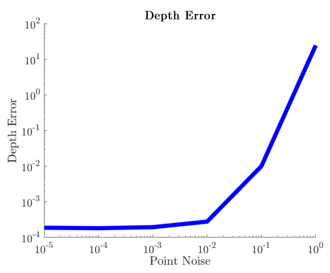
(b)

Figure 3.12: Inferred depth metrics using operators at different steps in training. Dotted lines represent values of error metrics for depths inferred using ground truth operators with additive Gaussian noise with the standard deviation specified in the legend. These values are obtained from the plots in Figure 3.10a. (a) Median depth error for depth inference performed with operators at different steps in training. The depth error decreases significantly after 200 training steps and continues to decrease until the end of training. The depth error achieves a value consistent with the estimates using ground truth operators with a additive Gaussian noise with a standard deviation of 10^{-2}. (b) Mean Kendall's Tau for 5 randomly selected points. The Kendall's Tau increases significantly around 200 training steps and starts to plateau around 1000 training steps. The Kendall's Tau reaches a value consistent with ground truth operators with a noise standard deviation of 10^{-3}

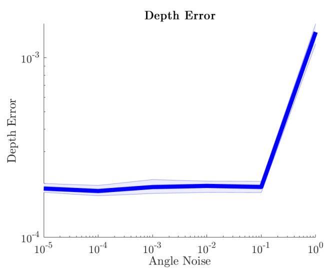
3.4 Model Analysis

In this section, we analyze the robustness of our model to a mismatch between the model assumptions and data characteristics. The first assumption is that we have ground truth keypoints locations. We analyze the model robustness to noise in the keypoints locations when using ground truth rotational operators by adding Gaussian noise with progressively larger standard deviations to the 2D point locations and observing the variations in the depth error (Figure 3.13a). For reference, points in these experiments have values in the range of $[-1, 1]$. Results indicate that the model is robust to noise with a standard deviations of up to around 10^{-2}.

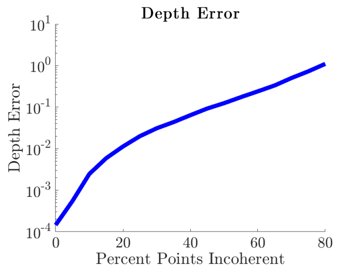
Next we analyze the assumption that each frame in the inference window results from rotations with the same speed and direction as preceding frames. In natural systems, objects are unlikely to undergo rotation at the exact same speed over many frames. To test the



(a)



(b)



(c)

Figure 3.13: Depth error for depth inference in the presence of mismatch between the model assumptions and stimulus characteristics. (a) Depth error as the standard deviation of additive Gaussian noise in the 2D point locations increases. The model is robust to noise with a standard deviation up to around 10^{-2}. (b) Depth error as we increase the standard deviation of the Gaussian noise added to the magnitude of the rotation angles between frames in the inference window θ. The model is robust to noise with a standard deviation up to around 10^{-1}. (b) Depth error as the percent of incoherently moving points is increased. The depth error quickly increases after $5 - 10\%$ of points are incoherent.

effect of variations in the rotation speed, we create rotation sequences in which each frame uses the same axis of rotation but the magnitude of the rotation angle is varied by adding Gaussian noise with a standard deviation scaled by the angle of rotation[3]. In this setting, we infer the depth and coefficients using ground truth rotational operators and compute the depth error shown in Figure 3.13b. Our model is robust to noise in rotation angle up to a standard deviation of around 10^{-1}. Finally we analyze the performance of our model in the presence of points that are moving incoherently with the rotating points. Figure 3.13c shows the depth error as we increase the percentage of incoherently moving points when $N_P = 100$. This shows that incoherent points significantly impact the accuracy of the 3D structure estimate.

3.5 Kinematogram Experiments

Random dot kinematograms are displays of dots on the surface of or within invisible rotating shapes. Still frames of kinematogram inputs appear as random dots with no perceptible structure (see Figure 3.14a). However, the motion of the dots elicits the perception of a 3D

[3]The angles used in this experiment have a mean of $2°$.

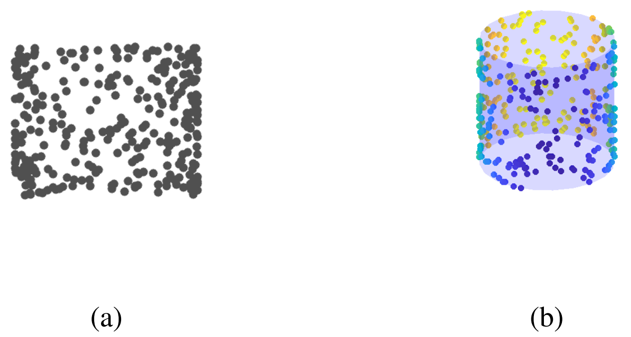
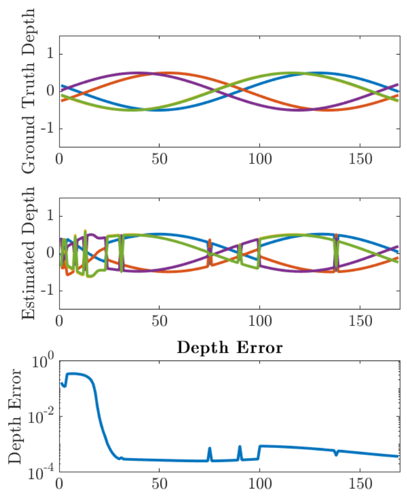


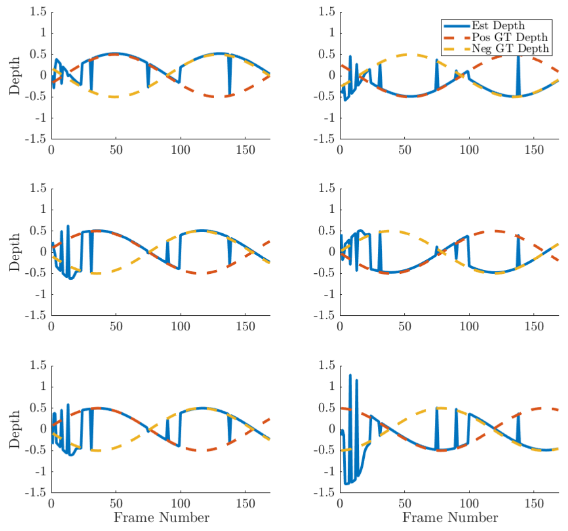
Figure 3.14: Visualization of kinematogram visual stimuli. (a) Example of a 2D kinematogram stimulus which is the projection of random dots on the surface of a cylinder. (b) 3D ground truth structure of the points in the kinematogram stimulus. The points are randomly sampled from the cylinder surface and colored by their depth.

structure. Figure 3.14 shows the 2D projection of random points along with the 3D structure of the points on the surface of a cylinder. This perception of depth through motion is termed the ”kinetic depth effect” [121]. The random dot kinematogram visual stimulus has been used for many structure from motion experiments because it isolates the use of motion cues from the use of other possible depth cues. We use our depth inference model with transport operators learned from 2D projections of rotational motion in order to estimate depths for random points that are located within the volume of invisible rotating shapes. We compare characteristics of our experimental results to the performance of humans on structure from motion tasks with random dot kinematograms.

For these experiments, we create kinematogram stimuli by randomly selecting N_P 3D points within the volume of a cylinder. Sequences are generated by rotating points around the x-axis at a rotational speed specified by the angle θ. The points in each frame are ortographically projected to the xy-plane. Point correspondences are estimated by pairing nearest neighbors in the projected inputs from one frame to the next using the Hungarian algorithm [171]. We use the inference procedure described in subsection 3.3.2 to infer the depth and coefficients. During inference, we use an inference window of $N_T - 1$ preceding frames to infer depths for the points in the current frame. We can vary the parameters of



(a)



(b)

Figure 3.15: Example of depths inferred for random dots in a kinematogram sequence. (a) In the top plot, each line represents the ground truth depth of a single random point over the rotational sequence of the kinematogram. In the middle plot each line represents the estimated depth for the same points as in the top plot. The bottom plot shows the depth error between the estimated and ground truth depths over the sequence. (b) Each plot shows the estimated depth for a single point with the sequences of positive and negative ground truth depths overlaid.

the stimuli and the inference procedure and analyze their impact.

Figure 3.15 shows depths that are inferred for a random dot kinematogram sequence on a cylindrical structure by minimizing the objective in Equation 3.10. In this experiment, $N_P = 20$, $\theta = 2°$ and $N_T = 30$. Each line in the top and middle plot of Figure 3.15a is the depth for one of five stimulus points. In the early stages of the kinematogram sequence, the number of frames in the inference window is only as large as the number of frames that has appeared (which is less than N_T). Once more than N_T frames have appeared, the depth and coefficients inference will make use of only the current frame and the $N_T - 1$ preceding frames. This build up in the angular extent of rotation explains the larger depth errors early in the sequence, and we will analyze this further below.

The estimated depth in Figure 3.15 has discontinuities that result from the sign of the depth switching. This natural phenomenon is due to the fact that the orthographically projected random dot kinematogram stimulus is a bistable perceptual representation [172]. That is, it is an ambiguous representation in which there are two correct perceptual structures. All the points could be rotating in a clockwise direction points with a specific combination of positive and negative depths or they could be rotating in a counterclockwise direction with the opposite combination of positive and negative depths. Each of these perceptual estimates is equally correct for the stimulus. Therefore, when computing the error metrics, we correct for the direction of the inferred rotation so it corresponds to the ground truth direction (which is clockwise in all of our experiments). This is done by generating a path with the inferred transformation coefficients and identifying the rotation direction of the points on that path. If the inferred rotation is moving in a counterclockwise direction, we reverse the signs of the depths prior to computing the error metrics. The bottom plot Figure 3.15a shows the depth error for the kinematogram sequence. The depth error is high at the beginning of the sequence due to the limited angular extent of rotation. As the angular extent of rotation increases, the depth error decreases and remains low even while the signs of the depths switch. In Figure 3.15b, we overlay the estimated depth for individual points on top of sequences with both of positive and negative ground truth depth values. This shows, whichever direction of rotation is inferred, the depths are aligned with either the positive or negative ground truth depth values.

This bistable phenomenon is observed in pyschophysical experiments as well. Specifically, subjects incorrectly identify the rotation direction of orthographically projected stimuli 50.3% of the time [98]. In the experiments shown in Figure 3.17c, the clockwise rotation is estimated 50.04% of the time.

Our model has the capability of reducing the amount of depth sign switching and transformation direction switching by adding a dynamic regularizer during inference. We assume that there is a constant speed of rotation and encourage the transport operator coefficients to be similar from one frame to the next. Note that we already utilize this coefficient consistency in the inference procedure by inferring the same coefficients for all frames in the inference window. The additional regularizer encourages the set of coefficients to be similar from one inference window to the next. This amounts to adding a term to the inference objective that incorporates the previous coefficient estimate:

$$\begin{aligned} L = \frac{1}{2N_T} \sum_{n=1}^{N_T} \sum_{i=1}^{N_P} \left[\|\mathbf{y}^{(i)}_{N_T - n} - \mathbf{KT}(n\mathbf{c})\widehat{\mathbf{x}}^{(i)}_{N_T}(\lambda)\|_2^2 \right] + \zeta \|\mathbf{c}\|_1 \\ + \frac{\beta}{2}\|\lambda\|_2^2 + \frac{\xi}{2}\|\mathbf{c} - \mathbf{c}_{\text{prev}}\|_2^2. \end{aligned} \tag{3.11}$$

With the addition of the dynamic regularizer on the coefficients, we infer smooth depth estimates for kinematogram sequences. Figure 3.16 shows the estimated depths and depth error with dynamic regularization. This regularization eliminates sign-flipping but it also impacts the depth error in the beginning frames of the kinematogram sequence. The initial depth estimates are less accurate because they use only a small window of frames for depth inference and the coefficient regularizer encourages coefficient estimates in later frames to be similar to the initial inaccurate estimates. The depth estimates eventually achieve low error as the inference window gets larger.

Figure 3.17 contains plots demonstrating the performance of our model on random dot kinematogram stimuli as we vary parameters of both the inference algorithm and the kinematogram inputs. These plots compute the depth MSE for all N_P points and the Kendall's Tau for 5 randomly selected dots in the stimulus. Figure 3.17a and Figure 3.17b each show the impact of including the dynamic regularizer from Equation 3.11. When $\xi = 0$, the median depth error looks the same as in Figure 3.15, where it begins high due to the limited angular extent of rotation and decreases as the kinematogram sequence continues. The Kendall's Tau values with both $\xi = 0$ and $\xi = 20$ have a spike at the beginning of the kinematogram sequence. Without the dynamic regularizer, the Kendall's Tau value plateaus and remains around the same value until the end of the sequence. As we saw in Figure 3.16, the integration of the dynamic regularizer on the coefficients leads to a delay in achieving optimal accuracy for depth estimates measured by both the depth MSE and Kendall's

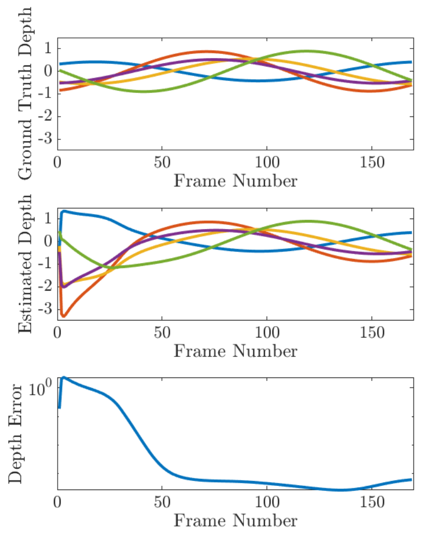


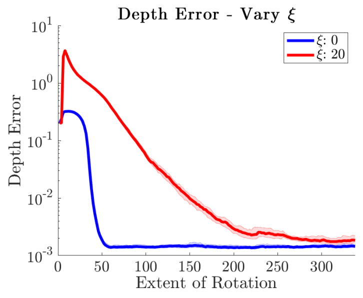
Figure 3.16: Example of depths inferred for random dots in a kinematogram sequence with a dynamic regularizer. In the top plot, each line represents the ground truth depth of a single random point over the rotational sequence of the kinematogram. In the middle plot each line represents the estimated depth for the same points as in the top plot. The bottom plot shows the depth error between the estimated and ground truth depths over the sequence. The dynamic regularizer removes the sign switching of the depth values observed in Figure 3.15 but it also leads to larger error early in the kinematogram sequence.

Tau. We examine the influence of the number of stimulus points in Figure 3.17c and Figure 3.17d[4]. Increasing the number of points improves the accuracy of the depth estimates but does not significantly impact the accuracy of the depth ordering. However, it is notable that decreasing the number of points eliminates the spike in the Kendall's Tau at the very beginning of the rotational sequence[5]. Finally, we examine the impact of adding Gaussian noise to the point locations in Figure 3.17e and Figure 3.17f. Similar to the results we saw in Figure 3.13b, the depth is consistently accurate with point location noise up to a standard deviation of 10^{-2} and depth error increases after that. The introduction of point noise also eliminates the spike in Kendall's Tau at the beginning of the sequence.

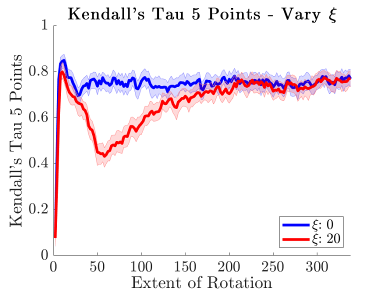
This perceptual build up of an accurate estimate of point depths is observed in structure

[4]Note in this experiment we use ground truth correspondences between sample points in each kinematogram frame in order to focus on the impact of N_P on the inferred depth sequence independent of our point correspondence technique.

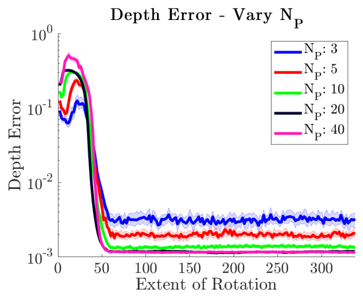
[5]The Kendall's Tau we report in Figure 3.17d is for five randomly selected stimulus points so the value for $N_P = 3$ is set to zero because there are fewer than five points to use for this metric computation.



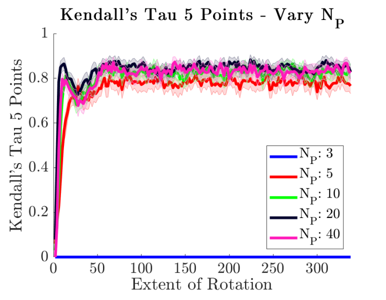
(a)



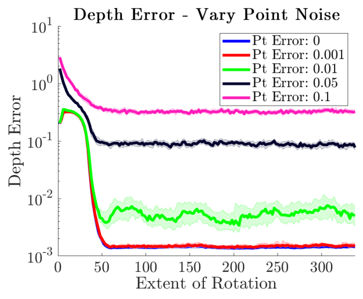
(b)



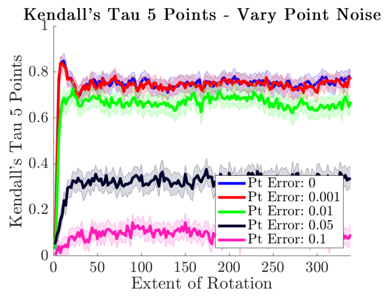
(c)



(d)



(e)



(f)

Figure 3.17: Quantitative metrics for random dot kinematogram depth estimates. Depth error and Kendall's Tau: (a-b) with and without a depth regulerizer. (c-d) as N_P increases. (e-f) as the standard deviation of noise added to the point locations increases.

from motion experiments [169]. Hildreth, Grzywacz, Adelson, & Inada perform experiments where they display orthographic projections of three points rotating about a central axis and ask subjects to order the depths of the three points [169]. They computed the percentage of correct ordering responses - a metric similar in nature to our Kendall's Tau metric. They found that the percent of correct depth ordering increased as the angular extent of rotation increased up to about $40°$ of rotation after which it plateaued. We observe the same build up and plateau of point ordering accuracy (as judged by Kendalls' Tau). They also observed degredation in performance as Gaussian noise was added to the point locations as we see in Figure 3.17f.

3.6 Discussion

In this chapter we develop a method for learning 3D manifold-based transformation models from 2D training stimuli using an inference procedure that jointly estimates point depth and transformation coefficients. We show that we can learn dictionary elements that generate rotational motion from 2D projections of rotating points. This model lays the groundwork for explaining the development and adaptation of internal representations of natural variations that are observed in the world. Additionally, our depth inference model enables the investigation of data characteristics that may influence the capacity for accurate depth estimation. This allows us to connect model performance with various data characteristics and algorithmic parameters to human performance on perceptual studies.

We show the importance of a large angular extent of the rotation stimuli for accurate depth estimation and ordering (see Figure 3.7 and Figure 3.17). This supports the notion that humans build up their perception of 3D structure during random dot kinematogram rotation sequences [169]. We also show that increases in the number of random dot stimuli result in improvements in depth inference performance (see Figure 3.8, Figure 3.17c, and Figure 3.17d). This connects to kinematogram experiments that indicate a greater number of coherently transforming points results in a stronger depth percept from moving points [100, 101, 102, 99]. Our model also demonstrates the same direction switching phenomenon with the bistable kinematogram stimulus that humans perceive [98].

3.6.1 Future Investigations

Our model has the flexibility to adapt to many different test scenarios that are inspired by human performance on mental rotation and structure from motion tasks. In our experimentation, we tuned parameters like the inference window length N_T, the angle between frames θ, the number of stimulus points N_P, and the weight on the dynamic regularizer ξ to achieve the most accurate depth estimates. However, experiments show, even when humans perceive the correct shape, they often have inaccurate estimates of depth magnitude [100], especially when viewing limited numbers of transforming points [101]. From our model performance, this may suggest that humans rely on a smaller angular extent of rotation for inferring depth or that they do not utilize a prior on expected depths. In the future, we can vary our model parameters to explore comparisons with potentially inaccurate human depth estimation in various tests settings.

In this work we do not directly relate the internal rotation model developed here to the rich area of mental rotation experiments. In particular, the seminal work in that area suggests a monotonically increasing relationship between rotation angle between views of an object and the human reaction time [17]. The manifold based model presented here may have a similar connection between processing time and rotation angle because the transport operators can generate transformations similar to the analog spatial representation described by humans in these studies. It may be a fruitful to examine the performance of our transformation model on mental rotation tasks and to compare to human performance on similar tasks.

3.6.2 Future Improvements

In this work we develop the framework for inferring structure from motion and learning the operators that represent rotational motion. With this baseline, we can move forward towards a more biologically plausible methodology. The inference and learning in the current model are performed using quasi-Newton optimization and gradient descent, respectively. The optimization objective is non-convex and does not naturally lend itself to a parallel representation similar to neural architectures. Moving forward, we suggest developing an optimization procedure that is more biologically plausible.

As our focus in this work is on the development of a transformation learning framework, we assume ground truth point correspondences for the learning and inference experiments in subsection 3.3.2 through subsection 3.3.4. However, identifying point correspondences from different views of the same scene is a challenging task and one that has been a focus of many computer vision algorithms [173, 105, 174, 175]. Going forward, incorporating point correspondence estimates into this framework will lead to a more versatile, biologically plausible model.

Finally, a step towards biological plausibility is extending this model to be robust to incoherent point motion and additional moving objects. As we showed in Figure 3.13c, the current inference model significantly decreases in performance with greater than 10% of points moving incoherently. An initial approach to improving the robustness to incoherent motion is to employ random sample consensus (RANSAC) [105]. With this method, transport operator coefficients could be estimated from random subsets of points in the scenes and the final transformation parameters could be chosen as those that explain the transformation between the largest number of random subsets of points.

CHAPTER 4
GENERALIZATION OF TRANSPORT OPERATOR MANIFOLD REPRESENTATION

4.1 Introduction

As we previously described[1], there are several requirements for a transformation model that can be utilized in a human-inspired classification system. Namely the manifold-based transformation model should learn a representation from the data, be capable of generating new samples and identifying relationships between samples, and be transferable between classes. By factor of its construction, the transport operator model is capable of learning generative operators that constrain motion to the manifold and inferring relationships between samples. The ability to transfer the learned transport operators between classes is the focus of our work in this chapter.

Because natural transformations are shared between classes, the manifold structure of classes can potentially be exploited for several transfer learning tasks of interest. For example, creating augmented datasets by transferring identity-preserving variations from a rich source class with dense samples to target classes with sparse training samples has the potential to significantly enhance performance in tasks such as few-shot classification and data visualization. Our work leverages the transport operator model for a novel approach to transfer learning that would not be possible using classic manifold learning methods (where the learned representation is inherently built out of embeddings of discrete training points).

[1]This chapter is in collaboration with Dr. Benjamin J. Culpepper, Huy Nguyen, and Dr. Christopher Rozell. MC analyzed the transport operator manifold representation, developed the transfer techniques, and ran experiments with the swiss roll and USPS digit data. MC learned the facial expression transport operators on facial landmark points and BC and HN trained the GAN model for generating facial image outputs. CR supervised the project. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1148903 and DGE-1650044.

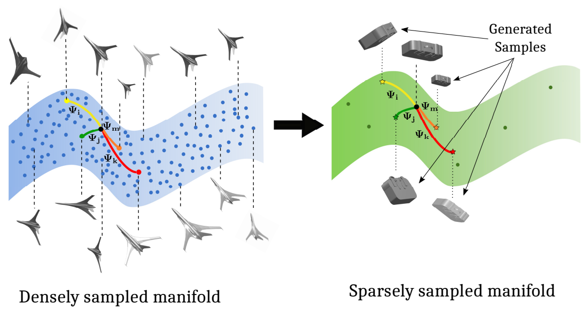


Figure 4.1: Transferring manifold paths learned from point pairs on a densely sampled manifold to a new point in sparsely sampled manifold in order to generate novel samples that are consistent with the manifold model of natural variations.

Initially we highlight that the transport operator model is a robust and accurate manifold representation of the data that can effectively embed manifold points and identify errors in other embeddings. After that we describe two transfer learning approaches. First, we describe an approach where the transfer process consists of the probabilistic exploratory mapping of the manifold around individual data points in a target class. This is akin to seeing a new object and visualizing its appearance after undergoing natural transformations. Second, we describe an approach where the transfer process consists of interpolation/extrapolation along the specific manifold paths associated with transformations between pairs of points on an object manifold. This is similar to envisioning an object evolving over an action sequence that transforms the object from a beginning state to an end state. Figure 4.1 shows a picture of transferring manifold models learned in a densely sampled source domain to a sparsely sampled target domain to generated novel samples that are consistent with the learned natural variations.

In this chapter, we describe the proposed transfer learning approaches and show that manifold transport operators [23] display consistently superior performance on transfer tasks due to a combination of capturing richer nonlinear structure and achieving decomposability of the representation (which allows more variation in the structure applied to sample generation, as well as additional semantic labeling of known transformations). We show the capability of the proposed transfer learning approach to augment data with natural transformations and to generate realistic data sequences in both stylized ground-truth simulations as well as machine learning tasks of interest on established datasets. Specifically, we demonstrate that the proposed approach can achieve superior performance on few-shot classification tasks and photorealistic generation of animated facial expressions as part of a GAN architecture.

4.2 Background

As we described in section 2.1, there are a few categories of manifold learning techniques. The first we refer to as embedding techniques because they represent the manifold through a low-dimensional embeddings of the data points [41, 42, 43, 44, 45]. Unfortunately, such approaches capture manifold structure through a transformation of the data points themselves into a lower dimensional space and there is no generative model of the data. These methods require a manifold neighborhood definition for the data points in order to estimate the low dimensional data structure, and they are sensitive to inaccuracies in the defined neighborhoods. Our second category of manifold learning techniques is tangent plane estimation techniques. These techniques represent the manifold through functions that map high-dimensional points to planes tangent to the manifold at those points [47, 48, 49, 50]. They generate new samples in the local neighborhoods around data points but the linear approximation of the neighborhood structure limits their effectiveness for representing long manifold paths. By contrast, the transport operator model possesses two characteristics required for the successful generalization of the manifold model of natural variations: 1) it can generate samples consistent with the manifold model, and 2) as we will show, it is the most effective manifold learning method for generalizing to new samples outside the source domain.

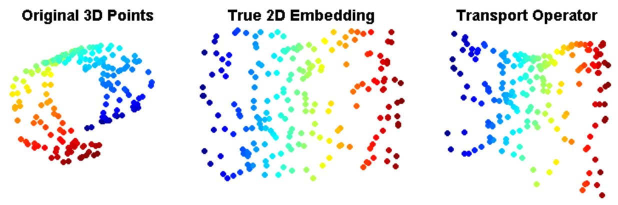


Figure 4.2: Example 2D embedding formed using the transport operator objective function as a similarity metric.

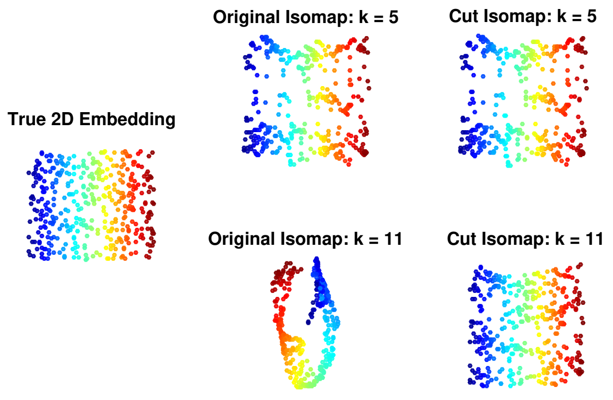
4.3 Learning Accurate Manifold Representations

The objective function in (Equation 2.3) is the negative log posterior for the transport operator model, and serves as a quantifiable metric for how well a given pair of points is represented in the model. This metric combines the approximate length of the path between the points (measured by coefficient magnitudes) and the accuracy of the approximation (measured by the MSE). With this objective function as a similarity metric between pairs of points and running classical Multidimensional scaling (MDS), we can create an embedding of the points in 2D that shows their intrinsic low-dimensional structure (Figure 4.2). As with conventional manifold learning techniques, the quality of this embedding demonstrates that the learned transport operators are accurately capturing the low-dimensional structure of the manifold data.

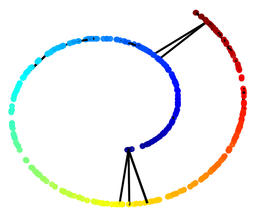
4.4 Learning Robust Manifold Representations

The manifold representation learned by the transport operators is robust. Most other manifold learning techniques require a local neighborhood definition, and even small mistakes in this neighborhood definition can lead to major errors if shortcuts are created (i.e., points nearby in Euclidean distance are considered neighbors even though they are far away on the manifold). Additionally, with many algorithms there is no way to know if a neighborhood definition has caused a shortcut error.

While the transport operator model does require a neighborhood definition to pick point



(a)



(b)

Figure 4.3: (a) Left column: True embedding of 400 swiss roll points. Middle column: Isomap embedding of swiss roll points for neighborhoods defined by k-nearest neighbors with $k = 5$ and $k = 11$. When $k = 11$, the embedding is distorted due to the large neighborhoods which include shortcuts. Right column: Embeddings after the transport operator objective function is used to identify and eliminate shortcuts. (b) View of the swiss roll points with lines between the points which had connections broken due to large objective function values that identified shortcut errors.

pairs during training, this algorithm is robust to errors in this definition because the information from each pair is averaged with many other point pairs in the learning process. Furthermore, after learning the transport operators on the dataset, the objective function value in Equation 2.3 can be used to evaluate the likelihood of points being in a neighborhood.

Figure 4.3 provides an example of how the objective function metric can be used to correct the neighborhoods for an Isomap embedding. When the Isomap embedding is computed with a neighborhood defined by the k-nearest neighbors with $k = 11$, there are shortcut connections that lead to a distorted embedding. We compute the transport operator objective function between each pair of points and “cut” the connections where this value surpasses a threshold (this threshold is determined based on outliers in a histogram of the objective function values). The corrected neighborhood definition produces a much more accurate embedding. While LL-LVM [50] uses an objective function that can identify when an entire embedding is bad due to a neighborhood error, the transport operator approach defines an objective function on each point pair that can be used to identify precisely which pairs are causing erroneous neighborhood definitions.

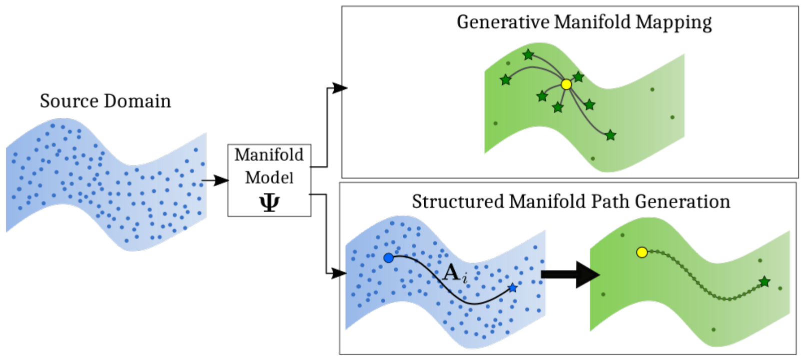

Generative Manifold Mapping
Source Domain
Manifold Model
Ψ
Structured Manifold Path Generation
$\mathbf{A}_i$

Figure 4.4: Visualization of the two transfer tasks that learn a generative manifold model from a densely sampled source domain which is used to apply transformations to points in the target domain. Generative manifold mapping applies transport operators with coefficients sampled from a probability distribution to a single point. Structured manifold path generation infers a transformation between points in the source domain that is applied to a starting point in the target domain to generate a path.

4.5 Generalization Tasks

Our transfer approach uses transport operators to represent transformations that are transferred from a source domain to a target domain, which lacks data, in order to generate new samples and enrich the target domain. We present two transfer techniques that both use a probabilistic generative model of a manifold but address different transfer learning approaches. Both of these approaches can be visualized in Figure 4.4. In the first approach, the goal is to map out a manifold around individual data points to augment a sparsely sampled dataset. We develop a method to achieve this goal, called generative manifold mapping, which generates new samples around a given data point $\mathbf{x}_0$ by applying the manifold representation with a probabilistic model on the parameters associated with each manifold motion direction. For the manifold tangent plane representations, like LSML and NLMT, these parameters define the direction of exploration on the linear tangent plane. For the transport operator representation, the parameters are coefficient values and this amounts to applying learned transport operators with coefficients sampled from a specified distribution:

$$\mathbf{x}_i = \text{expm}\left(\sum_m \mathbf{\Psi}_m w_{i,m}\right)\mathbf{x}_0 \tag{4.1}$$

$$w_{i,m} \sim U(0, a_m) \tag{4.2}$$

The coefficients are sampled from a uniform distribution for the applications in this work. However, the different distribution can be selected to match a new application.

In the second transfer approach, the goal is to generate paths associated with specific transformations. In contrast to the first approach, where the transferred transformations were defined entirely by the learned manifold model, this approach also requires a pair of points that represent the starting and ending points of the manifold path that we wish to transfer. This addresses settings in which the manifold data represents transformation sequences such as walking, speaking, facial expressions, or when manifold paths associated with labeled transformations are transferred. Our developed method for this second approach, which we call structured manifold path generation, infers a transformation between a starting point $\mathbf{x}_{i,0}$ and an ending point $\mathbf{x}_{i,1}$ that can be applied to new starting points. This is challenging for models based on manifold tangent plane representations which are not designed to generate large non-linear manifold paths. The transport operator method represents a transformation i between $\mathbf{x}_{i,0}$ and $\mathbf{x}_{i,1}$ by a set of inferred coefficients ($\mathbf{c}_i^*$) which define the operator, $\mathbf{A}_i$, that can be transferred to new points.

$$\mathbf{c}_i^* = \underset{\mathbf{c}}{\text{argmin}}\, \frac{1}{2}\left\|\mathbf{x}_{i,1} - \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m \mathbf{c}\right)\mathbf{x}_{i,0}\right\|_2^2 + \zeta\|\mathbf{c}\|_1 \tag{4.3}$$

$$\mathbf{A}_i = \sum_{m=1}^{M} \mathbf{\Psi}_m c_{i,m}^* \tag{4.4}$$

This transformed path can be applied to the original starting point, $\mathbf{x}_0$, to interpolate the path or it can be applied to a new point, $\mathbf{x}_j$, to extrapolate a new path with N steps:

$$\mathbf{x}[n] = \text{expm}(\sum_m \mathbf{\Psi}_m(\frac{n}{N}c^*_{i,m}))\mathbf{x}_j. \tag{4.5}$$

4.5.1 Generative Manifold Mapping

In this section we show how a manifold model of natural variations learned in a source domain can be used to map out unseen regions of the manifold. First we show a visualization of generative manifold mapping transfer technique on a swiss roll manifold to provide intuition before moving to a few-shot learning application with digit data. To establish a transfer learning setting with the swiss roll, we train the manifold representations on a limited portion of the swiss roll that does not represent the behavior of the manifold as a whole. Specifically, the training set does not represent the range of tangent plane orientations of the full model, and it does not extend the full width of the transverse directions present. Figure 4.5a shows the training domain in dark gray. The testing region is shown in blue, and it has no overlap with the training region. From the training domain, we are able to learn the transport operators with the motion trajectories shown in Figure 4.5b. Each plot uses a few example starting points on the swiss roll, $\mathbf{x}_i(0)$, and visualizes the effect of a single transport operator by applying a single dictionary element $\mathbf{\Psi}_m$ to each point in time steps $\mathbf{x}_i(t) = \exp(\mathbf{\Psi}_m t)\,\mathbf{x}_i(0)$, $t = 0, ..., T$. In this case, the two operators have learned to move in an expanding (shrinking) rotational motion and in a transverse motion.

Using the learned transport operators, we extrapolate to data outside the training region. Specifically, we apply the learned transport operators with coefficients sampled from a uniform distribution to data points on the edge of the training region as in (Equation 4.1). Repeating that process many times, Figure 4.5c shows how learned knowledge can be transferred to map out an approximation of the new portion of the swiss roll manifold. In this demonstration, each colored dot is the result of an extrapolation from a data point on the edge of the training region using a single randomized set of coefficients. The other

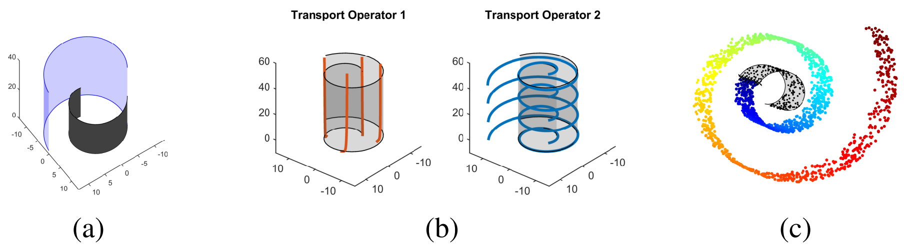
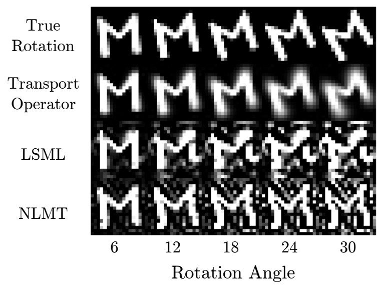


Figure 4.5: (a) The dark gray shaded area shows the training domain and the blue shaded area shows the testing domain. (b) Trajectories of the two dictionary elements, Ψ_m, trained on point pairs on a swiss roll. Each plot shows a single transport operator acting on several example points. (c) The transport operators are applied with uniformly distributed coefficients to the black 'x' points at the edge of the training domain to extrapolate the points outward.

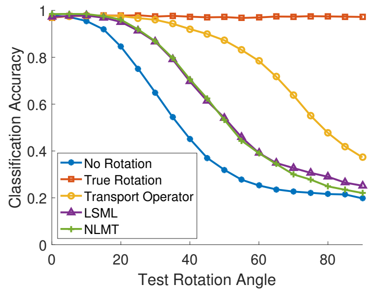
techniques that learn global manifold mappings (i.e., LSML, NLMT) can extrapolate by computing local tangent directions at each point and projecting from known points along those directions. Because of the linearity assumption inherent in tangent plane approximations, there are limits to the size of the extrapolations that are likely before departing from the manifold.

We utilize the USPS handwritten digit image dataset [176] to demonstrate the capability of generative manifold mapping for a few-shot learning classification task. To train the transport operators, we create a dataset that consists of examples of the digit '8' rotated to a randomly selected starting angle paired with that same image rotated an additional $1°$. The transport operators are learned between those rotated image pairs[2]. The neighborhood graph is defined with one nearest neighbor and all techniques are parameterized to learn a one-dimensional manifold (i.e. $M = 1$ and $\gamma = 0$). In other words, without telling the algorithms about the concept of rotation, we seek to have them learn the general transformation manifold from examples of only pairs of slightly rotated '8' digits. This task extends the transfer learning example described in [49] where they trained the Non-Local Manifold Tangent Learning mapping on all numerical digits and tested on a letter. In this case, we are training only on a single digit, providing less information for learning and increasing the chance of overfitting to the training class.

[2]The images are padded to 18×18 to allow rotation and cropping without losing digit information.



(a)



(b)

Figure 4.6: Data augmentation through transfer learning with USPS digits. (a) An example of manifold transformations trained on '8' digits being applied to the letter 'M'. The top row shows a digit rotated to specific angle using a ground truth rotation matrix. Rows 2-4 show the effect of applying each of the manifold learning techniques to the letter with the magnitude of the applied transformation increasing from left to right. (b) Convolutional neural network classifiers are tested on rotated USPS digits. The classifiers are trained on USPS digits in five variations: the original data, data trained with true rotations, and data augmented through transfer learning using transport operators, LSML, and NLMT manifold approximations based on only seeing rotated '8' digits.

To highlight the performance when information is transferred between manifolds, we apply the transformation learned on rotated '8' digits to the letter 'M'. Figure 4.6a shows the effect of applying each of the generative manifold learning techniques to an example of the letter 'M'. Each column shows an increase in the magnitude of the applied transformation. For the tangent plane estimation techniques, this corresponds to increasing the length of the path along the linear tangent plane. The transport operator approach can easily rotate the letter to 30 degrees of rotation. The tangent plane estimation techniques, LSML and NLMT, can effectively generate small rotations of about $10°$ but they are unable to generate large transformations. This highlights the benefit of using a nonlinear generative manifold learning technique for transferring large transformations.

Through generative manifold mapping, an impoverished training set from some classes can be bolstered by transferring variability learned from other classes with more abundant training data. We test this data augmentation technique on rotated USPS digits classified by a convolutional neural network classifier [177].[3] In this demonstration we use five versions

[3]We employed a LeNet convolutional network with 5×5 filters in the first and second convolutional layers.

of the classifier that are trained using different augmented datasets: 1) only the original USPS digits with no rotation introduced (naive classification), 2) the original USPS digits with ground truth rotation matrices applied (oracle classification), 3) the USPS digits transformed many times by applying the transport operator that was learned only on rotated ‘8’ digits with uniformly distributed coefficients, 4) the USPS digits transformed by the LSML-estimated tangent vectors, and 5) the USPS digits transformed by the NLMT-estimated tangent vectors. For both LSML and NLMT, the functions mapping sample points to tangent vectors were learned only on rotated ‘8’ digits.

The classification accuracy for networks trained on each of these augmented datasets is shown in Figure 4.6b. As expected, the network trained with the rotated digits (oracle classification) is not significantly affected by a change in the rotation angle of the test digit and the network trained only on the original USPS data with no rotation (naive classification) experiences a significant performance decrease with larger rotations. The network trained using the dataset augmented with transport operators improves the performance significantly over naive classification. Data augmentation with LSML and NLMT improves the performance for smaller rotation angles but the accuracy on larger angles with these augmentation techniques is significantly lower that the accuracy using the transport operator augmented network.

4.5.2 Structured Manifold Path Generation

In this section we will show how we can infer transformations between point pairs on the manifold and transfer the structured manifold paths to new starting points to generate specific transformations. Path interpolation is first demonstrated on the sphere where we can visualize how well interpolated paths follow the underlying manifold. To enable transfer learning on this manifold, we specify one half of the sphere as the source domain from which training points are selected and use the other half as the target domain. Two points are selected from the target domain and the manifold models are used to infer a path on the

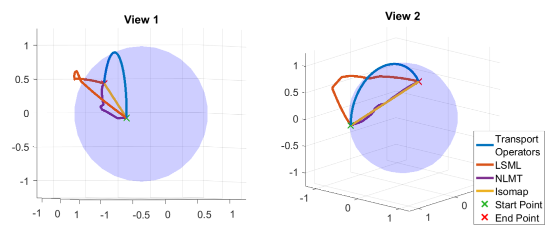


Figure 4.7: Two views of path estimates on the sphere manifold between points from the target domain in a transfer learning setting. The transport operator path follows the sphere manifold closely while the other three techniques generate paths that depart significantly from the manifold. The NLMT and Isomap paths in this figure go through the sphere rather than moving along it.

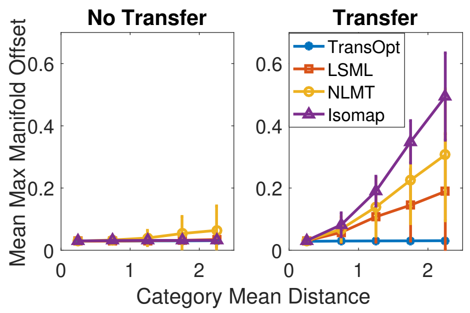


Figure 4.8: Average maximum manifold path offset for interpolated paths as a function of true path distance in both the no transfer (left) and transfer (right) scenarios. All techniques have small path offsets in the source domain. The path offsets of the paths interpolated using transport operators are still small in the transfer scenario while the paths estimated from the other three techniques deviate from the manifold.

manifold between the two points. For LSML and NLMT, the paths are generated using a snake active contour method described in [47]. Figure 4.7 shows example paths between two points selected from the target domain. In this case, the transport operator path is the only one that does not depart significantly from the manifold. Because the LSML and NLMT paths are estimated by recomputing tangents along the route to ensure the path remains on the manifold, the path offset suggests an error in the tangent mappings in this area due to the differences between the training and testing regions.

To quantify the error in the manifold representation, we compute a path offset metric that indicates how far an estimated path departs from the true manifold. This offset is computed by calculating the maximum distance between any point on the estimated path and its closest corresponding point on a densely sampled manifold. The manifold points are defined using a uniformly spaced grid over a sphere with a distance of 0.046 between neighboring points. The manifold path offset that we report is the mean value of the maximum offsets for nearly 4,000 paths in the test set. In the no transfer scenario, we evaluate each algorithm using point pairs from the source domain, and in the transfer scenario, using point pairs from the target domain. Figure 4.8 shows the average maximum manifold offset for both of these scenarios. In these figures the test cases are divided into distance categories based on the ground truth path length to analyze how the performance changes with path length. When no transfer is required, all algorithms can generate paths with small manifold offsets. In the transfer scenario, the performance degrades for all algorithms except for the transport operator approach. The ability to map out longer paths that remain close to the manifold highlights the benefit of using transport operators for defining and transferring paths that represent specific transformations.

We demonstrate the effectiveness of structured manifold path generation using facial expression sequences. In this setting, we can learn transport operators associated with expressions manifolds that can be used to apply specific expressions, represented through a structured set of transport operator coefficients, to input faces. Both the MUG facial expression database [178] and the Extended Cohn-Kanade (CK+) database [179, 180] contain image sequences from a variety of subjects making six expressions (anger, disgust, fear, happiness, sadness, and surprise). Each sequence starts with a neutral face and reaches the apex of the expression. By learning transport operators on facial landmark point sequences, we can identify natural facial dynamics that lead to expressions.

The transport operators are trained on 68 facial landmark points from pairs of images in the MUG database expression sequences, using 12 of the 52 subjects. The training does not require expression labels for the sequences – only pairs of images from the same sequence. Landmark points in each image are identified using the facial landmark detection functions in the dlib python library [181]. We sweep over the parameters in Equation 2.3 and find

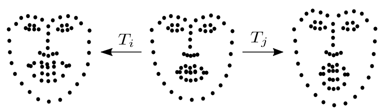


Figure 4.9: The center image shows the neutral facial landmark points. The leftmost and rightmost images are the result of applying a transport operator that qualitatively creates "happiness" and "surprise" respectively.

the best performance for $\gamma = 5 \cdot 10^{-5}$, $\zeta = 0.01$, and $M = 20$. Because the training is unsupervised, there is not a single transport operator associated with each expression. However, several operators elicit transformations that can be qualitatively associated with expressions (see Figure 4.9 for two examples).

The learned transport operators can be used to generate realistic expression sequences for new subjects by extrapolating an expression from neutral landmark points. Prior to extrapolating expressions from neutral landmark points, the coefficients need to be inferred for each desired expression as in Equation 4.3. Coefficient inference requires landmarks from an image at the apex of the expression ($\mathbf{x}_1$) and landmarks from a neutral image in the same sequence ($\mathbf{x}_0$). Once the coefficients ($\mathbf{c}_i^*$) are obtained for expression i, the resulting dynamics matrix, $\mathbf{A}_i$, can be applied to any set of neutral landmark points to create expression i. We use the generated facial landmark point paths animate faces with specified expressions using a Generative Adversarial Network (GAN) [80] conditioned on the landmark point locations.

We incorporate the landmark sequences into a progressive GAN [182] that is trained on the CelebA dataset [183] by conditioning the generator on individual attributes and the generated landmark points locations from the expression paths. See Figure 4.10 for a detailed look at the GAN architecture used. Figure 4.11 shows two examples of how transport operators are used to animate new faces. The top row shows an expression sequence from the CK+ dataset. Coefficients are inferred between the first and last image in this sequence and used to estimate the expression transformation. This expression transformation is applied to landmark points for several starting faces to generate the landmark expression sequences

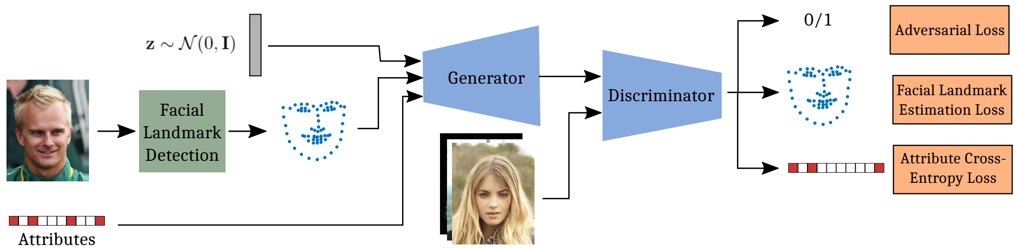

$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
Facial Landmark Detection
Attributes
Generator
Discriminator
0/1
Adversarial Loss
Facial Landmark Estimation Loss
Attribute Cross-Entropy Loss

Figure 4.10: Visualization of conditional GAN used to generate faces with expressions specified by transferred manifold paths. The generator takes a normally distributed vector ($\mathbf{z}$) as input and is conditioned on facial landmark points and attributes associated with CelebA images. The discriminator outputs a real/fake indicator and an estimate of the landmark points and attributes from the generated image.

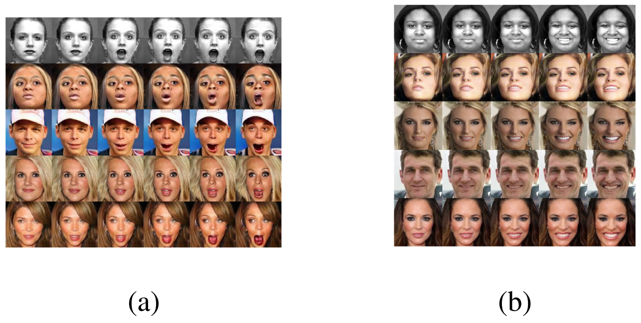


Figure 4.11: In each image, the top row shows a ground truth expression sequence from the CK+ dataset. Rows 2-4 show the faces created with landmarks from expression sequences generated through structure manifold path generation. Each row is the result of a single latent vector, $\mathbf{z}$, and single set attributes.

that the trained generator network is conditioned on in order to output the faces shown in rows 2 - 4. Each row is the result of a single latent vector, $\mathbf{z}$, and single set of attributes.

In addition to facial animation, a learned model of manifold operators can be used for few-shot learning to classify samples of labeled transformations from only a few training points. The transport operator coefficients are an efficient representation of labeled transformations that define operators which can be transferred to new starting points to augment a training set. We test this structured data augmentation by using the transport operators trained on facial landmark points from images in the MUG database to augment a small amount of labeled data from the CK+ database. Therefore, we transfer not only between individual faces but also between datasets. We assume the only labeled data for each ex-

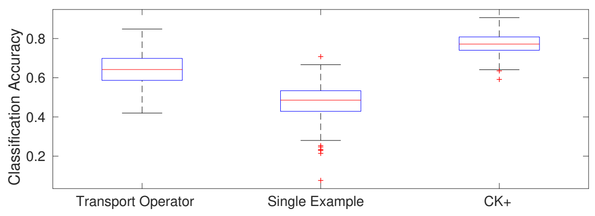


Figure 4.12: Example of few shot learning on facial expression landmark points. The boxes represent the classification accuracy distribution when the classifier is trained using the transport operator-augmented dataset (transport operator), using only one example per expression without data augmentation (single example), and using landmarks from the expression frames in all of the training sequences from the CK+ database (CK+).

pression is one neutral image and an associated expression image from the same sequence. In this one-shot learning scenario, we randomly choose one example sequence for each of the six expressions (anger, disgust, fear, happiness, sadness, and surprise) and infer the coefficients between landmarks in the first frame (neutral) and the last frame (expression apex). The transformations defined by these coefficients are then applied to the landmark points in the neutral frames from all the remaining sequences in the training set. A support vector machine (SVM) is trained with this generated expression data to classify the six expressions as well as the neutral expression.

Figure 4.12 shows the classification accuracy for 500 trials. In each trial, the data was split randomly into training, validation, and testing sets and a new set of labeled examples for each of the six expressions is chosen. The boxes in this plot extend from the 25th percentile to the 75th percentile of classification accuracy. When training with a fully labeled training set (CK+), the median accuracy was 0.7729. The low classification accuracy in this supervised scenario may be due to the difficulty of discriminating expressions given only landmark points. When training with only single examples of each expression, the median accuracy is 0.4857. The transport operator data augmentation improves the median accuracy to 0.6419. This shows how the manifold model of unlabeled, natural variations can be used to represent and transfer labeled transformations in a classification scenario with limited training samples.

4.6 Conclusion

We have shown that the transport operator manifold model is effective for transferring learned transformations between classes. We define two transfer tasks that use generative manifold models to transfer meaningful transformation information from a densely sampled source domain to a target domain with limited data, and we develop techniques to perform each of these tasks. In the first task, we use learned manifold transport operators, which represent natural variations in the data, to map out the manifold around individual data points. In the second task, we transfer structured paths associated with specific transformations by inferring the manifold transformation between two points and incrementally applying that transformation to new starting points. The transfer learning potential was shown in the context of data generation and augmentation applications by animating individuals with new facial expressions and providing examples of few-shot learning for digit and facial expression classification. In the future, these techniques can be used to learn transformations on a few densely sampled object manifolds which can be transferred to new classes that are constrained by the same natural physical processes in order to reduce the amount of training data required to obtain accurate classification for all classes.

CHAPTER 5
LEARNING NATURAL TRANSFORMATIONS IN NEURAL NETWORK LATENT SPACE

5.1 Introduction

An effective model[1] for learning manifold-based data transformations needs to be applicable to high dimensional data inputs. Standard manifold learning algorithms rely on nearest neighbor similarity metrics which can be problematic with high dimensional data in which similarity in the input space may not be most indicative of data similarity. Additionally, it is inefficient to work directly with the high-dimensional inputs when many of the input features are meaningless for understanding the true data structure. This motivates learning a low-dimensional feature space within which the manifold model can be learned. In subsection 2.2.2, we describe autoencoders, VAEs, and GANs which all learn generator functions $f : \mathcal{Z} \rightarrow \mathcal{X}$ that map points from a low-dimensional latent space, $\mathcal{Z} \subseteq \mathbb{R}^d$, to a high-dimensional data space, $\mathcal{X} \subseteq \mathbb{R}^D$. The latent space of these networks is often described as a manifold in which natural data transformations are defined by linear paths [184]. However, there are several characteristics in these embeddings that make them poor manifold representations. Namely distances in the latent space may not correspond directly to data similarity, linear paths in the latent space may depart from the manifold and result in unrealistic data points, and linear transformation representations will ultimately

[1]This chapter is in collaboration with Kion Fallah and Dr. Christopher Rozell. MC developed the manifold autoencoder model, the manifold offset distance, and performed all experiments with closed transformation paths resulting in the publication [1]. This chapter contains material from [1] with modifications. The original material is located here. This material is under copyright by Association for the Advancement of Artificial Intelligence. MC and KF contributed equally on the extension to learning unlabeled natural variations using the perceptual loss metric and the development of the coefficient encoder network and coauthored the corresponding publication [2]. CR supervised the project. Code for the closed transformation path experiments and the extensions are found here: https://github.com/siplab-gt/Manifold_Autoencoder and here https://github.com/siplab-gt/manifold-autoencoder-extended

extend to infinity and result in meaningless extended extrapolated paths. By incorporating a manifold model in an autoencoder latent space we not only improve the efficiency and the effectiveness of the manifold learning strategy but also integrate an objective that encourages the autoencoder latent representation to better match the nonlinear manifold structure of the data.

There are two general approaches that incorporate natural data structure into latent spaces of generative networks to represent more complex transformations. The first approach encourages geometric priors in the latent space that may better accommodate natural data variations [81, 82, 83, 84, 85, 86, 83, 95, 96]. The second approach defines transformations using geodesic paths in the latent space rather than linear paths [90, 91, 92, 93, 94]. We combine both of these approaches by explicitly incorporating a learned generative manifold model into the latent space of an autoencoder. This manifold model will allow us to both learn structure associated with natural variations in the data and define nonlinear paths necessary for representing paths along the manifold.

With this model, which we title the *manifold autoencoder* (MAE), we focus our analysis on two specific settings in which current methods are insufficient. The first is when the data has closed transformation paths that extend from a starting point, $\mathbf{z}_0$, and return to nearly the same point $\mathbf{z}_N \approx \mathbf{z}_0$ after N steps. There are many examples of applications that fit this setting including 3D rotating objects, temporal action sequences, and natural systems governed by underlying dynamical processes. These transformations cannot be represented by linear paths in the neural network latent space but instead require a more complex representational model for latent data variations.

Figure 5.1 provides an example of why it is necessary to address closed transformation paths directly and why we are using an autoencoder model to define our encoded latent space. The training data for these plots are 20-dimensional features that are mapped from the two-dimensional, ground truth latent space in Figure 5.1a. The points are embedded onto two circular manifolds. Figure 5.1b shows the embedding from a learned autoencoder with a two-dimensional latent space. This autoencoder effectively represents the concentric circular latent structure of the data. By contrast, the embedding from the trained VAE mixes the points from the two manifolds (Figure 5.1c). This is due to the VAE training objective which encourages the latent vectors to have a Gaussian prior distribution which aims to center all latent points around the origin. This Gaussian prior is clearly not appropriate for data with closed transformations paths. To address this manifold mismatch, the hyperspherical VAE implements a hyperspherical prior into the latent space [81]. The hyperspherical structure in the latent space introduces the possibility of creating closed paths. However, this technique does not effectively define individual object manifolds but rather combines the manifolds of all training objects onto the same hyperspherical space. Figure 5.1d shows this effect because points from the two data manifolds are embedded on the same circle in the two-dimensional latent space. The manifold autoencoder helps represent closed transformation paths by learning the nonlinear structure from the data itself and adapting the latent space of the autoencoder to fit that learned structure. We highlight the effectiveness of the MAE model on data with closed transformation paths in section 5.4.

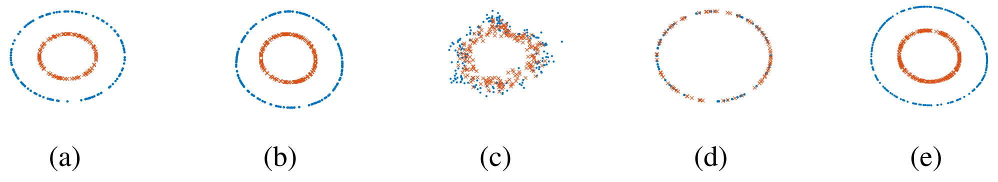


Figure 5.1: Embedding of concentric circle inputs in network latent spaces. (a) Ground truth latent embedding. (b) Circle points embedded in the autoencoder latent space. (c) Circle points embedded in the VAE latent space. (d) Circle points embedding in the hyperspherical VAE latent space (e) Circle points embedded in the manifold autoencoder latent space

The second setting that we focus on is when a dataset has no labeled transformation structure. As described in section 2.1, the transport operator training objective is computed using point pairs on the manifold. These points should be nearby on the manifold and related through transformations that the user desires to learn. For most complex datasets, information about the transformations of interest is not available. This necessitates a strategy for selecting training point pairs which can be used to learn natural transformations in complex datasets without requiring additional transformation supervision. In the absence of labeled transformations, 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. Point pairs can also be selected as nearest neighbors in the pixel space or autoencoder latent space, but many semantic transformations of interest may not be exhibited through pixel similarity nor through unsupervised autoencoder feature similarity. We introduce a method for selecting training point pairs without transformation supervision in subsection 5.3.2 and detail experiments using this method in section 5.5.

Overall, in this chapter, we introduce an approach to learn and represent natural manifold structure in a neural network, with the following specific contributions:

- We develop a model for learning generative, nonlinear transport operators between pairs of points in a network latent space that represent natural transformations in the data. We show that this model can interpolate paths between points and extrapolate paths from arbitrary starting points.

- We create a network architecture that incorporates the manifold transport operators into an autoencoder latent space. This enables us to define a latent space structure that adheres to the structure in the data itself. We show that incorporating a learned, nonlinear manifold structure in the latent space greatly improves our ability to generate extended transformations paths.

- We define a distance metric that determines the likelihood that two points, $\mathbf{z}_0$ and $\mathbf{z}_1$, lie on the same manifold given the learned manifold transport operators. We show that this distance can be used to find new samples that exist on the same transformation path as a reference point.

- We introduce a point pair selection strategy to learn a manifold representation of natural variations shared across multiple data classes without requiring transformation labels. We show that training the manifold autoencoder with this point pair selection strategy leads to learned transport operators that generate semantically meaningful transformations.

- We develop a method that uses a pretrained classifier (measuring identity-preservation of transformed samples) to learn the local regions where each transport 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 pretrained classifier.

5.2 Background

Representation learning is an area of machine learning research that involves learning representations of high-dimensional data from which it is easier to extract useful information [185]. This is often necessary because it can be difficult to discern features of interest directly from high-dimensional data inputs. Additionally, due to the curse of dimensionaity, as the dimension of the data increases, it becomes more challenging to accurately determine similarity between samples. Finally, it can be computationally taxing to work with high-dimensional inputs. Representation learning encompasses many possible areas of research (such as supervised learning, unsupervised learning, autoencoders, and manifold learning), and there are many different desired goals when performing representation learning.

One representation learning goal relevant to our manifold autoencoder work is learning representations in which similar samples are close together in the feature space. This type of representation can be useful for classifying samples based on their feature location. Contrastive representation learning techniques learn data embeddings using objectives that keep similar points close and dissimilar points farther apart. These objectives include the contrastive loss [186], the triplet loss [187, 188], and soft nearest neighbor loss [189]. Some representation learning techniques are integrated with clustering approaches in order to maintain geometric proximity between similar samples in a latent space without requiring labels [190, 191, 192]. While the techniques presented above focus on feature similarity in a Euclidean feature space, the manifold autoencoder model enables feature comparison in the latent space using geodesic distances on the manifold and the manifold offset distance described in subsection 5.3.3.

Another relevant goal of representation learning is defining features that are disentangled [193]. In the ideal scenario, disentangled features should each vary one distinct characteristic of the data. With such a feature representation, users would be able to identify naturally varying characteristics of the data, including identity-preserving characteristics that do not impact classification outputs. In a supervised learning scenario, there are many disentangling techniques that separate data style from content by encouraging similarity between the content features in samples from the same class [194, 195, 196, 197] or encouraging a subset of features to map to known class labels [198] or known transformation labels [199, 200, 201]. There are also many techniques that do not require supervision in order to disentangle latent representations. InfoGAN [202] and β-VAE [203] are two standard models used to disentangle representations. InfoGAN is a GAN-based architecture that maximizes the mutual information between a subset of the latent dimensions and the generated output. β-VAE uses the same network structure as the VAE but with a weight on the KL-divergence term in the VAE training objective that can be increased to encourage latent space disentanglement. There have been several follow-up papers to the original β-VAE network that improve reconstruction quality [204, 205] and encourage additional interpretability of the latent factors [206]. Though the disentangling of the data representation is not the prime motivation of the manifold autoencoder model, it has many characteristics that encourage disentangling of data representations including a learned dictionary of operators that represent data variations, an objective that encourages a sparse set of operators to represent a given transformation between data samples, and a training strategy

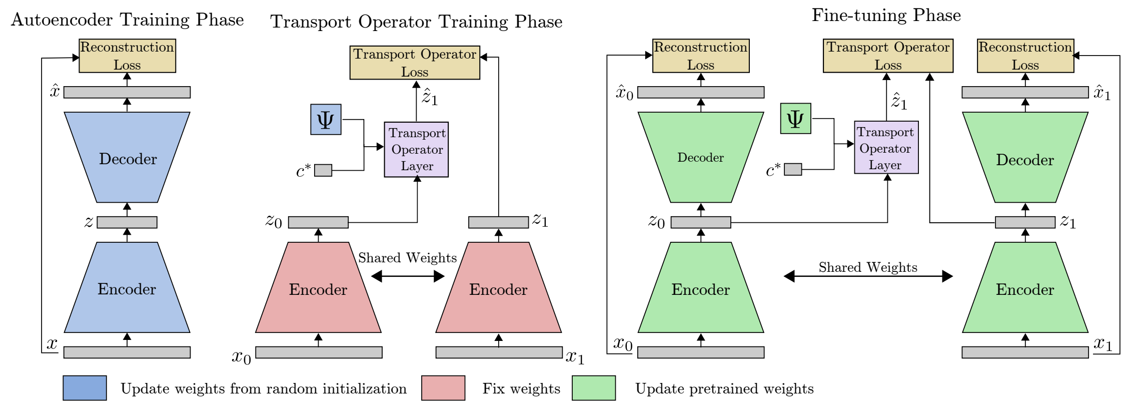


Figure 5.2: Visualization of the three phases of training the manifold autoencoder.

that estimates transformations between points that should be nearby on the data manifold which likely share the same class. Experiments in section 5.5 show qualitative results that suggest the learned operators disentangle data variations.

In this chapter we will first describe the details of the manifold autoencoder model and its training and implementation details in section 5.3. We then detail the experiments using data with closed transformation paths in section 5.4 and detail the experiments learning unlabeled natural variations in section 5.5. Finally we will describe our method for estimating the local transport operator statistics on the data manifold in section 5.6.

5.3 Manifold Autoencoder Model

5.3.1 Manifold Autoencoder Training

To incorporate the transport operators into the autoencoder network architecture, we create a new transport operator layer that applies a transformation defined by the current dictionary of M operators, $\mathbf{\Psi}_m \in \mathbb{R}^{d \times d}$, and a set of coefficients, $\mathbf{c} \in \mathbb{R}^M$, to an input latent vector, $\mathbf{z}_{in}$:

$$\mathbf{z}_{out} = \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m\right) \mathbf{z}_{in}.$$

We develop a three phase approach for training the manifold autoencoder. See Figure 5.2 for a visualization of the phases.

- **Autoencoder Training Phase**: Train the autoencoder on input data.
- **Transport Operator Training Phase**: Fix the autoencoder weights and train the transport operators on pairs of samples from the encoded latent space as described in section 2.1.
- **Fine-tuning Phase**: Fine-tune the autoencoder network weights and transport operators simultaneously.

We begin with the autoencoder training phase in which we train an autoencoder with a reconstruction loss. We then progress to the transport operator training phase in which we fix the network weights and train the transport operators as described in section 2.1 with the objective:

$$E_\Psi = \frac{1}{2} \left\| \mathbf{z}_1 - \operatorname{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m \right) \mathbf{z}_0 \right\|_2^2 + \frac{\gamma}{2} \sum_m \|\mathbf{\Psi}_m\|_F^2 + \zeta \|\mathbf{c}\|_1. \tag{5.1}$$

which is comparing the estimated manifold transformations between latent vectors $\mathbf{z}$ rather than data inputs $\mathbf{x}$ (as was described previously). The weights γ and ζ are both greater than or equal to 0. The strategy for selecting training point pairs $\mathbf{x}_0$ and $\mathbf{x}_1$ associated with latent vectors $\mathbf{z}_0$ and $\mathbf{z}_1$ is described in subsection 5.3.2.

As mentioned in section 2.1, the Frobenius norm regularizer on the dictionary elements in Equation 5.1 can aid in model order selection because it encourages the operators not being used to reduce their magnitudes to zero. This built-in selection of dictionaries allows us to initially overestimate the number of dictionaries, M, and rely on the model to identify the number necessary for representing the transformations.

During the fine-tuning phase, we simultaneously update the transport operators and the autoencoder weights. The dictionary elements are updated with the same objective function as in Equation 5.1. The network weight, ϕ, updates are supervised by both reconstruction losses and a transport operator loss:

$$E_{\phi} = \|\mathbf{x}_0 - \widehat{\mathbf{x}}_0\|_2^2 + \|\mathbf{x}_1 - \widehat{\mathbf{x}}_1\|_2^2 + \lambda E_{\Psi}, \tag{5.2}$$

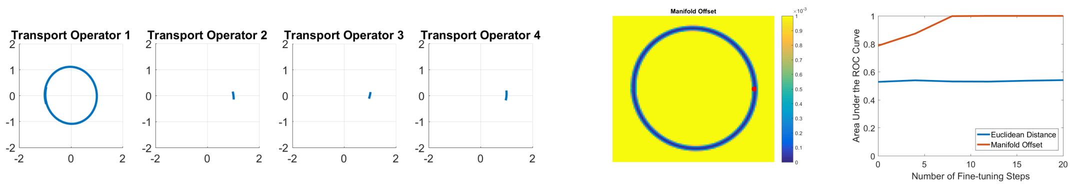
where $\widehat{\mathbf{x}}_0 = f(\mathbf{z}_0)$ and $\widehat{\mathbf{x}}_1 = f(\mathbf{z}_1)$ are the reconstructed image outputs.

As with transport operator training, fine-tuning the network weights and dictionary elements involves alternating between inferring transformation coefficients between samples and taking gradient steps on the dictionaries and network weights. Algorithm 1 shows the pseudo-code for the training procedure during the fine-tuning phase. The procedure for coefficient inference was improved by Kion Fallah and details on the optimization procedure are available in appendix section B.2 and in the revelant sections of our joint paper [2].

Algorithm 1: Fine-tuning of network weights and transport operators

```
Data: Training samples $\mathbf{x} \in \mathcal{X}$, pretrained dictionaries $\mathbf{\Psi}$, pretrained network
      weights $\phi$
Result: Fine-tuned operator dictionary elements $\mathbf{\Psi}$ and network weights $\phi$
for $i = 0, ...., N$ do
    Select a batch of point pairs $\mathbf{x}_0$ and $\mathbf{x}_1$;
    Encode point pairs to get $\mathbf{z}_0$ and $\mathbf{z}_1$;
    for $j = 0, ...., \text{num\_pairs}$ do
        Initialize $\mathbf{c}$ as $c_m \sim \text{Unif}[0, 1]$;
        Fix $\mathbf{c}$ to $\mathbf{c}^* = \text{argmin}_{\mathbf{c}} E_{\Psi}$;
        $\mathbf{\Psi} = \mathbf{\Psi} - \eta \frac{\delta E_{\Psi}}{\delta \Psi}$;
        $\phi = \phi - \zeta \frac{\delta E_{\phi}}{\delta \phi}$;
```

To visualize the transport operators learned in the latent space, we revisit the dataset used in Figure 5.1 and train an autoencoder with a two-dimensional latent space on 2D circular points that are mapped into a 20-dimensional space. Figure 5.1b shows these points encoded in the latent space after the autoencoder training phase. Point pairs for transport operator training are created by randomly selecting $\mathbf{z}_0$ from the training set and selecting a $\mathbf{z}_1$ that is one of the 20 nearest neighbors of $\mathbf{z}_0$. Figure 5.3a shows the orbits of the four transport operators after the transport operator training phase. The number of transport operators starts at $M = 4$ but the Frobenius norm term in Equation 5.1 reduces the magnitude of the unused dictionary elements to nearly 0, resulting in only the rotational operator (transport operator 1). Finally, the fine-tuning phase adjusts both the transport operators and the latent space to accommodate one another. Figure 5.1e shows the embedded points after fine-tuning. These points have a more clearly circular structure than the initial embedding points in Figure 5.1b.



(a) (b) (c)

Figure 5.3: (a) Orbits of transport operators learned in the neural network latent space. (b) Heat map of the manifold offset distance from the red reference point to every point on the grid. The manifold offset distance is small on the 1D circular manifold on which the reference point resides. (c) AUC curve for classification of points on the same or different circles during fine-tuning process. As fine-tuning progresses, the manifold offset distance becomes better able to separate samples from the inner and outer circles.

5.3.2 Training Point Pair Selection

To learn transport 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. The best selection process for these point pairs depends on the application. If the application involves temporal data, neighboring points are defined as points that are a few time steps apart. If the data does not include temporal sequences, there are several other methods for selecting point pairs during training including finding points that are close together in an embedding space, points that share similar attribute labels, points that are identified as similar through human input, and points that are close in a neural network feature space.

We generalize transport operator training to incorporate an unsupervised learning strategy that can be applied to a wide array of datasets that do not have external transformation information. In particular, we select point pairs using a perceptual loss metric [207]. 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 high-level features in pretrained classifier networks correspond to general semantic characteristics of the data and finding points that are nearby in this feature space can result in image pairs that are perceptually similar without being exactly the same [207, 208].

Figure 5.4 shows examples of nearest neighbors selected using pixel similarity, similarity in the autoencoder 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, in the set of images on the left, the perceptual loss metric identifies another bald man as a nearest neighbor which has similar characteristics of the initial image even though the exact image looks quite different.

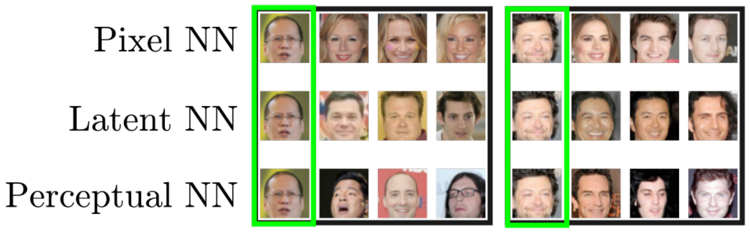


Figure 5.4: Examples of nearest neighbors identified through pixel similarity, latent similarity, and the perceptual loss metric. In each set of 12 images, the images in the first column (contained in green boxes) are the initial reference images and the images to the right of those show three selected 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 the stability of generated paths. We consider generated paths stable if the latent vectors do not expand quickly to infinity. 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 continuous time linear dynamical systems model and we can analyze their stability as dynamical systems by observing their eigenvalues [209].

In continuous time linear dynamical systems, there are three classifications of system stability. If a system is stable, the real parts of all eigenvalues are below zero. In a stable system, the magnitude of an input shrinks as time progresses forward. If a system is unstable, the real part of at least one eigenvalue is above zero. In unstable systems, the magnitude of an input expands to infinity as time progresses forward. Finally, if a system is marginally stable, then the magnitude of an input remains constant and the real parts of all of its eigenvalues are zero[2] . The transport operator model is unique because operators can represent dynamics matrices that define the natural data variations which do not have the same directionality as temporal systems (where time moves forward). Therefore transport operators can be applied with positive or negative coefficients and the traditional view of stability of temporal systems no longer applies.

In order to maintain the stability of generated paths, the operators need to generate cyclic transformation paths that neither increase nor decrease the latent vector magnitudes. This corresponds to marginal stability in linear dynamical systems. A marginally stable system has only imaginary eigenvalues with no real parts [209]. Therefore, we can identify transport operators approaching marginal stability by investigating the magnitudes of the real parts of their eigenvalues. The maximum real magnitude of an eigenvalue of a dynamics matrix drives the speed with which the generated paths expand to infinity. Therefore this maximum real magnitude is the focus of our investigation into the stability of paths generated by learned operators. 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

[2]Note that a system can also be marginally stable with at least one pair of eigenvalues with a zero real part and non-zero imaginary parts and the rest of the eigenvalues with non-positive real parts. For simplification of this analysis, we will focus on striving for systems which have eigenvalues with all real parts equal to 0.

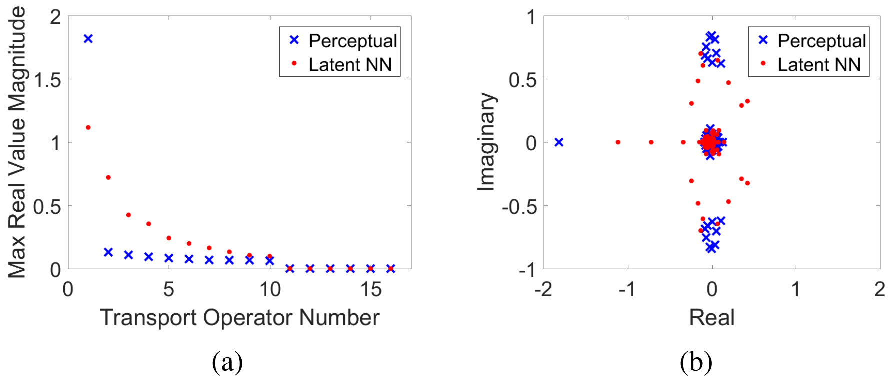
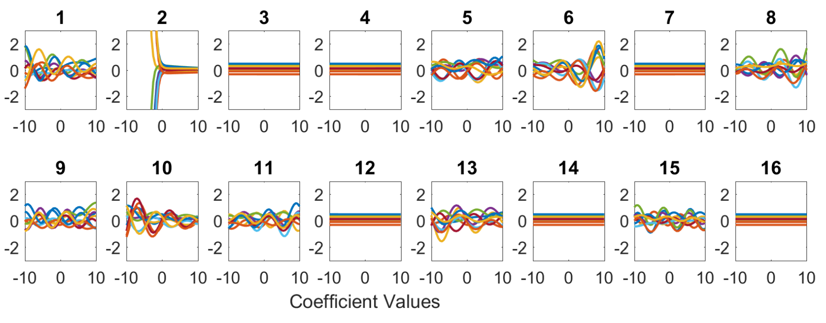


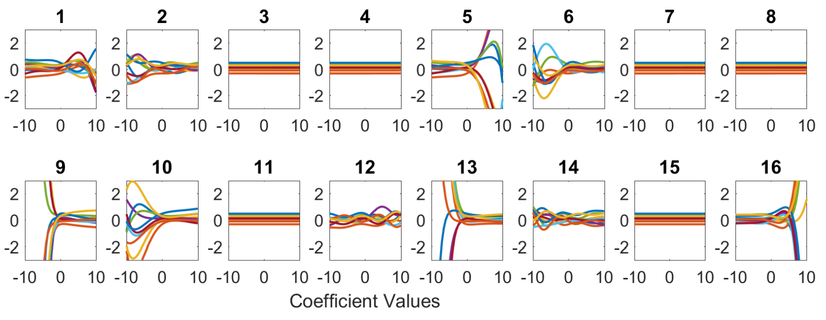
Figure 5.5: Analysis of eigenvalues of operators learned in the MNIST experiment. (a) The maximum absolute value of the real parts of eigenvalues computed from each learned operator. (b) Plot of the real and imaginary parts of the eigenvalues for each operator.

closer to marginal stability. Therefore, we quantify the stability of transport operator generated paths by looking at the maximum magnitude of real parts of eigenvalues associated with each transport operator. Figure 5.5a shows the sorted maximum magnitude of the real parts of eigenvalues in each of the 16 operators learned in the experiment in section 5.5 which learns natural MNIST variations. The blue x's are associated with the operators trained using the perceptual point pair selection strategy and the red dots are associated with the operators trained using a simple strategy of selecting point pairs as nearest neighbors in the latent space. With the exception of 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 learned operators more intuitively, we plot the values of the latent dimensions as individual operators are applied in Figure 5.6. Each line in these plots shows the influence of the operators on a single latent dimension (Note that this experiment utilizes a latent dimension of 10). Figure 5.6a shows the paths generated by transport operators trained with the perceptual point pair selection strategy . The plots with straight lines (i.e., operators 3, 4, 7, 12, 14, and 16) indicate that those operators had their magnitudes reduced to zero during training and therefore they have no impact on the latent vectors when applied. 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 Figure 5.5a). By contrast, several of the operators trained with point pairs selected as nearest neighbors in the latent space extend to infinity (Figure 5.6b). This can explain the larger maximum real value magnitudes in Figure 5.5a. This analysis leads us to conclude that the perceptual point pair selection strategy has a greater potential to yield identity-preserving transport operators.



(a)



(b)

Figure 5.6: 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.

5.3.3 Manifold Offset Distance

The learned transport operators define motion on class manifolds and constrain possible directions of transformations. We can use the operators to define a distance that indicates the likelihood that two latent vectors lie on the same manifold path. We call this distance the *manifold offset distance* and it indicates the how well a point $\mathbf{z}_1$ can be estimated from a starting point $\mathbf{z}_0$ using the learned operators $\mathbf{\Psi}$. The manifold offset distance is defined as:

$$d_{\text{offset}} = \left\| \mathbf{z}_1 - \operatorname{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m^* \right) \mathbf{z}_0 \right\|_2^2, \tag{5.3}$$

with the inferred coefficients $\mathbf{c}^* = \operatorname{argmin}_{\mathbf{c}} E_{\Psi}$.

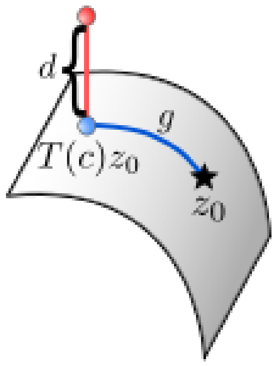


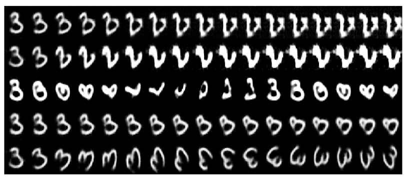
Figure 5.7: Stylized view of the manifold offset distance (labeled here as d). The manifold offset distance is a measure of the distance between the transformed point on the manifold using inferred coefficients $\widehat{z}_1$ (blue dot) and the true value of z_1 (red dot)

We use the concentric circle data set and the transport operators shown in Figure 5.3a to intuitively understand the usefulness of the manifold offset distance. Figure 5.3b shows a heat map of the manifold offset distances from the red reference point. The manifold offset is very small for the points that exist on the same 1D circular manifold as the reference point and much larger for points off of this circular manifold.

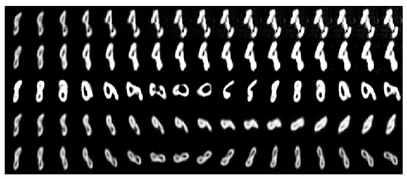
We also use the manifold offset distance to perform a simple classification task to showcase the usefulness of the fine-tuning phase of training. The classification task is to determine whether two points are on the same circle in the concentric circle data set or different circles (see Figure 5.1a). To perform this classification, we compute the manifold offset distances between two points and use a binary classifier on the distances to determine whether two points are from the same circle or different circles. We quantify the performance using the area under a receiver operator characteristic (ROC) curve. The area under the curve (AUC) is a measure of class separability that spans from 0 to 1 with 1 indicating perfect separability. Figure 5.3c shows the evolution in the AUC during the fine-tuning phase. We compare the AUC using the manifold offset distance and the Euclidean distance. This plot shows that the classification is no better than chance when using the Euclidean distance between points. By contrast, there is high AUC using the manifold offset distance and an increase in AUC as fine-tuning progresses. This provides evidence that the fine-tuning is structuring the latent space so the transport operators more accurately fit the encoded data.

5.4 Closed Transformation Paths Experiments

In this section, we analyze the performance of our transport operator model on two datasets that contain closed transformation paths: rotated MNIST digits and gait sequences from the CMU Graphics Lab Motion Capture Database.[3] We demonstrate the usefulness of the manifold autoencoder for transformation extrapolation and manifold identification.



(a)



(b)

Figure 5.8: Two examples of extrapolated rotations in the autoencoder latent space. Row 1: Extrapolated linear path in latent space prior to fine-tuning. Row 2: Extrapolated linear path in latent space after fine-tuning. Row 3: Extrapolated hyperspherical VAE path Row 4: Extrapolated transport operator path prior to fine-tuning. Row 5: Extrapolated transport operator path after fine-tuning.

[3]CMU Graphics Lab Motion Capture Database found here: http://mocap.cs.cmu.edu/

5.4.1 Rotated MNIST Digits

We train our autoencoder and transport operators using a subset of 50,000 images from the MNIST training set [177]. In the autoencoder training phase, we train on batches of rotated MNIST digits. During the transport operator training phase, we generate pairs of rotated training digits. The first image in the pair is a digit image rotated to an angle, θ, between 0 and 350° degrees and the second image is that same digit image rotated to $\theta + 6^\circ$. The training results in one operator with a much higher magnitude than the others (see Figure B.1b for transport operator magnitudes after transport operator training). Therefore, during the fine-tuning phase, we only use the single high-magnitude operator in the transport operator layer. We compare our technique against a hyperspherical VAE trained on rotated digit data. This network was trained using the network architecture and the loss functions in the code provided by the authors [81]. In order to define paths in the hyperspherical VAE latent space, we compute geodesic paths on a hypersphere [210]. See the appendix sections section B.3 and subsection B.11.1 for more details of the network architecture and training process.

Figure 5.8 shows a visualization of extrapolated paths in the latent space. These paths are defined by estimating the transformations between latent representations of pairs of digits with 1° of rotation between them and extrapolating the transformation to estimate the full rotation path. To generate the visualizations, we i) encode a starting image into the latent space using the encoder function $g(\cdot)$: $\mathbf{z}_0 = g(\mathbf{x}_0)$; ii) apply the transformation to the the starting point; and iii) decode the image output: $\widehat{\mathbf{x}}_t = f(\mathbf{z}_t)$. This figure shows that the linear paths produce a small amount of rotation but quickly transform the appearance of the digit itself. The path in the hyperspherical VAE latent space also induces a small rotation, but as the path extends, the appearance of the original digit transforms. This is an example of how the single hyperspherical latent space is not effective for representing large, identity-preserving transformations.

The transport operator before fine-tuning extrapolates rotation to about 45° before the

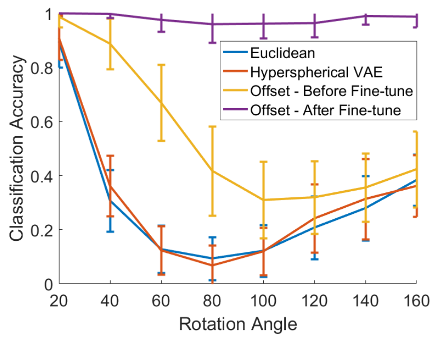
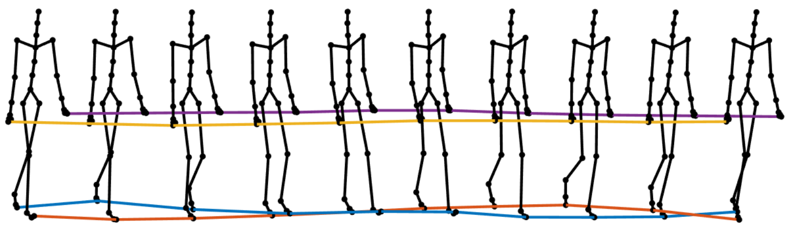


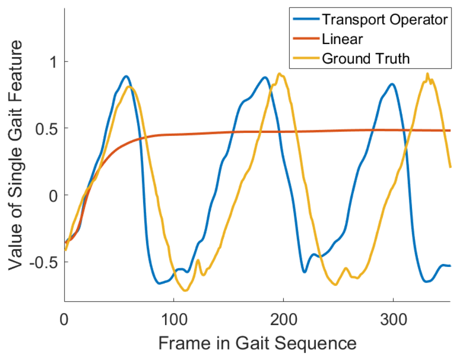
Figure 5.9: Average nearest neighbor classification accuracy of rotated test digits with standard deviation error bars.

digit appearance significantly changes. The transport operators resulting from the fine-tuning phase can generate full, 360-degree rotations with only small changes to the digit appearance. This makes it clear that the fine-tuning phase is needed to adjust the network weights and transport operators to represent the full closed transformation paths.

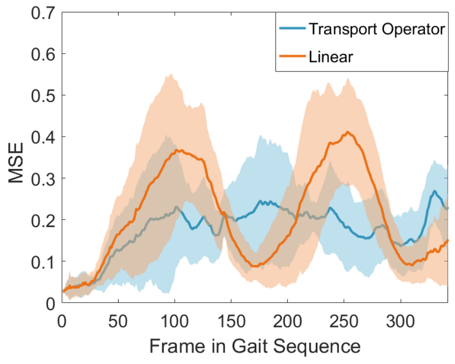
The manifold offset distance can be used to identify samples on the same manifold transformation path. The rotated paths of individual images represent 1D manifolds in this setting and our task is to identify points on the same rotated digit manifolds using the manifold offset distance. This is shown through nearest-neighbor classification in the latent space. We define 10 training samples, the latent representation of one image for each digit class at zero degrees of rotation. The testing samples are latent representations of each of the training samples rotated to different angles. We classify each rotated sample using the class label of the nearest neighbor in the training set. Figure 5.9 shows the nearest neighbor classification accuracy at different rotation angles using the Euclidean distance, the geodesic distance in the hyperspherical VAE latent space, and manifold offset distance before and after fine-tuning for 50 trials. This shows that the manifold offset distance prior to fine-tuning, the Euclidean distance, and the geodesic distance in the hyperspherical VAE are poor measures for nearest neighbor classification at large rotation angles. However, after fine-tuning, the manifold offset distance is very effective for identifying points on the same rotational manifold.



(a)



(b)



(c)

Figure 5.10: Analysis of gait sequence paths generated with learned transport operators. (a) The effect of applying a learned operator which generates a full gait sequence. (b) Example of one feature from the data space as it progresses during an extrapolated gait sequence. (c) The error in between the estimated gait sequences and the ground truth gait sequences. The estimated sequences are extrapolated from a starting point in the latent space.

These experiments highlight key benefits of our method which encourages nonlinear manifold structure in the autoencoder latent space. First, we show how the learned transport operators can be used to extrapolate full closed transformation paths from arbitrary starting points. Second, we demonstrate the advantage of incorporating the manifold model into the training of the network itself to fine-tune the encoder mapping. Finally, we show the manifold offset distance can be used to identify pairs of points on the same transformation manifold.

5.4.2 Gait Sequences

The CMU Graphics Lab Motion Capture Database includes human walking sequences that were recorded by a motion capture system. The motion data is represented through 62-dimensional feature vectors that specify locations and orientations of joints. We use the preprocessing procedure discussed in [211] which converts the 62-dimensional features into 50-dimensional features. We train on walking sequences 1-16 from subject 35. Walking data from this subject is abundant and this subject has been widely used for gait analysis [91, 211, 212].

We train an autoencoder with a five-dimensional latent space using features from the gait sequences. The point pairs used to train the transport operators are composed of features from two frames that are separated by a five frame interval. We begin with 10 transport operator dictionary elements but we eliminate all but three operators due to their low magnitudes after the transport operator training phase (see Figure B.1c for transport operator magnitudes after transport operator training). Each of these three operators induces a continuous gait sequence with some difference between the mechanics of the movement. See the appendix section B.3 for more details on the training process and an analysis of how the three highest magnitude operators are jointly used.

Figure 5.10a shows an example gait sequence generated using the learned operator with the highest magnitude after the transport operator training phase. This operator can represent the full loop of a gait transformation. To quantify our ability to estimate ground truth gait paths, we begin at a latent vector that encodes a neutral standing pose and use both a learned manifold transport operator and a linear path to estimate the ground truth gait sequence. We estimate the transformation between latent representations of features that are six frames apart in the gait sequence. We then use the estimated transformations to extrapolate full gait sequence paths in the latent space. We test gait estimates on test sequences 30 -34. Figure 5.10b shows the extrapolated paths of a single feature in a gait sequence that is decoded from the extrapolated latent paths. We see that the linear path can effectively estimate the ground truth gait sequence in the initial frames but it eventually plateaus at a meaningless value. In comparison, the transport operator extrapolated path can match the ground truth for several gait sequences. Figure 5.10c shows the mean squared error between the ground truth gait sequence and the estimated gait sequences for five test walking sequences. The error in the transport operator paths is largely due a mismatch in the speed of the extrapolated and the ground truth gait sequences.

5.5 Unlabeled Natural Variations Experiments

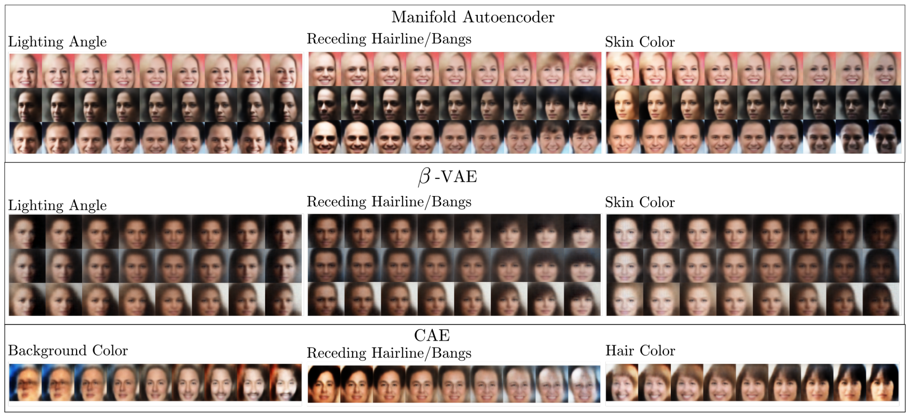
In these experiments, we examine the ability of the MAE to learn natural transformations from complex datasets in which the underlying identity-preserving transformations are not easily identifiable using our perceptual point pair selection strategy. We work with three datasets: MNIST [177], Fashion-MNIST [**xiao2017online**], and CelebA [183]. 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 [203, 202, 213, 214], CelebA contains semantically meaningful natural transformations that may be amenable to qualitative labeling.

In all experiments, we follow the three-phase training procedure as described in subsection 5.3.1. We select training point pairs that are nearest neighbors in the feature space of the final, pre-logit layer of a ResNet-18 [215] classifier pretrained on ImageNet [216]. Additional details on the datasets, network architectures, and training procedure are available in the section B.4.

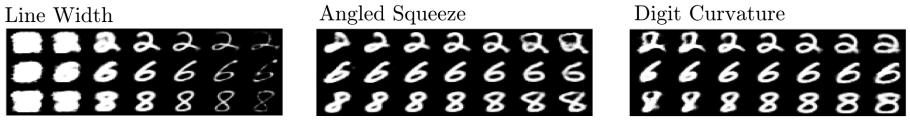
We compare against the contractive autoencoder (CAE) [87] and β-VAE [203], two other autoencoder based methods that incorporate data structure into the latent space. The CAE represents another technique (discussed in more detail in subsection 2.2.2) that learns a manifold representation in a neural network 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.

First we show how well our MAE model is able to learn natural data variations in a 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. Figure 5.11 shows the paths generated by transport operators trained in this model. Each block of images corresponds to the same operator or variational factor being applied to multiple inputs images. 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, though relying on network trained on ImageNet, is effective over a range of datasets.

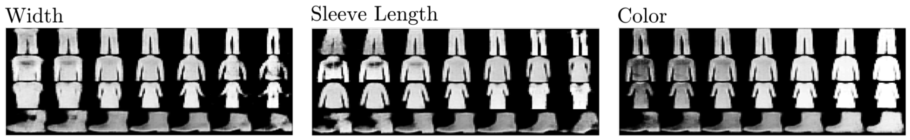
While many of the operators can be assigned a semantic label through qualitative visual inspection, the CelebA dataset has attribute labels for the images which enable a quantitative analysis of the connection between learned transport operators and dataset attributes. To classify attributes, we fine-tune a ResNet-18 pretrained on ImageNet with 16 classification heads for attributes including smile, beard, hair color, and pale skin [217]. With this classifier, we are able to identify transport operators that correspond to specific dataset attributes. Figure 5.12 shows the classification outputs of the attribute classifier for example transport operators. Our model learns operators that vary hair color, skin paleness, smiling, beard, sunglasses, and bangs.



(a)



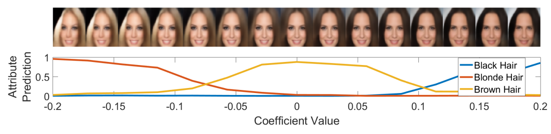
(b)



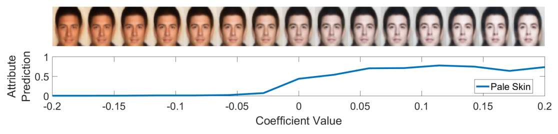
(c)

Figure 5.11: Paths generated by applying a subset of learned transport operators on three datasets. Each block of images corresponds to the same operator or variational factor being applied to multiple inputs images. 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.

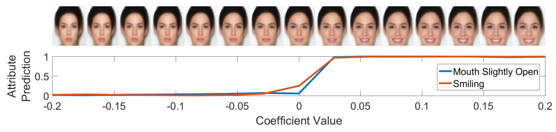
In Figure 5.11 we compare the transport operator paths to paths generated by the CAE and the β-VAE. The CAE-generated paths represent the directions of motion on the tangent planes estimated by a SVD at individual points so each transformation is specific to the input point. The β-VAE paths are generated 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 sacrific-



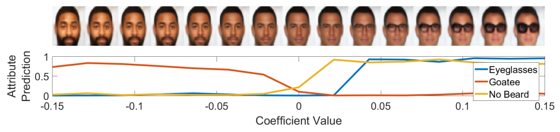
(a)



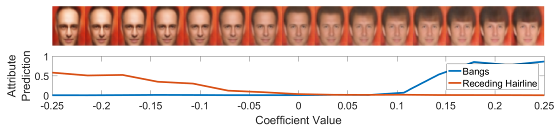
(b)



(c)



(d)



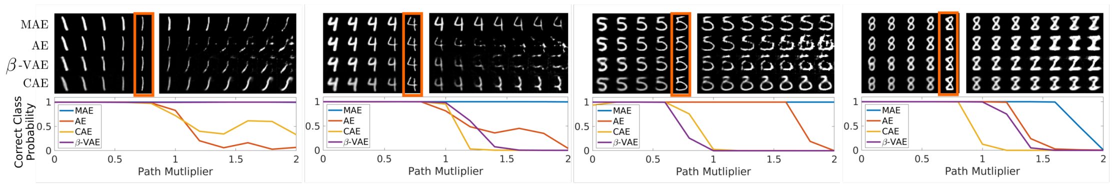
(e)

Figure 5.12: Paths generated by the application of 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) beard/sunglasses (e) bangs

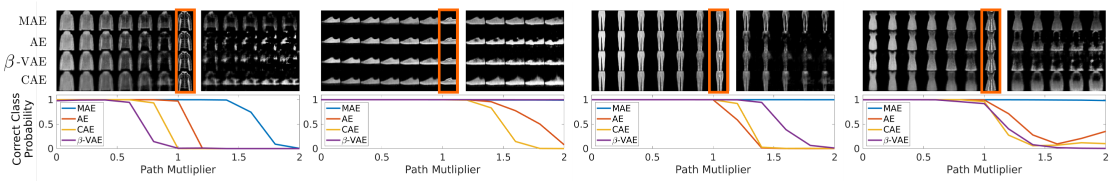
ing reconstruction performance. In the appendix section B.4, we include examples of the transformations generated by each learned transport operator.

In Figure 5.13, to highlight the ability of the transport operators to estimate nonlinear manifold paths between points in the latent space, 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 latent space 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 along the path 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: $\mathbf{z}_t = \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m^* t\right)\mathbf{z}_0$. 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 final column in that block showing 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. For instance, if the input class for a given example is '1' then the plot shows the probability output for class '1'. All four methods perform interpolation effectively but our trained model estimates the extrapolated paths more accurately. Figure 5.14 shows how the classification accuracy varies during extrapolation sequences for 4000 samples. This shows that 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.



(a)



(b)

Figure 5.13: 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 $\mathbf{x}_1$ 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

Figure 5.15 shows an interesting feature in the transport operators learned on CelebA images – many of them generate cyclic paths that begin and end at nearly the same point.

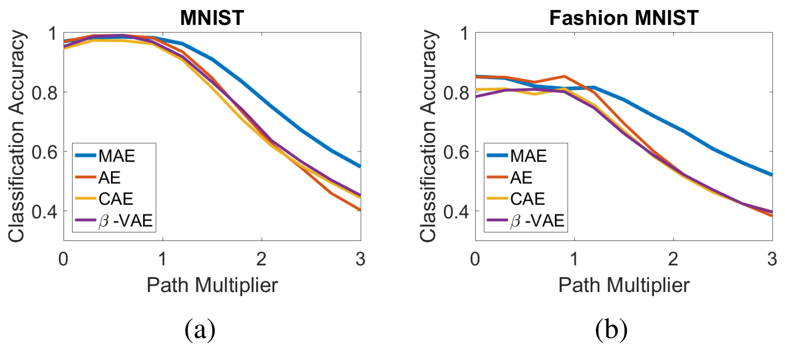
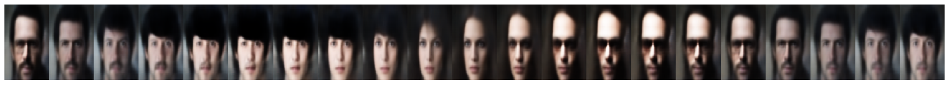


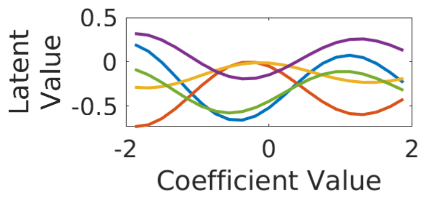
Figure 5.14: The average accuracy of the classifier output for images at each point along the interpolation/extrapolation sequence. When the path multiplier is between 0 and 1 that indicates interpolation and path multipliers beyond 1 indicate extrapolation. (a) MNIST (b) Fashion-MNIST

Also, in these cyclic sequences, the transformations seem to lead to a change in gender. The image sequence in Figure 5.15a 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. 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. Figure 5.15b 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 section 5.4 we 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.



(a)



(b)

Figure 5.15: 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.

5.6 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:

$$p_\zeta(\mathbf{c}) = \prod_{m=1}^{M} \frac{1}{2\zeta} \exp\left(-\frac{|c_m|}{\zeta}\right). \tag{5.4}$$

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. 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. 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:

$$\widehat{\mathbf{z}} = T_{\Psi}(\mathbf{c})\mathbf{z} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_{\epsilon}^2 I), \tag{5.5}$$

where $T_{\Psi}(\mathbf{c}) = \operatorname{expm}\left(\sum_{m=1}^{M} \Psi_m c_m\right)$.

Given labeled observations $(\mathbf{z}, y)$ and a pretrained classifier $r(\cdot)$, we aim to learn a network $q_{\phi}(\mathbf{c}|\mathbf{z})$, which we call a coefficient encoder, 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 [218]. In our context, we have 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-divegernce between the classifier output $r(\mathbf{z})$ and the estimated probability of the transformed output $r(\widehat{\mathbf{z}})$.

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 Equation 5.4. Our final objective for training the coefficient encoder network is:

$$E = D_{KL}(r(\mathbf{z}) \| r(\widehat{\mathbf{z}})) + D_{KL}\left(q_\phi(\mathbf{c}|\mathbf{z}) \| p_\zeta(\mathbf{c})\right). \tag{5.6}$$

A more detailed derivation can be found in appendix section B.10

This coefficient encoder introduces 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.

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 fix the transport operators and network weights and then train the coefficient encoder network 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 [41] of latent vectors and color each point by the encoded scale parame-

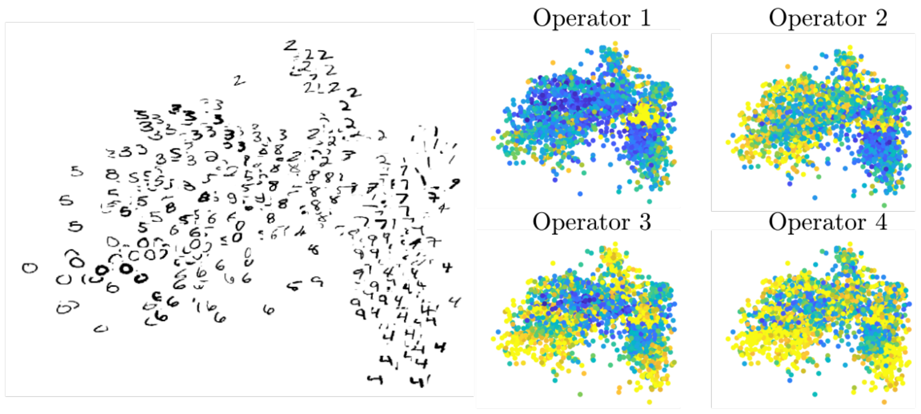


Figure 5.16: Visualizations of the encoded coefficient scale weights for the MNIST dataset. The leftmost image shows an Isomap embedding of the latent vectors with input images overlaid. The scatter plots on the right show the same Isomap embedding colored by the encoded coefficient scale weights for 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.

ter for coefficients associated with each of the transport operators. Figure 5.16 shows these embeddings for MNIST data and Figure 5.17 shows these embeddings for Fashion-MNIST. Each operator has regions of the latent space where their use is concentrated (indicated by the yellow coloring).

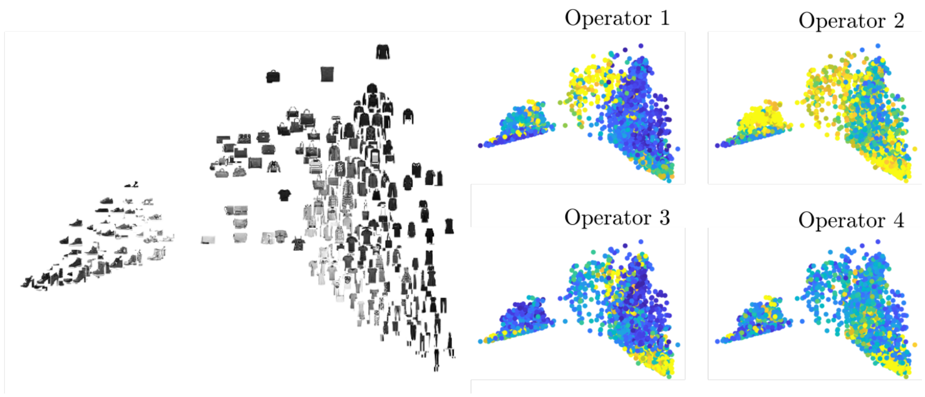


Figure 5.17: Visualizations of the encoded coefficient scale weights for the Fashion-MNIST dataset. The leftmost image shows an Isomap embedding of the latent vectors with input images overlaid. The scatter plots on the right show the same Isomap embedding colored by the encoded coefficient scale weights 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.

In data augmentation the goal is to create new samples by applying identity-preserving

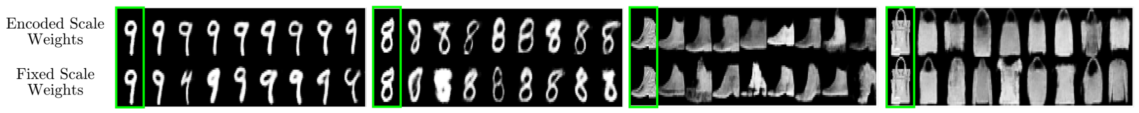
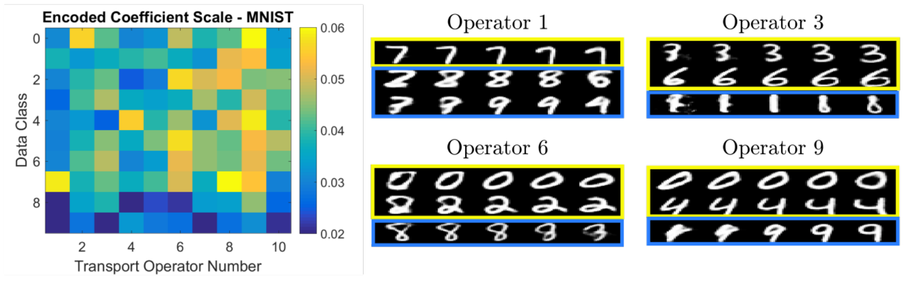


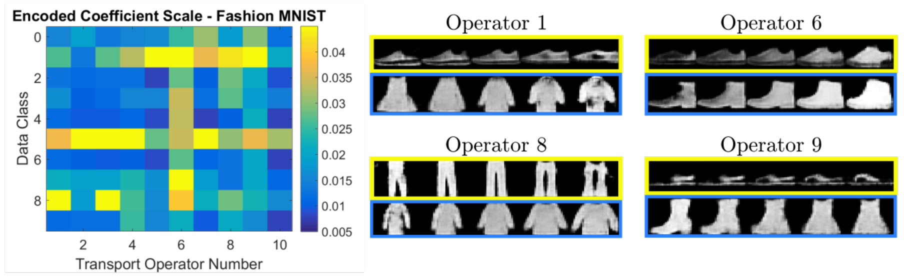
Figure 5.18: Examples of samples generated by transport operators using coefficients sampled with encoded scale weights (top row) and with a fixed scale weight (bottom row). Images in the green box are the input images and the remaining images in each row are transformed outputs.

variations. This task can be aided by the coefficient encoder which identifies scale weights for the local coefficient distributions that best preserve the identify of transformed outputs. To highlight this benefit, in Figure 5.18 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 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.

With the coefficient encoder, we can now investigate the patterns of scale weights across the operators and classes. This analysis can be used to inform us about the types of manifold transformations the classifier is invariant to. The colored plots in Figure 5.19a and Figure 5.19b show the average coefficient scale weights for each class (rows) and each transport operator (columns) for MNIST and Fashion-MNIST. From this we can identify classes for which the classifier is both robust and sensitive to transformations, represented by large and small scale weights respectively. For instance, the Fashion-MNIST classifier seems to be robust to transformations in the trouser and sandal classes (classes 1 and 5) because they have large scale weights for a majority of operators. On the other hand, the MNIST classifier seems to be sensitive to transformations in the '9' class because it has



(a)



(b)

Figure 5.19: 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 bottom blue boxes). The examples with low coefficient scale weights change classes more easily than other examples. (a) MNIST (b) Fashion-MNIST

small scale weights for all operators. The sensitivity to transformations in ‘9‘s is intuitive because they can fairly easily transform into ‘4‘s and even ‘1‘s. We can also examine which classes share the use of the same transformations. The images on the right in Figure 5.19a and Figure 5.19b 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 Figure 5.19a 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.

5.7 Summary

In this chapter we develop the manifold autoencoder which incorporates learnable manifold transport operators into the latent space of an autoencoder. This allows us to learn a model of the data manifold and adapt the structure of the autoencoder latent space to fit that manifold model. The MAE enables the interpolation and extrapolation of paths in the latent space, the computation of manifold distances between points, and the generation of realistic transformed outputs. We perform experiments on data with closed transformation paths, including rotated MNIST digits and human gait sequences, to highlight the value of having a built-in method for defining nonlinear transformation paths. We show how the fine-tuning of the autoencoder enables accurate extended extrapolated paths and how the manifold offset distance can be used to classify samples that are on the same manifold. We learn transport operator representations of natural, unlabeled transformations in MNIST, Fashion-MNIST, and CelebA datasets using the perceptual point pair selection strategy during training and show that these operators generate semantically meaningful transformations. Finally, we create the coefficient encoder which uses a pretrained classifier to learn local regions where operators are likely to be used while preserving the identity of transformed samples. Overall, the work in this chapter establishes a foundational representation of transformations that can be used in a human-inspired classification system. We have shown the potential of this model to learn complex, unlabeled transformations on a variety of datasets, perform identity-preserving data augmentation with the learned operators, and achieve distance based classification on a learned manifold.

CHAPTER 6
VARIATIONAL AUTOENCODER WITH LEARNED LATENT STRUCTURE

6.1 Introduction

Natural data[1] is distributed with an unknown probability distribution. A goal in accurate data generation is to be able to sample directly from the data distribution. However, for most datasets, the data distribution is complex and difficult to parameterize. Generative models have been introduced with the aim of approximating sampling from natural data distributions. In particular, the variational autoencoder (VAE) model is derived with the objective of maximizing the marginal likelihood of the data $p(\mathbf{X})$. To do this, a variational inference algorithm is introduced that learns a generator from a low-dimensional latent vectors to the high-dimensional data outputs. This enables sampling from the latent space with a specified prior distribution in order to generate new, realistic data samples.

There are several aspects of the traditional VAE framework that prevent it from faithfully representing complex natural variations on manifolds associated with separate data classes. First, VAEs enforce a global structure in the latent space through the use of a prior distribution, and that prior may not match the true data manifold; this model mismatch can result in a less accurate generative model of the data. Second, natural paths, which interpolate or extrapolate natural data variations, are poorly defined in the latent space of traditional VAEs. In many cases, the data transformations are defined using linear paths in a Euclidean latent space [184]. These simple paths can diverge from the true data manifold, leading to interpolated points that result in unrealistic decoded image outputs. Finally,

[1]This chapter is in collaboration with Dr. Gregory Canal and Dr. Christopher Rozell. MC and GC jointly developed the probablistic framework for the VAELLS model and MC performed the experiments and was the lead author on the associated publication [3]. CR supervised the project. This chapter contains material from [3], with some modifications. The original material is located here. This work is copyrighted with the Creative Commons Attribution 4.0 International License. Code is available here: https://github.com/siplab-gt/VAELLS

traditional VAEs encourage points from all data classes to cluster around the origin in the latent space [30]. Without adequate class separation, traversing the latent space can easily result in a change of class, making it difficult to learn identity-preserving transformations and subsequently use them to understand within-class relationships.

In this chapter, we incorporate the transport operator model into the latent space of a VAE which enables the model to learn the manifold structure from the data and use that structure to define a latent space prior that is a better match to the true data manifold. The resulting model, which we call the Variational Autoencoder with Learned Latent Structure (VAELLS), allows us to sample points from an estimated manifold.

Overall in this chapter we have the following specific contributions:

- We adapt the VAE probabilistic model by integrating transport operators which learn the latent structure from the data.
- We introduce anchor points which are points in the data space that correspond to samples on the desired manifold. This enables the manifold structure to be defined by a combination of anchor points and learned transport operators and alleviates the need for selecting point pairs during transport operator training.

This model ensures the prior is well matched to the data and allows us to define nonlinear transformation paths in the latent space and describe class manifolds with transformations stemming from examples of each class. The VAELLS model shares a similar motivation to the manifold autoencoder introduced in the previous chapter but it incorporates the manifold model in the context of a full probabilistic generative framework.

6.2 Background

The VAE model learns a low-dimensional latent representation by defining a generator function $g : \mathcal{Z} \rightarrow \mathcal{X}$ that maps latent points $\mathbf{z} \in \mathbb{R}^d$ to high-dimensional data points $\mathbf{x} \in \mathbb{R}^D$. The desired objective for training a VAE is maximizing the log-likelihood of a dataset $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ given by $\frac{1}{N}\log p(\mathbf{X}) = \frac{1}{N}\sum_{i=1}^{N}\log\int p(\mathbf{x}_i, \mathbf{z})d\mathbf{z}$. However, this objective is difficult to maximize, especially when parameterized by a neural network. To address this complication, VAEs instead maximize the Evidence Lower Bound (ELBO) of the marginal likelihood of each datapoint $\mathbf{x}_i$:

$$\log p(\mathbf{x}_i) \geq \mathcal{L}(\mathbf{x}_i) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}_i)}[-\log q_\phi(\mathbf{z} \mid \mathbf{x}_i) + \log p_\theta(\mathbf{x}_i, \mathbf{z})], \tag{6.1}$$

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

Kingma and Welling [30] developed an efficient method to approximate the ELBO by introducing the *reparameterization trick* that enables the stochastic latent variable $\mathbf{z}$ to be represented by a deterministic function $\mathbf{z} = h_\phi(\mathbf{x}, \varepsilon)$, where ε is an auxiliary random variable with a parameter-free distribution. In the traditional VAE framework, the variational posterior is selected to be a multivariate Gaussian distribution, meaning that $\mathbf{z}$ is reparameterized around the encoded point $\mathbf{z} = f_\phi(\mathbf{x}) + \sigma\varepsilon$ where $\varepsilon \sim \mathcal{N}(0, I)$. Additionally, the prior $p_\theta(\mathbf{z})$ is modeled as a zero-mean isotropic normal distribution that encourages the clustering of latent points around the origin.

6.3 VAELLS model

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

We start with the expanded ELBO from [30]:

$$\mathcal{L}(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})\right]. \tag{6.2}$$

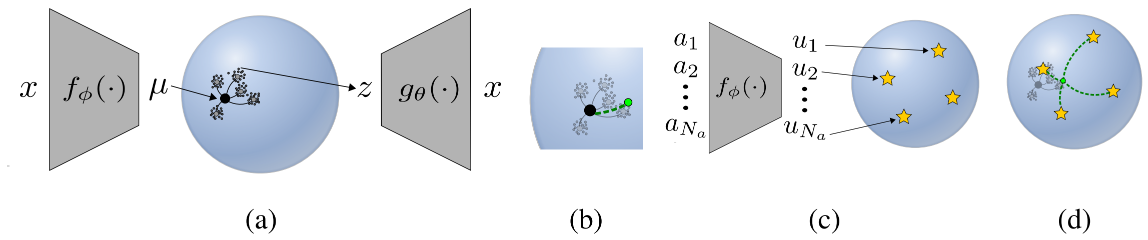


Figure 6.1: Visualizations of the VAELLS model: (a) Posterior sampling process using transport operator-generated paths and Gaussian noise. (b) Transformation path (green dotted line) inferred between $f_\phi(\mathbf{x})$ (black dot) and $\mathbf{z}$ (green dot) when computing the posterior. (c) Encoding of anchor points into the latent space. (d) Transformation paths (green dotted lines) inferred between $f_\phi(\mathbf{a_i})$ (yellow stars) and $\mathbf{z}$ (green dot) when computing the prior.

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

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

To approximate Equation 6.2 with sampling, first let $L_{\mathbf{x}}(\mathbf{z}) \equiv \log p_\theta(\mathbf{x} \mid \mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})$ and note that by marginalizing over transport operator coefficients $\mathbf{c}$ we have $\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}\left[L_{\mathbf{x}}(\mathbf{z})\right] = \mathbb{E}_{\mathbf{z},\mathbf{c} \sim q_\phi(\mathbf{z},\mathbf{c}|\mathbf{x})}\left[L_{\mathbf{x}}(\mathbf{z})\right]$ which allows us to estimate Equation 6.2 by sampling from $q_\phi(\mathbf{z}, \mathbf{c} \mid \mathbf{x})$. We draw a sample from $q_\phi(\mathbf{z}, \mathbf{c} \mid \mathbf{x})$ in two steps: first, as defined in the generative model in Equation 2.2, we sample a set of coefficients $\widehat{\mathbf{c}}$ from a factorized Laplace distribution $q(\mathbf{c})$, and then sample $\mathbf{z}$ from $q_\phi(\mathbf{z} \mid \widehat{\mathbf{c}}, \mathbf{x})$. Both of these sampling steps can be achieved with deterministic mappings on parameter-free random variates, allowing for the use of the reparameterization trick. Specifically,

$$\begin{aligned} \widehat{\mathbf{c}} &= l(\mathbf{u}; b) \quad \mathbf{u} \sim \text{ Unif}\left(-\frac{1}{2}, \frac{1}{2}\right)^M \\ \mathbf{z} &= \mathbf{T}_{\mathbf{\Psi}}(\widehat{\mathbf{c}}) f_\phi(\mathbf{x}) + \gamma\varepsilon \quad \varepsilon \sim \mathcal{N}(0, I), \end{aligned} \tag{6.3}$$

where $l(\mathbf{u}; b)$ is a mapping described in Appendix section C.1 with Laplace scale parameter b and $\mathbf{T}_{\mathbf{\Psi}}(\mathbf{c}) = \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m \mathbf{c}_m\right)$.

Figure 6.1a shows this sampling process where the data $\mathbf{x}$ is encoded to a mean location $\mu = f_\phi(\mathbf{x})$. The latent vector $\mathbf{z}$ is the result of transforming μ by $\mathbf{T}_{\mathbf{\Psi}}(\widehat{\mathbf{c}})$, which moves the vector on random paths along the manifold, and adding Gaussian noise specified by γ.

The resulting transport operator variational posterior follows as

$$\begin{aligned} q(\mathbf{c}) &= \left(\frac{1}{2b}\right)^M \prod_{m=1}^{M} \exp(-\frac{|c_m|}{b}) \\ q_\phi(\mathbf{z} \mid \mathbf{c}, \mathbf{x}) &\sim \mathcal{N}\left(\mathbf{T}_{\mathbf{\Psi}}(\mathbf{c}) f_\phi(\mathbf{x}), \gamma^2 I\right) \\ q_\phi(\mathbf{z} \mid \mathbf{x}) &= \int_{\mathbf{c}} q_\phi(\mathbf{z} \mid \mathbf{c}, \mathbf{x}) q(\mathbf{c}) d\mathbf{c} \qquad (6.4) \\ &\approx \max_{\mathbf{c}} q_\phi(\mathbf{z} \mid \mathbf{c}, \mathbf{x}) q(\mathbf{c}), \qquad (6.5) \end{aligned}$$

where the approximation in Equation 6.5 is motivated by the fact that the sparsity-inducing Laplace prior on $\mathbf{c}$ typically results in joint distributions with $\mathbf{z}$ that are tightly peaked in coefficient space, as described in [219] (see appendix section C.1 for details). This approximation is widely used in sparse dictionary learning for computational efficiency. The inferred coefficients $\mathbf{c}^*$ that maximize $q_\phi(\mathbf{z} \mid \mathbf{c}, \mathbf{x}) q(\mathbf{c})$ define the estimated transformation between the encoded latent coordinates and the sampled point. The dotted green line in Figure 6.1b shows a visualization of the inferred transformation path between $f_\phi(\mathbf{x})$ (the black dot) and $\mathbf{z}$ (the green dot).

Our next key contribution lies in the construction of a prior distribution learned directly from the underlying data manifold using transport operators. To gain intuition about this prior, imagine a set of N_a *anchor* points in the data space that correspond to samples on the desired manifold. The data manifold structure is represented by a combination of these anchor points, encoded in the latent space, and the learned transport operators that can extrapolate the manifold structure in the latent neighborhood of each of them. Figure 6.1c shows a set of anchor points encoded into the latent space which represent the scaffold off of which the manifold structure is built. The prior is defined by the same probabilistic manifold model used in the variational posterior, but starting at each anchor point $\mathbf{a_i} \in \mathbb{R}^D$ rather than $\mathbf{x}$. This prior requires that paths be inferred between $\mathbf{z}$ and each encoded anchor point $\mathbf{u}_i = f_\phi(\mathbf{a_i})$, visualized in Figure 6.1d as green dotted lines. The overall prior density for $\mathbf{z}$ is then defined as

$$p_\theta(\mathbf{z}) = \frac{1}{N_a} \sum_{i=1}^{N_a} q_\phi(\mathbf{z} \mid \mathbf{a_i}). \tag{6.6}$$

The introduction of anchor points into the definition of the prior provides a unique opportunity for the user to define the classes within which they want to learn natural, identity-preserving transformations. If desired, the user can choose a set of anchor points, independent of class labels, to define the full data manifold and learn transformations throughout the entire dataset. If the user wants to learn identity-preserving transformations on specific class manifolds, they can separate the anchor points into classes and define the prior with respect to class-specific anchor points. Even when the prior is defined with respect to anchor points in each class individually, the set of learned transport operators is shared over all classes. Finally, if the user desires to learn a manifold representing specific transformations of individual samples, they can select anchor points that are sampled from transformation paths of specific example data points. In practice, the anchor points are initialized by either uniformly sampling the manifold space or through manual selection by a practitioner (e.g., by selecting several anchors per data class) and they can be updated throughout training.

In Tomczak & Welling [220], a prior is adopted that is similarly composed of a sum of variational posterior terms, and is motivated as an approximation to an optimal prior that maximizes the standard ELBO. However, our motivation for the prior in Equation 6.6 as a direct sampling of the data manifold is novel, and our prior more directly aligns with the data manifold because it is constructed from operators that traverse the manifold itself.

The final addition to the VAELLS objective is a Frobenius-norm regularizer on the dictionary magnitudes as used in the original transport operator objective [23]. This prevents the magnitudes of the dictionaries from increasing without bound and helps identify how many transport operators are necessary to represent the manifold by reducing the magnitude of operators that are not used to generate paths between samples. We define the loss function to be minimized as the approximate negative ELBO in Equation 6.2, restated here for convenience:

$$\begin{aligned} L_{\mathbf{x}}(\mathbf{z}) &\equiv \log p_\theta(\mathbf{x} \mid \mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \\ \mathcal{L}_{VAELLS}(\mathbf{x}) &= -\mathbb{E}_{\mathbf{u},\varepsilon}\left[L_{\mathbf{x}}(\mathbf{T}_{\mathbf{\Psi}}(l(\mathbf{u};b))f_\phi(\mathbf{x}) + \gamma\varepsilon)\right] \\ &\qquad\qquad\qquad + \frac{\eta}{2}\sum_{m=1}^{M}\|\mathbf{\Psi}_m\|_F^2, \end{aligned} \tag{6.7}$$

We optimize this loss simultaneously over encoder-decoder networks f_ϕ and g_θ, anchor points $\{\mathbf{a_i}\}_{1:N_a}$, and transport operators $\mathbf{\Psi}$. The implementation details of our ELBO and its optimization are described in more detail in appendix section C.1.

6.4 Related Work

There are currently many adaptations of the original VAE that handle a subset of the limitations addressed by VAELLS, such as learning the prior from the data, defining continuous paths in the latent space, and separating the individual class manifolds. Table 6.1 provides a comparison of techniques which we describe in detail below.

Traditionally, the latent space prior is defined as a Gaussian distribution for model simplicity [30]. However this prior encourages all points to cluster around the origin which may not occur in the natural data manifold. A mismatch between the latent prior distribution of a VAE and the data manifold structure can lead to over regularization and poor data representation. Other models incorporate more complex latent structures such as hyperspheres [81], tori [82, 83], and hyperboloids [84, 85, 86]. These models have demonstrated their suitability for certain datasets by choosing a prior that is the best match out of a predefined set of candidates.

However, these methods are only capable of modeling a limited number of structured priors and are not able to adapt the prior to match the data manifold itself. This is a serious drawback since, in most practical cases, the latent structure of data is unlikely to easily fit a predefined prior. The variational diffusion autoencoder (VDAE) and the $\mathcal{R}$-VAE define the latent space prior directly from the data using Brownian motion on a Riemannian manifold [221, 96]. The VampPrior is defined using a sum of variational posterior distributions with learnable hyperparameters [220]. As noted in discussion around Equation 6.6, this definition of a prior incorporating variational posterior terms is the same as what is used in VAELLS. However, the form of the variational posteriors differ between VampPrior and VAELLS with VAELLS using the structured manifold model in Equation 6.4. This affords the VAELLS model the additional benefits noted below.

In addition to differences in how latent space structure is represented, there are various approaches for defining natural paths in this space. In the simplest case, paths in VAEs with a Euclidean latent spaces are modeled as linear paths. Some methods define geodesic transformation paths in the latent space. This can be done by incorporating a structured latent prior, like a hypersphere, on which geodesic paths are natural to compute [81]. Interpolated geodesic paths can also be estimated in a Euclidean latent space using an estimated Riemannian metric [96, 90, 91, 92]. However, these methods are limited to defining extrapolated paths by random walks in the latent space rather than structured paths. Finally, there are other methods that lack straightforward definitions for how to compute continuous paths from one point to another [220, 82].

Another limitation of most VAE models is that they do not encourage class separation and therefore generated paths often do not represent natural identity-preserving transforma-

Table 6.1: Comparison of VAE Techniques

Model	Adaptive Prior	Defines Paths	Class Separation
VAE [30]	No	Linear	No
Hyperspherical VAE [81]	No	Geodesic	No
ΔVAE [82]	No	No	No
VDAE [221]	**Yes**	Linear	No
VAE with VampPrior [220]	**Yes**	No	No
$\mathcal{R}$-VAE [96]	**Yes**	**Nonlinear**	No
Lie VAE [83]	No	**Nonlinear**	No
VAELLS (our approach)	**Yes**	**Nonlinear**	**Yes**

tions on separate class data manifolds. This makes it difficult to understand the within-class relationships in the data. Some techniques encourage class separation through the choice of a prior that does not encourage data clustering [81] but they do not explicitly define separate class manifolds. By defining the prior structure with respect to anchor points on specified data manifolds, VAELLS has the flexibility to define which manifolds it wants to learn transformations on. This results in a latent space structure where transformations correspond to identity-preserving variations in the data.

One model that has notable similarities to ours are the Lie VAE [83]. This model also uses Lie group representations of transformations in the latent space. The Lie VAE model encodes the data into latent variables that are elements in a Lie group which represent transformations of a reference object. This model requires the type of Lie group transformations that the network will represent (e.g., $SO(3)$) to be specified prior to training which may result in a model mismatch.

6.5 Experiments

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

The unique characteristics of the VAELLS variational posterior and prior lead to specific training considerations. First, as shown in Equation 6.5, to compute both the variational posterior and prior distributions we must infer transformation coefficients that maximize $q_\phi(\mathbf{z} \mid \mathbf{c}, \mathbf{x})q(\mathbf{c})$ and $q_\phi(\mathbf{z} \mid \mathbf{c}, \mathbf{a_i})q(\mathbf{c})$ respectively. This involves coefficient inference between each sampled point $\mathbf{z}$ and its neural network encoding $f_\phi(\mathbf{x})$ as well as between $\mathbf{z}$ and all encoded anchor points $f_\phi(\mathbf{a_i})$. The coefficient inference is performed using a conjugate gradient descent optimization solver.

For training the networks weights we use the Adam algorithm [222]. To add stability and improve efficiency of training, we alternate between steps where we update the network and anchor points while keeping the transport operators fixed, and steps where we update the transport operators while keeping the network weights and anchor points fixed.

The selection of anchor points is important for learning identity-preserving transport operators. Training VAELLS on a dataset with multiple classes requires class labels for both the anchor points and training points in order to compare each training sample with only anchor points from the same class. Anchor points are initialized by selecting training samples from each class in the input space. While we do allow for updates to the anchor points, in practice they only vary by a negligible amount during the entire training process. Details of the network architectures and training parameters for each experiment as well as an algorithmic view of the training procedure (algorithm 2) are available in the Appendix C.

6.5.1 Swiss Roll

We begin by applying VAELLS to a dataset composed of 20-dimensional vector inputs that are mapped from a 2D ground truth latent space with a swiss roll structure (Figure 6.2a). We selected this classic manifold test structure because many VAE techniques that incorporate specific structured priors into the latent space have not demonstrated the ability to adapt to this specific geometry. The latent space is two-dimensional and the VAELLS prior uses four anchor points that are spread out along the swiss roll (shown as black x's in Figure 6.3b). Because this experiment involves only one class, the same four anchor points are used for each training sample. In the swiss roll test we learn a single operator.

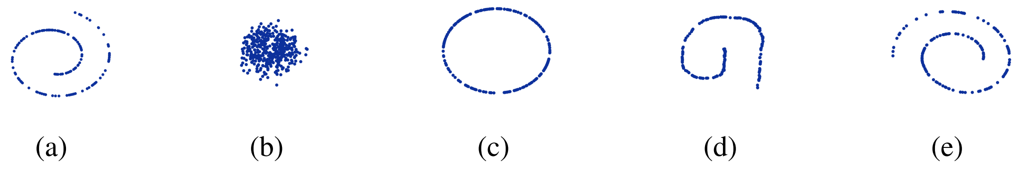


Figure 6.2: Embedding of swiss roll inputs in VAE latent spaces. (a) Ground truth latent structure. (b) VAE. (c) Hyperspherical VAE. (d) VAE with VampPrior. (e) VAELLS.

Figure 6.2 shows the latent space embedding for several VAE techniques. The traditional VAE with a Gaussian prior in the latent space loses the latent structure of the true data manifold because it encourages all points to cluster around the origin (Figure 6.2b). The hyperspherical VAE similarly loses the true data structure because it distributes the latent points on a hypersphere (Figure 6.2c). The VAE with VampPrior is able to estimate the spiraling characteristic of the swiss roll structure (Figure 6.2d), but it is not a smooth representation of the true data manifold. By contrast, the encoded points in VAELLS (Figure 6.2e) clearly adapt to the swiss roll structure of the data.

We also utilize the swiss roll dataset to provide an intuitive understanding of how the prior in our method is formed as a combination of the learned transport operators and the encoded anchor points. Figure 6.3a contains the encoded latent points overlaid with the orbit of the operator learned by VAELLS. Specifically, the colored line shows how the transport operator trajectory for our learned operator evolves over time when applied to a single starting point. This trajectory can be generated for any dictionary element $\mathbf{\Psi}_m$: $\mathbf{z}_t = \text{expm}(\mathbf{\Psi}_m \frac{t}{T})\mathbf{z}_0$, $t = 0, ..., T$. Figure 6.3b contains latent points sampled from the prior using the sampling detailed in Equation 6.3 with increased b and γ parameters to

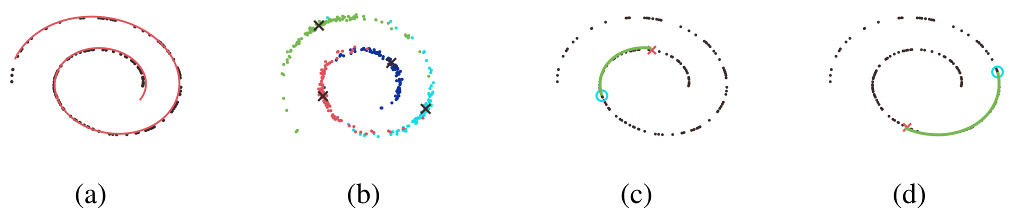


Figure 6.3: Visualization of the components of the VAELLS prior on the swiss roll dataset: (a) The orbit of the transport operator learned on the swiss roll dataset plotted on top of encoded latent points. (b) Sampling of the latent space prior from each of the anchor points (labeled with black x's); each color indicates a separate anchor point of origin. (c-d) Transport operator paths inferred between pairs of points on the swiss roll manifold in the latent space. The cyan circle is the path starting point and the red x is the desired path ending point

make the sampling easier to visualize. This shows how the prior has been well-adapted to the swiss roll structure. Finally, we demonstrate how transport operators can be used to define nonlinear paths in the latent space. To generate paths between pairs of points with our learned operators, we first infer the coefficients c^* between each pair. We then interpolate the path from the starting point $\mathbf{z}_0$ as follows: $\mathbf{z}_t = \text{expm}\left(\sum_{m=1}^{M} \mathbf{\Psi}_m c_m^* t\right) \mathbf{z}_0$. Figure 6.3(c-d) show two example inferred paths between points encoded on the swiss roll manifold. This experiment highlights three beneficial characteristics of VAELLS - learning a specific latent space manifold structure, sampling points from that manifold, and generating paths directly on the manifold surface.

6.5.2 Concentric Circle

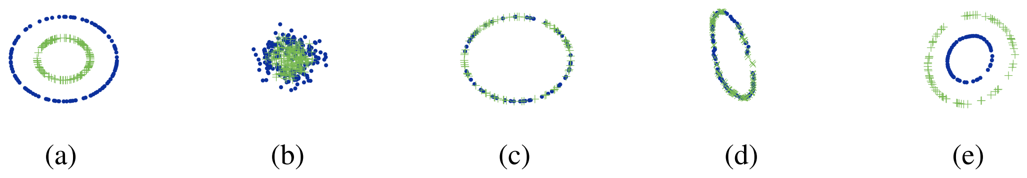


Figure 6.4: Embedding of concentric circle inputs in VAE latent spaces. (a) Ground truth latent structure. (b) VAE. (c) Hyperspherical VAE. (d) VAE with VampPrior. (e) VAELLS.

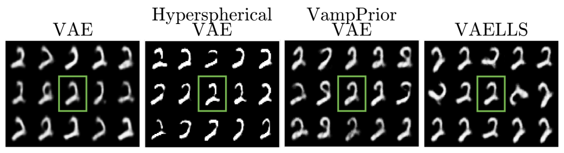
Next we apply VAELLS to a dataset composed of 20-dimensional data points that are mapped from a 2D ground truth latent space with two concentric circles (Figure 6.4a). As in the previous example, our network maps these inputs into a two-dimensional latent space. This experiment has two classes (inner circle and outer circle) so we select three anchor points per concentric circle with the anchor points evenly spaced on each circle. During training, the training points from each circle are compared against only those anchor points on the same circle manifold. In the concentric circle test we learn four operators.

This dataset in particular is well-suited for assessing how well each method is able to discriminate between the two concentric circle manifolds once points are mapped into the latent space. Figure 6.4 shows the encoded latent points for several different VAE approaches. All three comparison techniques (Figure 6.4(b-d)) lose the class separation between the ground truth concentric circle manifolds. Additionally, as in Figure 6.2, the Gaussian prior of the traditional VAE distorts the true data structure (Figure 6.4b), and the VAE with VampPrior encodes a latent structure with similar characteristics to the ground truth manifold but fails to model the exact shape (Figure 6.4d). By contrast, the encoded points in the VAELLS latent space maintain the class separation while simultaneously encoding the true circular structure. This verifies two characteristics of our approach that improve upon the traditional VAE model – learning the prior from the data and class separation.

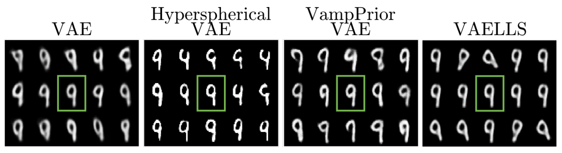
6.5.3 Rotated MNIST Digits

The rotated MNIST dataset [177] is a natural choice for demonstrating VAELLS because it consists of real images in which we have an intuitive understanding of what the rotational transformations should look like. To define the rotated digit manifold, we specify anchor points as rotated versions of training inputs and aim to learn a transport operator that induces a latent space transformation corresponding to digit rotation. In practice this means that for each training sample we select several rotated versions of that digit as anchor points.

First, to highlight that we can learn a transformation model that is adapted to the ro-



(a)

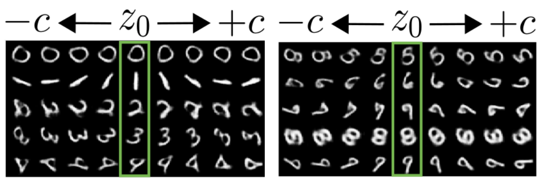


(b)

Figure 6.5: Examples of images decoded from latent vectors sampled from the posterior of models trained on rotated MNIST digits. In each example, the center digit (in the green box) is the decoded version of the input digit and the surrounding digits are images decoded from the sampled latent vectors. Sampling in the VAELLS latent space results in rotations in the sampled outputs.

tated digit manifold, we show the result of generating data from input points in the test set using the sampling procedure described in Equation 6.3. Figure 6.5 shows the decoded outputs of latent vectors sampled from the posterior for two example test points using four different VAE models. In each example, the center image (enclosed in a green box) is the decoded version of the input test sample. The images surrounding each center are decoded outputs of latent posterior samples. In order to visualize noticeable sampling variations, we increase the standard deviation and scale of the sampling noise in each of these models. The key result is that the VAELLS sampling procedure using the learned transport operator leads to latent space transformations that correspond to rotations in decoded outputs. This verifies that the transport operator corresponds to movement on a learned rotated digit manifold, unlike comparison techniques which only capture natural digit variations and not specifically rotation.

In Figure 6.6 we show how we can extrapolate rotational paths in the latent space using our learned transport operator. To generate this figure, we randomly select example MNIST digits with zero degrees of rotation and then encode those digits to get each starting point $\mathbf{z}_0$. The decoded versions of these initial points are shown in the middle columns (enclosed by a green box) in each figure. We then apply the learned operator with both positive and negative coefficients to $\mathbf{z}_0$ and decode the outputs. The images to the left of center show the path generated with negative coefficients and the images to the right of center show the path generated with positive coefficients. This shows how we can generate rotated paths using the learned transport operator. It also highlights the ability for VAELLS to define identity-preserving transformations with respect to selected input points. In these examples, the class identity of the digit is qualitatively preserved for about 180 degrees of rotation.


$-c \longleftarrow z_0 \longrightarrow +c$
$-c \longleftarrow z_0 \longrightarrow +c$

(a) (b)

Figure 6.6: Extrapolated rotation paths in the VAELLS latent space. The center digit in each row is a decoded version of an input digit and the digits to the left and right of center show decoded outputs from latent paths extrapolated using the learned transport operator with negative and positive coefficients respectively.

6.5.4 Natural MNIST Digits

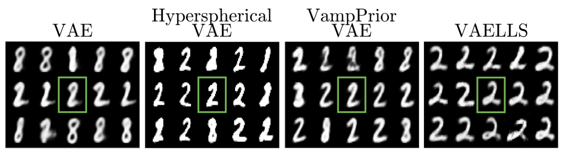
In our final experiment, we highlight our ability to learn the natural manifold structure in MNIST digits. The anchor points are initialized by randomly selecting training examples from each digit class without any special consideration given to the specific image selection. Without a priori knowledge of the manifold structure, we have flexibility in how to parameterize our model; two parameters of specific interest are the number of transport operator dictionary elements M used to define latent space transformations and the number of anchor points per class N_a. These parameters impact the two components of the prior definition: the learned transport operator model and the anchor points. Table 6.2 shows how varying these parameters impacts the quantitative performance of VAELLS as measured by estimated log-likelihood (LL) and mean-squared error (MSE) between input and reconstructed images. Our estimated log-likelihood is based off of Bruda et al. [223] and the details of its computation with our model are given in appendix section C.4. The number of dictionary elements M has a large impact on the estimated log-likelihood. As the number of dictionary elements increases, the value of $-\log\left[q_\phi(\mathbf{z} \mid c, \mathbf{x})\right]$ decreases significantly which increases the prior term in our log-likelihood computation. This indicates

Table 6.2: Evaluation Metrics of VAELLS Trained on MNIST

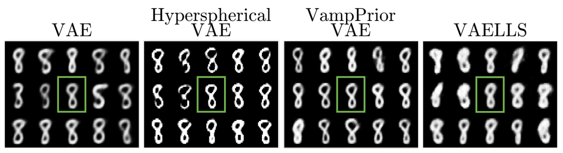
M	N_a	LL	MSE
2	8	-4.026×10^6	0.0315
4	8	-1.169×10^6	0.0232
8	8	-731.91	0.0244
4	12	-9.248×10^5	0.0221
4	16	-7.523×10^5	0.0232

that more dictionary elements enable the model to more accurately estimate paths between sampled latent points and the encoded anchor point locations. The number of anchor points does not have a clear impact on the log-likelihood and the MSE stays fairly constant as we vary the M and N_a parameters. For the rest of the results we use $M = 8$, $N_a = 8$.

Figure 6.7 shows the result of sampling the variational posterior in a similar manner to Figure 6.5. Note that sampling in the VAELLS latent space leads to natural digit transformations in the decoded outputs that maintain the class of the original digit. By contrast, the sampling in the latent space of the comparison techniques can lead to changes in class. Appendix section C.9 contains paths generated by each of the learned transport operators that highlight how they represent natural transformation paths. This experiment demonstrates the strengths of VAELLS in cases where the data manifold is unknown. By training on points associated with anchor points of the same class, we are able to define the prior using learned identity-preserving transformations, sample from the class manifold, and generate continuous paths on the latent space manifold.



(a)



(b)

Figure 6.7: Examples of images decoded from latent vectors sampled from the posterior of a model trained on naturally varying MNIST digits. In each example, the center digit (in the green box) is the encoded digit input and the surrounding digits are from the sampled latent vectors. Sampling in the VAELLS latent space results identity-preserving variations.

6.6 Summary

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

Another benefit of this model is the introduction of the anchor points into a transport-operator-based manifold representation. Until now, the transport operator model was always be trained using training point pairs. Modeling the manifold as a combination of specific anchor points and associated transport operators provides additional model flexibility. Training can be performed by selecting individual training points (rather than point pairs) and comparing the manifold paths between those points and the anchor points. Data augmentation can be initiated from the anchor points themselves rather than requiring new samples from class manifolds. Importantly for a potential human-inspired classification model, the anchor points could act as the stored representations of individual objects and new objects could be identified as transformed versions of these stored anchor points where the transformations are constrained by the transport operators.

CHAPTER 7
CONCLUSION

7.1 Summary

This dissertation is motivated by the goal of developing a human-inspired classification system. Rather than achieving transformation invariant classification through the use of large labeled datasets, as is done in modern state-of-the-art classification systems, the human-inspired classification system would directly incorporate models of transformations that are shared among classes. Therefore classification could be performed using a global dictionary of learned transformations and a few samples of each class. We develop and investigate a learnable transformation model, based on the transport operator manifold model, that would be the foundation to this human-inspired classification system. We show that the transformation model meets the following requirements set forth in the introduction. The transformations are:

- Learned from data with limited supervision.
- Constrained to the structure of an estimated data manifold.
- Transferable between classes.
- Capable of generating new samples.
- Able to infer relationships between samples.

We build models around this baseline transformation model that are applied to a diverse set of tasks which each highlight different aspects of the transformation model's performance and flexibility.

In Chapter 3, we develop a model that learns 3D transformations in the context of the depth inference inverse problem. We use this model to successfully learn rotational transport operators from 2D moving inputs. The manifold structure provided by the learned transport operators allows us to simultaneously infer the depth of projected points and the transformations between rotated versions of the same object. This model presents a hypothesis for how humans incorporate internal models of 3D rotation and use them to perform tasks like mental spatial transformation, depth inference, and third person perspective taking.

In Chapter 4, we extend the transport operator model to transfer learning scenarios. The ability for transport operators to generalize beyond the training domain is an essential characteristic necessary for developing our suggested transformation-informed classification system. To demonstrate transfer learning, we train transport operators in densely sampled source domains and present their successful application on samples in target domains with limited data. We define two transfer tasks which both highlight the ability of our transformation model to generate new samples. In the generative manifold mapping task, we generate new samples around data points in a target domain by applying the manifold representation with a probabilistic model on the coefficients associated with each manifold motion direction. In the structured manifold path generation task, we infer a transformation path between points in the source domain and apply that same transformation path to samples in the target domain. The transfer learning potential is shown in the context of data generation and data augmentation applications by animating individuals with new facial expressions and providing examples of few-shot learning for digit and facial expression classification.

In Chapter 5, we develop the manifold autoencoder which incorporates transport operators into the latent space of an autoencoder. With this model we are able to learn complex, semantically meaningful data transformations and adapt the latent space to fit the manifold structure associated with the learned transformations. By directly incorporating a manifold model into the latent space of an autoencoder, we have built-in methods for generating samples on the manifold and estimating geodesic paths between points. We train the manifold autoencoder on MNIST digits, Fashion MNIST images, motion capture gait sequences and CelebA face images. We perform nearest neighbor classification tasks using the manifold offset distance derived from the transport operator model. We also generate data by extrapolating estimated paths with inferred transport operator combinations and by randomly applying transport operators. To encourage identity-preserving data augmentation throughout the latent space, we incorporate a coefficient encoder which uses a pretrained classifier to learn local regions where operators are likely to be used while preserving the identity of transformed samples.

In Chapter 6, we introduce VAELLS which incorporate the transport operator model into the probabilistic framework of a VAE to define a manifold-based latent prior distribution. Developing this probabilistic model results in the introduction of anchor points which represent the scaffold off of which the transport operator-based manifold structure is built. The adaptable, manifold-based prior enables us to both interpolate and extrapolate nonlinear transformation paths in the latent space and to explicitly incorporate class separation by learning identity-preserving transformations. We verify the performance of this model on datasets with known latent structure and then extend it to real-world data to learn natural transformations.

7.2 Future Work

With the completion of this thesis work, we have contributed a transformation model that has been shown to learn complex, semantically meaningful transformations without requiring transformation labels. We have also demonstrated that the transport operators can represent structure that maps out the manifold beyond the source domain. With these key characteristics, we have laid the groundwork to build off of to develop a new human-inspired classification strategy. From here, there are three paths for innovation that could both build towards a human-inspired classification model and provide valuable insights on intermediate sub-problems. These three paths are:

1. Exploiting the flexibility of the manifold autoencoder model for learning representations of meaningful variations.
2. Developing a human-inspired classification system.
3. Enhancing biological-plausibility of the transformation model and comparing it to human visual performance.

7.2.1 Exploiting the Flexibility of Transport Operator Learning

The manifold autoencoder is a flexible model which can learn transport operators that represent data variations and adapt the latent space structure to fit the learned operators. This structure of the transport operators and latent space is enforced during training through the selection of training point pairs. By intelligently selecting point pairs, we can encourage the manifold autoencoder model to adapt the latent space to reflect the manifold associated with specific types of transformations.

One approach to intelligent supervision is to train the manifold model on data with semantic labels associated with training samples. Transport operators could be learned that are directly associated with specific, labeled transformations. This generative model of semantically labeled transformations could as the baseline for interactive robotics and recommender systems in which a machine could imagine an sample of interest through a transformation-based description provided by a human user.

Another appealing choice for intelligent supervision is to use training point pairs (or tuples) that are determined to be similar through human judgement. In many cases it is difficult to define the characteristics that humans use to identify whether samples are most similar to one another or whether samples should be placed in a certain order. By incorporating a human in the loop during point pair selection and manifold autoencoder training, the model could learn transformations that are specifically meaningful to humans. This could provide insights into which characteristics humans find important when comparing or ranking samples. A model of human-relevant transformations could aid in applications that typically require time-consuming expert guidance by distilling the information from experts and incorporating it into machine-based classifiers. Additionally, a system like this may be able to identify biases in how humans compare and rank samples.

While the above are just two examples of unique supervision strategies, there are many possibilities for future approaches which can tailor the developed model to new domains.

7.2.2 Developing a Human-Inspired Classification System

This thesis investigates the ability to transfer the transport operator manifold representation beyond the source domain and confirms that transport operators are generalizable to new points in target domains. Additionally, it demonstrates the effectiveness of the MAE and VAELLS models for representing complex transformations in rich datasets. Building from this starting point, future work can develop the discussed human-inspired classification system by learning a dictionary of operators that represent natural variations in the data and identifying new samples as transformed representations of anchor samples in each class. This is a new approach to classification which will require updates to the manifold model as well as new ways of identifying, defining, and storing anchor samples.

Steps towards achieving this goal of few shot learning involve investigating the ability to transfer transport operators between classes in the manifold autoencoder model. The manifold autoencoder model is necessary for learning complex transformations in high dimensional data but it adds complications to the transfer learning procedure. There are two transfer learning scenarios to consider in the manifold autoencoder framework. In the first scenario, there is ample unlabeled data for all classes. In this case, an autoencoder can be trained on all of the data and the transport operators can be learned on samples from limited classes and applied to new samples in the pretrained latent space. In the second scenario, there are novel classes with examples that will be presented after the initial training of the autoencoder network. In this case, the autoencoder may not present an accurate latent representation of the new data and greater consideration should be given to how to approach this problem. Suggested future work involves innovating in both of these scenarios to understand and improve the transferability of transport operators that represent complex transformations between classes.

Future work should also investigate the best way to combine the learned manifold model with anchor samples to develop the transformation-based classification system. Following from experiments shown in this thesis, a good initial approach is to investigate nearest neighbor classification by estimating the manifold offset distance from newly presented samples to anchor samples from each class. The selection for anchor samples for each class could be inspired by the anchor points introduced in the VAELLS model. The learned transport operators could also be used to augment exemplar anchor samples with known transformations to present a classifier with a diversity of samples during training without requiring a multitude of labeled data. Finally, the transport operator model could be incorporated directly into a neural network classifier in order to transform input points similar to the spatial transformer networks.

7.2.3 Enhancing Biological Plausibility

The models presented here have been inspired greatly by human performance but also by neuroscience research. In the future, this work can be expanded to more directly connect it to models of human visual performance. For instance, the work from chapter 3 can be compared more directly with human performance on mental rotation and structure from motion tasks. Specifically the computations in this model can be adjusted to be more biologically plausible. Additionally, our model can make estimates of performance with different stimulus characteristics. The hypotheses developed with our model can be tested in scientific experiments to develop a feedback loop between our machine learning model of the human visual performance and examples of real-world human visual performance.

Another path of research in this space is inspired by the work that identifies the untangling of the manifold structure in hierarchical visual processing stages [20, 21]. Specifically, future work can formulate the transport operator transformation model in a hierarchical fashion, akin to the layers of the visual cortex. This would allow for comparing performance of each transport operator-based layer to the performance of the layers of the visual cortex. Hierarchical structure in the transport operator model may be accomplished by learning transformations on data patches and integrating the data and transformation parameters from one layer to the next or by separating the the model into style and content paths where style is represented by the transport operators and content represents the underlying object.

Appendices

APPENDIX A
DETAILS ON LEARNING 3D MANIFOLD STURCTURE FROM 2D PROJECTIONS

A.1 Inference Details

We perform depth inference in subsection 3.3.2 through subsection 3.3.4 using the objective in Equation 3.10. In the inference experiments in subsection 3.3.3, we set $\gamma = 0.1$, $\zeta = 0.01$, and $\beta = 10^{-3}$. These parameters are selected because they yield accurate inferred depth estimates. We add Gaussian noise with a standard deviation of 10^{-3} to the ground truth operators in most experiments in subsection 3.3.3. Unless otherwise stated, we set $N_P = 20$, $\theta = 2°$, and $N_T = 30$. For the quantitative analysis of inference on many trials, we use inputs points that are 2D projections of random 3D points that are undergoing rotation about a randomly selected 3D axis. The rotation angle used for a given trial was sampled from a distribution with a mean of θ and a standard deviation of $0.516°$.

Because the training objective is nonconvex, optimization may result in local minima. To avoid resulting in local minima, we perform inference for the same inference window several times using several random restarts. That is, we randomly sample a new intialization and infer coefficients and depths using that starting point. This often results in different final inferred outputs. We choose the inferred output associated with the lowest final objective from inference. For the experiments in subsection 3.3.3, we use five random restarts.

For the inference experiments, we compute the mean squared error between the ground truth depths and estimated depths and the Kendall's Tau between the ordering of the truth depths and estimated depths. As mentioned in section 3.5, the kinematogram stimulus is bistable which means it can result in two separate percepts (i.e., clockwise rotation or counter-clockwise rotation). We observe switching in inferred direction and signs of the depths with our model and account for that when computing the depth inference metrics. Specifically we generate a path with the transformation defined by the inferred coefficients and observe the direction of motion of that path to determine which direction of rotation we inferred. If the inferred rotation is counter-clockwise then we multiply the depths by -1 because the rotation of the points in ground truth sequence is clockwise.

A.2 Training Details

We train the transport operators using gradient descent. The operators are intialized with Gaussian noise with a standard deviation of 0.3. The training run in the kinematogram experiments used the parameters in Table A.1.

Table A.1: Training parameters for learning rotational operators from 2D projected inputs.

Training Parameters
M : 3
lr_{begin} : 0.5
ζ : 0.1
γ : 0.15
β: 10^{-4}
θ : 10°
N_T : 20
N_P : 20
Training Steps: 10,000
number of restarts for coefficient inference: 25

We begin training at a specific learning rate and increase it if there is a successful learning step (i.e., one that decreases the learning objective) or decrease it if there is a failed learning step. While large learning rates aid in efficient gradient steps in the beginning of training, we find that decreasing the learning rate consistently towards the end of training leads to more stable final transport operator representations. Specifically, we start decreasing the learning rate at 3000 training steps and decay it by a multiplication factor of 0.9997 at each step.

Figure A.1 shows the trajectories of operators learned when $M = 6$. This highlights the usefulness of the Frobenius norm regularizer on the dictionary elements. If a transport operator is not being used for representing manifold paths, then its magnitude is reduced to nearly zero. Training with six operators utilizes the following parameters: $\zeta = 0.1$, $\gamma = 0.08$, $\beta = 10^{-4}$, $\theta = 10°$, $N_T = 20$, $N_P = 20$, 10000 training steps, and 25 random restarts.

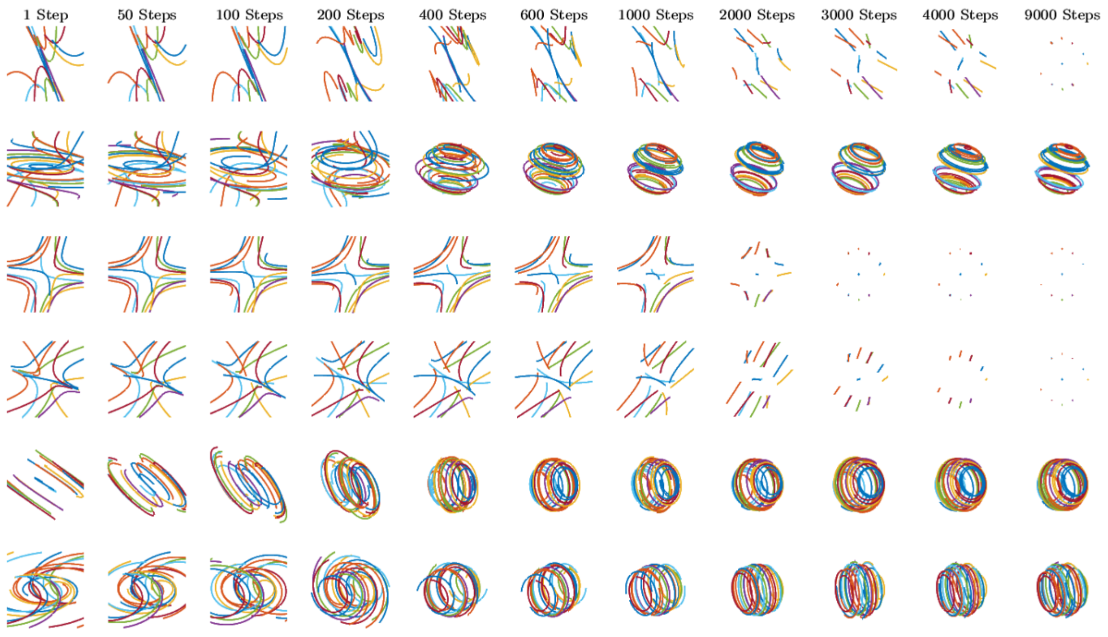


Figure A.1: Transport operator trajectories during training. Each row represents one of the six learned operators. Each column shows the trajectories at a different training step. The operators begin with random initializations at step 1 and reach a rotation structure around 400 steps. After that, the three operators that do not represent rotation have their magnitudes reduced because they are not being used. At the end of training, there are three operators with clear rotational structure.

A.3 Kinematogram Experimental Details

For the kinematogram experiments, we use a base set of parameters which are shown in Table A.2. For individual experiments, we vary subsets of the parameters from these baseline values. As with the other experiments, we correct for depth sign switching before computing the error metrics for the kinematogram tasks.

Table A.2: Parameters for random dot kinematogram experiments.

Kinematogram Inference Parameters
ζ : 0.01
γ : 0.1
β: 0
θ : 2°
ξ : 0
N_T : 30
N_P : 20
number of restarts for coefficient inference: 5

APPENDIX B
MANIFOLD AUTOENCODER DETAILS

B.1 Manifold Autoencoder Training Details

B.1.1 Training Strategy

In all experiments, we follow the general training procedure put forth previously [1]. 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 (Equation 5.1). Pairs of training images $\mathbf{x}_0, \mathbf{x}_1$ are selected using different selection methods depending on the experiment or application. The images are then encoded into the latent space $\mathbf{z}_0, \mathbf{z}_1$. 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 appendix section B.2. 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 and dictionary magnitudes reach a 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:

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

where E_Ψ is defined in (Equation 5.1). 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 5.6 and appendix section B.10.

Hyper-parameter tuning for all experiments was performed on the Georgia Tech Partnership for Advanced Computing Environment (PACE) clusters [224]. Experiments were performed using a Nvidia Quadro RTX 6000.

B.1.2 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:

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

where $\widehat{\boldsymbol{\Psi}}_m$ 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 (Equation 5.1). 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 (Equation 5.1) 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 of the operators are used to represent all of 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.

B.2 Coefficient Inference

In order to perform coefficient inference in the extended manifold autoencoder model used to learn unlabeled natural variations, 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 Equation 2.3, we apply a proximal gradient step in a forward-backward splitting scheme [225]. For the ℓ_1 norm, the proximal gradient has a known, closed-form solution in the soft-thresholding function [226]:

$$\mathcal{T}_\lambda(\mathbf{c}) = \text{sign}(\mathbf{c}) * \max\Big(|\mathbf{c}| - \lambda, 0\Big) \tag{B.3}$$

Let $\nabla_{\mathbf{c}} \frac{1}{2} \left\| \mathbf{z}_1 - \text{expm}\left(\sum_{m=1}^{M} \boldsymbol{\Psi}_m c_m\right) \mathbf{z}_0 \right\|_2^2 \approx \nabla \widetilde{f}(\mathbf{c})$ 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:

$$\mathbf{c}_{k+1} = \mathcal{T}_{\zeta\alpha_k}\left(\mathbf{c}_k - \alpha_k \nabla \widetilde{f}(\mathbf{c}_k)\right) \tag{B.4}$$

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 learning natural unlabeled variations, 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 [222, 225], and found that in many cases they resulted in worse performance.

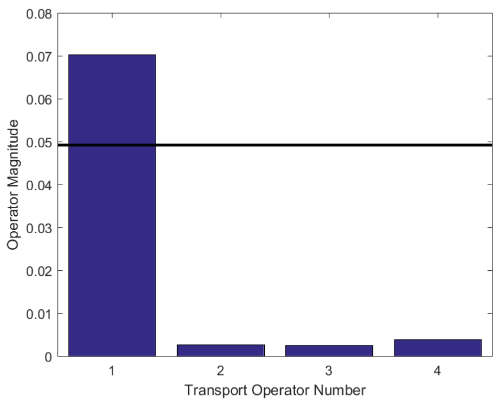
B.3 Details on Closed Transformation Path Experiments

All manifold autoencoder experiments were run in PyTorch [227]. In 2D circle experiment, we used single-layer neural networks with 512 hidden units and ReLU nonlinearities for both the encoder and decoder. Figure C.6 shows the magnitude of transport operators after the transport operator training phase. We select a threshold that is 70% of the magnitude of the transport operator with the maximum magnitude and use only transport operators with magnitudes above that threshold in the fine-tuning phase. In this setting the threshold is 0.0492 and only transport operator 1 surpasses the threshold. The training parameters are shown in Table C.3.

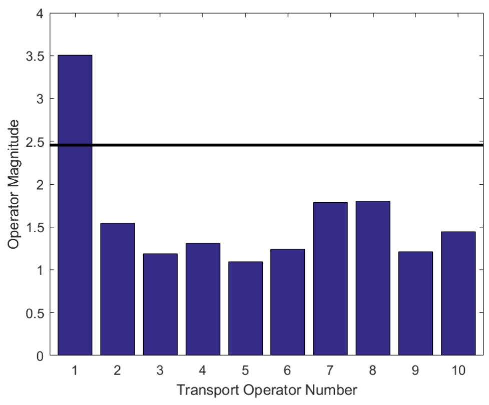
Table B.1: Training parameters for 2D circle experiment

Autoencoder Training	Transport Operator Training	Fine-tuning
batch size: 64	batch size: 10	batch size: 10
steps: 3000	steps: 3000	steps: 100
lr_ϕ : 0.0001	lr_ϕ : -	lr_ϕ : 0.0001
lr_Ψ : -	lr_Ψ : 5	lr_Ψ : 0.0005
ζ : -	ζ : 0.0001	ζ : 0.0
γ : -	γ : 0.005	γ : 0.0
λ : -	λ : -	λ : 1000
M : -	M : 4	M : 1

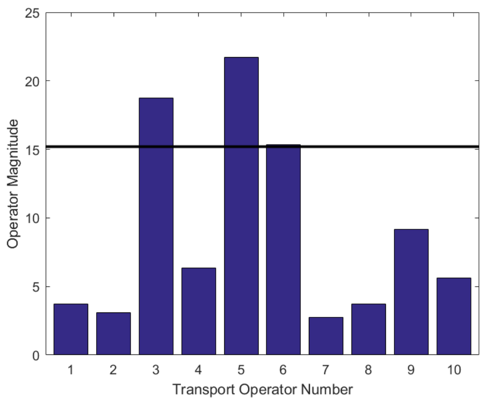
For training on MNIST digits we select 50,000 training digits from the traditional MNIST training set and save the additional 10,000 images for validation. We use the traditional MNIST test dataset for testing. We pre-process the MNIST digits by scaling the



(a)



(b)



(c)

Figure B.1: Magnitude of the operators after the transport operator training phase of the closed transformation path experiments. (a) For 2D circle experiment (b) For the rotated MNIST dataset (c) For gait sequences.

pixel values between 0 and 1. During transport operator training, we scale the latent vectors prior to performing coefficient inference in order to maintain a maximum value of about 1 for the latent vectors. The network architectures are given in Table B.2 and the training parameters are given in Table B.3. Figure B.1b shows the magnitude of transport operators after the transport operator training phase with the threshold for transport operator selection. In this experiment the threshold is 2.4556 and only transport operator 1 surpasses this threshold. Therefore only transport operator 1 is used during fine-tuning.

We use walking sequences from subject 35 in the CMU Graphics Lab Motion Capture Database in the gait experiment. We use sequences 1-16 for training, sequences 28 and 29

Table B.2: MAE architecture for rotated MNIST experiment

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

Table B.3: Training parameters for rotated MNIST experiment

Autoencoder Training	Transport Operator Training	Fine-tuning
batch size: 64	batch size: 32	batch size: 32
epochs: 25	steps: 2250	steps: 7800
lr_ϕ : 0.0001	lr_ϕ : -	lr_ϕ : 0.005
lr_Ψ : -	lr_Ψ : 0.01	lr_Ψ : 1
ζ : -	ζ : 0.01	ζ : 0.0
γ : -	$\gamma : 8e-5$	γ : 0.0
λ : -	λ : -	λ : 10
M : -	M : 10	M : 1

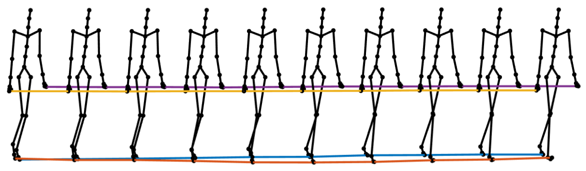
for validation, and sequences 30-34 for testing. During transport operator training, we scale the latent vectors prior to performing coefficient inference in order to maintain a maximum value of about 1 for the latent vectors. Figure B.1c shows the magnitude of transport operators after the transport operator training phase with the threshold for transport operator selection. The threshold in this experiment is 15.197 and transport operators 3, 5, and 6 all have magnitudes above that threshold. In this case, the transport operators from the transport operator training phase generate full gait sequences so we use these operators for our experiments without undergoing the fine-tuning phase. The network architectures are given in Table B.4 and the training parameters are given in Table B.5.

Table B.4: MAE architecture for gait experiment with closed transformation paths

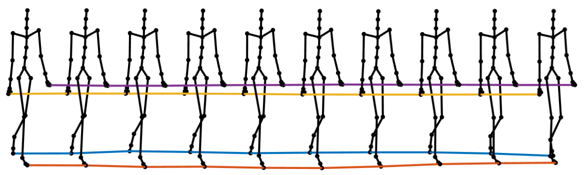
Encoder Network	Decoder Network
Input $\in \mathbb{R}^{50}$	Input $\in \mathbb{R}^{5}$
Linear: 512 Units	Linear: 512 Units
tanh	tanh
Linear: 512 Units	Linear: 512 Units
tanh	tanh
Linear: 512 Units	Linear: 512 Units
tanh	tanh
Linear: 5 Units	Linear: 50 Units

Table B.5: Training parameters for gait experiment with closed transformation paths

Autoencoder Training	Transport Operator Training
batch size: 64	batch size: 32
training steps: 15000	training steps: 14500
lr_ϕ : 0.0005	lr_ϕ : -
lr_Ψ : -	lr_Ψ : 0.005
ζ : -	ζ : 0.05
γ : -	γ : 0.0001
λ : -	λ : -
M : -	M : 10



(a)



(b)

Figure B.2: (a) Gait sequence generated from operator 3. (b) Gait sequence generated from operator 6.

B.4 Details on Unlabeled Natural Variation Experiments

B.4.1 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, and testing sets. The training set contains 60,000 images from the traditional MNIST training set. The traditional MNIST testing set is used for our testing set. The input images are scaled so their pixel values are between 0 and 1. The network architecture used for the autoencoder is shown in Table B.6. The training parameters for the transport operator training phase and the fine-tuning phase are shown in Table B.7 and Table B.8.

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 B.8. The image classifier we use is based on the simple LeNet architecture with two convolutional layers and three fully connected layers [177].

Table B.6: MAE architecture for MNIST and Fashion MNIST experiments with unlabeled natural variations

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 B.7: Training parameters for the MNIST transport operator training phase.

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 B.8: Training parameters for MNIST fine-tuning phase and coefficient encoder training

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

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

B.5 MNIST Experiment Additional Results

Here we show additional experimental details and results for the MNIST experiment. Figure B.3 shows the magnitude of all 16 operators after the fine-tuning phase. Six of the operators have their magnitudes reduced to zero. Figure B.4 shows the paths generated by transport operators trained on MNIST data.

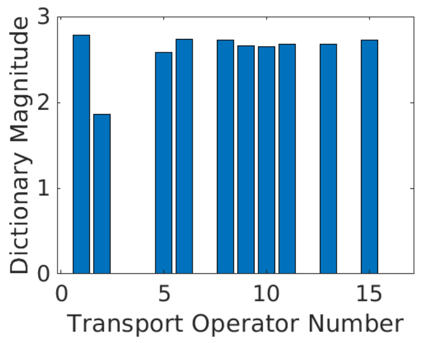


Figure B.3: The magnitudes of the learned operators after fine-tuning for the MNIST experiment with unlabeled natural variations.

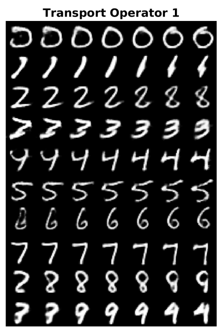
B.6 Fashion-MNIST Experiment Details

The Fashion-MNIST dataset is made available under the terms of the MIT license. We split the Fashion MNIST dataset into training, and testing sets. The training set contains 60,000 images from the Fashion-MNIST training set. The traditional Fashion-MNIST testing set is used for our testing set. The input images are scaled 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 B.6. The training parameters for the transport operator training phase and the fine-tuning phase are shown in Table B.9 and Table B.10.

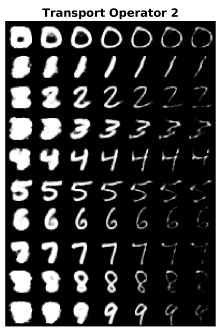
Prior to training the coefficient encoder for the Fashion-MNIST dataset, we train a classifier the latent vectors associated with labeled Fashion-MNIST data which we use to encourage identity-preservation during coefficient encoder training. The training parameters for the coefficient encoder are shown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

B.7 Fashion MNIST Experiment Additional Results

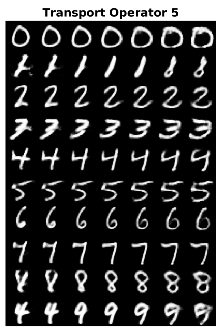
Here we show additional experimental details and results for the Fashion-MNIST experiment. Figure B.5 shows the magnitude of all 16 operators after the fine-tuning phase. Six of the operators had their magnitudes reduced to zero.



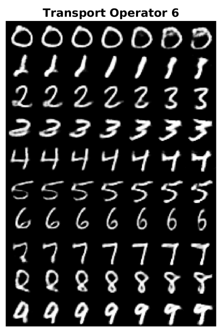
(a)



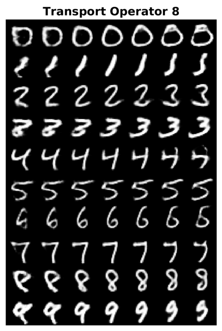
(b)



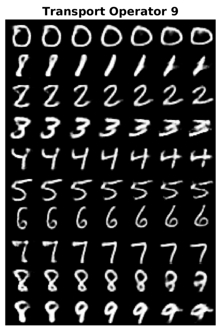
(c)



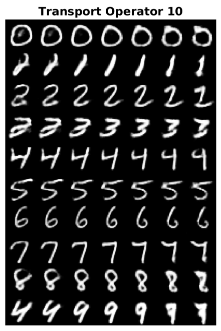
(d)



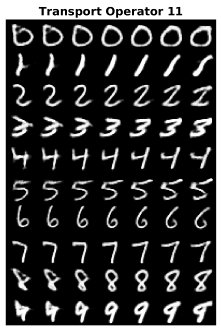
(e)



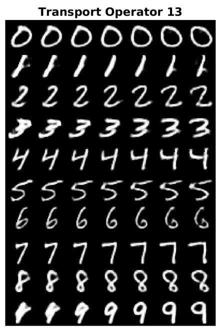
(f)



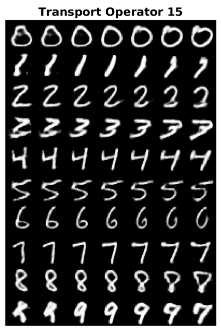
(g)



(h)



(i)



(j)

Figure B.4: Paths generated by all non-zero transport operators trained on the MNIST dataset with unlabeled natural variations. 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

B.8 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

Table B.9: Training parameters for the Fashion-MNIST transport operator training phase.

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

Table B.10: Training parameters for the Fashion-MNIST fine-tuning and coefficient encoder training.

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

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

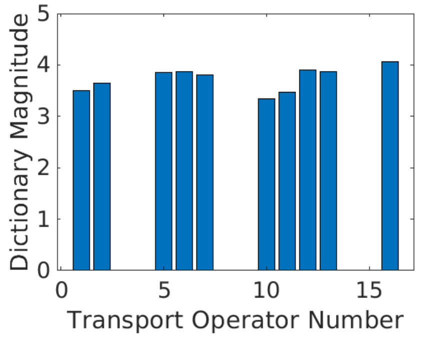
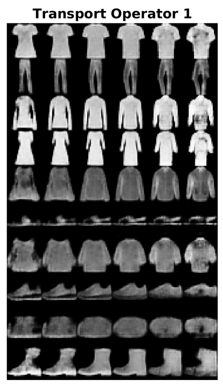
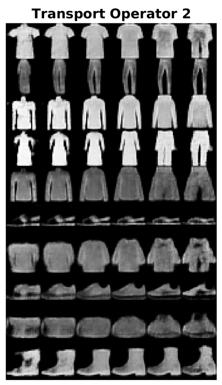


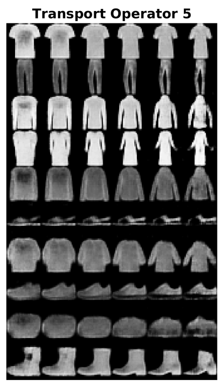
Figure B.5: The magnitudes of the learned operators after fine-tuning for the Fashion-MNIST experiment with unlabeled natural variations.



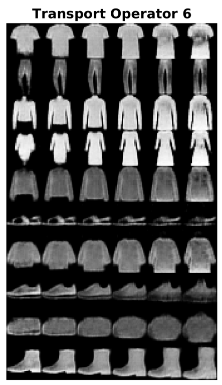
(a)



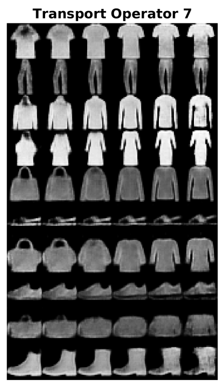
(b)



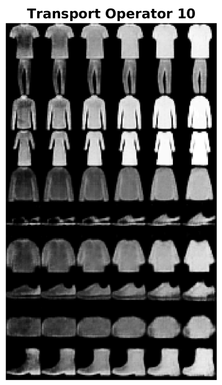
(c)



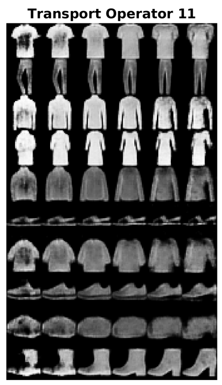
(d)



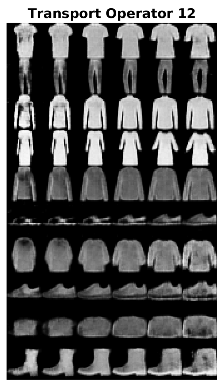
(e)



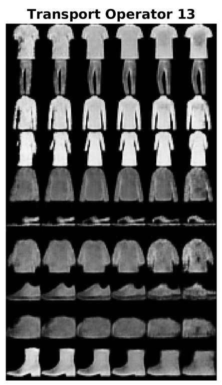
(f)



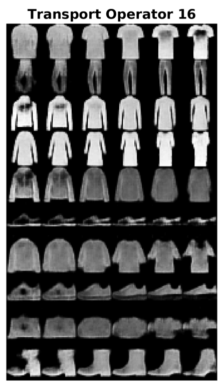
(g)



(h)



(i)



(j)

Figure B.6: 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

images accompanied with the entire test set. The input images are scaled so their pixel values are between 0 and 1. The network architecture used for the autoencoder is shown in Table B.11. The training parameters for the transport operator training phase and the fine-tuning phase are shown in Table B.12.

The attribute classifier is a ResNet-18 model adapted from Duo Li's github repository[1] 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 [217].

Table B.11: MAE 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

[1]https://github.com/d-li14/face-attribute-prediction

Table B.12: 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

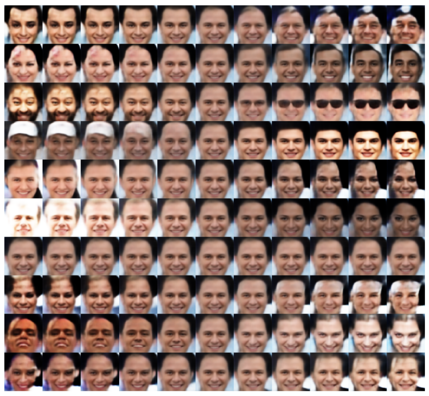
CelebA Fine-tuning Parameters
batch size: 500
fine-tuning epochs: 10
lr_{net} : 10^{-4}
lr_{Ψ} : 10^{-3}
ζ : 0.8
γ : 5×10^{-7}
λ: 0.75
number of network update steps: 50
number of Ψ update steps: 50

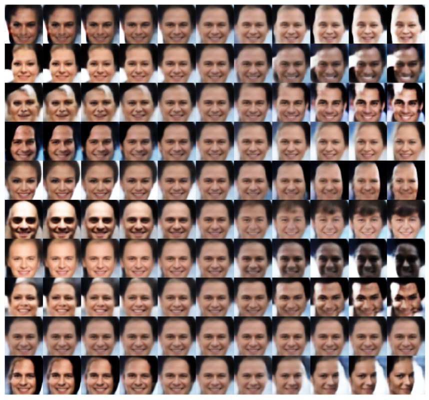
B.9 CelebA Experiment Additional Results

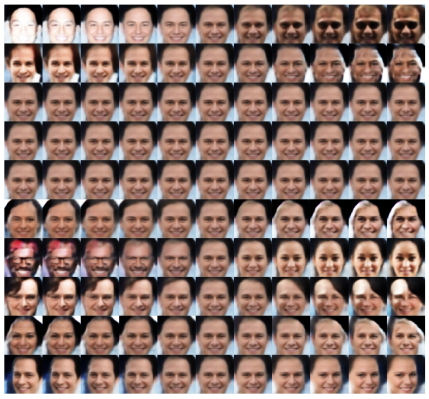
Here we show additional experimental results for the CelebA experiment. Figure B.7 shows the paths generated by the 40 transport operators trained on celebA data. Each row represents a different operator acting on the same input image. Images in the middle column of the image block are the reconstructed inputs and images to the right and left are images decoded from transformed latent vectors in positive and negative directions, respectively.

B.10 Coefficient Encoder Details

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 [30, 31]. 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.


(a)


(b)


(c)

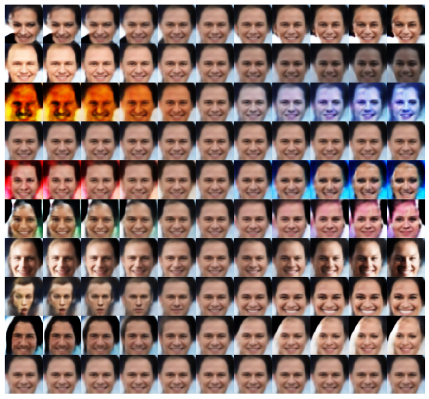

(d)

Figure B.7: 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

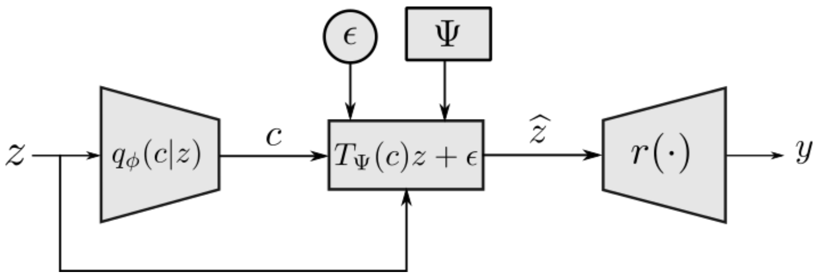


Figure B.8: System diagram for learning local coefficient statistics.

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 B.8. 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 [228] which introduces the variational posterior to estimate the log likelihood of the data.

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

$$\begin{aligned}
\log p_\theta(y|\mathbf{z}) &= \log \mathbb{E}_{p_\zeta(\mathbf{c})}\left[p_\theta(y|\mathbf{c},\mathbf{z})\right] && \text{(B.5)}\\
&= \log \int_{\mathbf{c}} p_\zeta(\mathbf{c}) p_\theta(y|\mathbf{c},\mathbf{z}) d\mathbf{c} && \text{(B.6)}\\
&= \log \int_{\mathbf{c}} q_\phi(\mathbf{c}|\mathbf{z}) \frac{p_\zeta(\mathbf{c})}{q_\phi(\mathbf{c}|\mathbf{z})} p_\theta(y|\mathbf{c},\mathbf{z}) d\mathbf{c} && \text{(B.7)}\\
&\geq \int_{\mathbf{c}} q_\phi(\mathbf{c}|\mathbf{z}) \log \left[\frac{p_\zeta(\mathbf{c})}{q_\phi(\mathbf{c}|\mathbf{z})} p_\theta(y|\mathbf{c},\mathbf{z})\right] d\mathbf{c} && \text{(B.8)}\\
&= E_{q_\phi}\left[\log p_\theta(y|\mathbf{c},\mathbf{z})\right] - D_{KL}\left(q_\phi(\mathbf{c}|\mathbf{z}) | p_\zeta(\mathbf{c})\right) && \text{(B.9)}\\
&= E_{q_\phi}\left[\log E_\epsilon p_\theta(y|\mathbf{c},\mathbf{z},\epsilon,\widehat{\mathbf{z}})\right] - D_{KL}\left(q_\phi(\mathbf{c}|\mathbf{z}) | p_\zeta(\mathbf{c})\right). && \text{(B.10)}
\end{aligned}$$

In Equation B.5 and Equation B.6, we marginalize over the coefficients $\mathbf{c}$ as required under our system model in Figure B.8. In Equation B.8, 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 Equation B.10.

In order to sample from $q_\phi(\mathbf{c} \mid \mathbf{z})$ when computing the expectations, we first use the reparameterization trick [3, 30, 31] to define the sampled coefficients $\widehat{\mathbf{c}}$ as a function of a uniform random variable $\mathbf{u} \sim \text{ Unif}\left(-\frac{1}{2}, \frac{1}{2}\right)^M$:

$$\widehat{\mathbf{c}} = l_\phi(\mathbf{u},\mathbf{z}) = -h_\phi(\mathbf{z})\,\text{sgn}(\mathbf{u})\log(1 - 2\,|\mathbf{u}|), \qquad \text{(B.11)}$$

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:

$$E_{q_\phi}\left[\log E_\epsilon p_\theta(y|\mathbf{c},\mathbf{z},\epsilon,\widehat{\mathbf{z}})\right] = E_{\mathbf{u}}\left[\log E_\epsilon p_\theta(y|\widehat{\mathbf{c}} = l_\phi(\mathbf{u},\mathbf{z}),\epsilon,\widehat{\mathbf{z}})\right] \tag{B.12}$$

$$\approx \frac{1}{JK}\sum_{j=1}^{J}\log\sum_{k=1}^{K} p_\theta\left(y|\widehat{\mathbf{c}}^{(j)} = l_\phi(\mathbf{u}^{(j)},\mathbf{z}),\epsilon^{(k)},\widehat{\mathbf{z}}\right). \tag{B.13}$$

Our pre-trained classifier defines p_θ which allows us to specify final objective using $r(\cdot)$. 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:

$$\log p_\theta(y|\mathbf{z}) \geq -\frac{1}{J}\sum_{j=1}^{J} D_{KL}\left(y_i|r(\widehat{\mathbf{z}}_i^{(j)})\right) - D_{KL}\left(q_\phi(\mathbf{c}|\mathbf{z})|p_\zeta(\mathbf{c})\right), \tag{B.14}$$

where in practice we use a single sample $J = 1$ and we estimate $D_{KL}\left(y_i|r(\widehat{\mathbf{z}}_i^{(j)})\right) \approx D_{KL}\left(r(\mathbf{z}_i^{(j)})|r(\widehat{\mathbf{z}}_i^{(j)})\right)$ for test samples for which we may not have class labels. These practical simplifications result in the objective in Equation 5.6 which we want to minimize. The KL divergence between the $q_\phi(\mathbf{c}|\mathbf{z})$ and $p_\zeta(\mathbf{c})$ has a closed form expression [229]:

$$D_{KL}\left(q_\phi(\mathbf{c} \mid \mathbf{z})\|p_\zeta(\mathbf{c})\right) = \log(h_\phi(\mathbf{z})) - \log(\zeta) + \frac{\zeta}{h_\phi(\mathbf{z})} - 1 \tag{B.15}$$

B.11 Training Comparison Techniques

B.11.1 Hyperspherical VAE

Our hyperspherical VAE implementation came from the Nicola De Cao's github page.[2] For both the concentric circle dataset and the gait sequences dataset, we trained the hyperspherical VAE with the same network architectures as our autoencoder experiments and a mean square error reconstruction loss. For the rotated MNIST dataset, we used the network architecture from the MNIST example in the hyperspherical VAE code and used the binary cross entropy loss for the reconstruction error on dynamically binarized rotated digit images.

To estimate paths on the hyperspherical VAE latent space, we used the Manifold-valued Image Restoration Toolbox[3] to compute geodesic paths on a 10-dimensional hypersphere.

B.11.2 Contractive Autoencoder and β- VAE

In order to train the contractive autoencoder, 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 Equation C.6). 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 [203] with $\beta = 10$ for CelebA and $\beta = 5$ for MNIST and Fashion-MNIST. We find that setting β any higher results in poor reconstructive performance. Scripts for training both comparison methods are included in the code repository[4].

[2]https://github.com/nicola-decao/s-vae-pytorch/tree/master/hyperspherical_vae
[3]https://ronnybergmann.net/mvirt/
[4]https://github.com/siplab-gt/manifold-autoencoder-extended

APPENDIX C
VARIATIONAL AUTOENCODER WITH LEARNED LATENT STRUCTURE DETAILS

C.1 Derivation of the Loss Function

In this section, we describe the details of our VAELLS implementation. We define the loss function E to be minimized as the approximate negative ELBO in Equation 6.7, restated here for convenience:

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

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

$$\begin{gathered}
\widehat{E}(\mathbf{x}) \equiv -\frac{1}{N_s}\sum_{s=1}^{N_s} L_{\mathbf{x}}(T_{\mathbf{\Psi}}(l(\mathbf{u}_s;b))f_\phi(\mathbf{x}) + \gamma\varepsilon_s) + \frac{\eta}{2}\sum_{m=1}^{M}\|\mathbf{\Psi}_m\|_F^2 \qquad \text{(C.1)} \\
\varepsilon_s \sim \mathcal{N}(0,I) \quad \mathbf{u}_s \sim \text{ Unif}\left(-\frac{1}{2},\frac{1}{2}\right)^M .
\end{gathered}$$

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

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

where for $b > 0$,

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

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

$$l_m(\mathbf{u}; b) = -b \operatorname{sgn}(\mathbf{u}_m) \log(1 - 2\,|\mathbf{u}_m|).$$

Next we derive expressions for each expanded term in Equation C.1. For the likelihood term, we have

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

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

$$\log q_\phi(\mathbf{z} \mid \mathbf{x}) = \log \int_{\mathbf{c}} q_\phi(\mathbf{z}, \mathbf{c} \mid \mathbf{x}) d\mathbf{c} \tag{C.2}$$

$$\approx \max_{\mathbf{c}} \log\left[q_\phi(\mathbf{z} \mid \mathbf{c}, \mathbf{x}) q(\mathbf{c})\right], \tag{C.3}$$

where the approximation in Equation C.3 is motivated by the fact that the sparsity-inducing Laplace prior on $\mathbf{c}$ typically results in joint distributions with $\mathbf{z}$ that are tightly peaked in the coefficient space, as described in Olshausen & Field (1997) [219]. Due to this tight peak in $q_\phi(\mathbf{z}, \mathbf{c} \mid \mathbf{x})$, we can approximate the integral in Equation C.2 by approximating the volume under the surface in a small neighborhood around the density maximizer. Specifically, we approximate this volume by evaluating the joint density at its maximum, weighted by the volume of a ball with a small radius centered at the maximizer. This volume approximation results in an additive constant in the log posterior that reflects the area of the chosen neighborhood, which we omit from this log-likelihood computation.

We have

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

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

$$\mathbf{c}^*(\mathbf{z}, \mathbf{x}; \zeta) = \operatorname*{argmin}_{\mathbf{c}} \left[\zeta_2 \left\|\mathbf{z} - T_{\mathbf{\Psi}}(\mathbf{c})f_\phi(\mathbf{x})\right\|_2^2 + \zeta \sum_{m=1}^{M} |c_m|\right], \tag{C.4}$$

we have (with hyperparameter ζ_q)

$$\log q_\phi(\mathbf{z} \mid \mathbf{x}) \approx C_2 - \zeta_2 \left\|\mathbf{z} - T_{\mathbf{\Psi}}(\mathbf{c}^*(\mathbf{z}, \mathbf{x}; \zeta_q))f_\phi(\mathbf{x})\right\|_2^2 - \zeta_3 \sum_{m=1}^{M} \left|\mathbf{c}_m^*(\mathbf{z}, \mathbf{x}; \zeta_q)\right|. \tag{C.5}$$

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

$$-\log p_\theta(\mathbf{z}) = -\log \frac{1}{N_a}\sum_{i=1}^{N_a} q_\phi(\mathbf{z} \mid \mathbf{a_i})$$

$$\approx \log N_a - C_2 - \log\sum_{i=1}^{N_a}\exp\left(-\zeta_2 \left\|\mathbf{z} - T_{\mathbf{\Psi}}(\mathbf{c}^*(\mathbf{z}, \mathbf{a_i}; \zeta_p))f_\phi(\mathbf{a_i})\right\|_2^2 - \zeta_3 \sum_{m=1}^{M} \left|\mathbf{c}_m^*(\mathbf{z}, \mathbf{a_i}; \zeta_p)\right|\right),$$

where we use the same approximation as Equation C.3.

All together, dropping additive constants and letting $\mathbf{z}_s = T_{\mathbf{\Psi}}(l(\mathbf{u}_s))f_\phi(\mathbf{x}) + \gamma\varepsilon_s$, we have

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

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

$$\widehat{E}(\mathbf{x}) = \frac{1}{N_s}\sum_{s=1}^{N_s}\zeta_1\left\|\mathbf{x}-g_\theta(\mathbf{z}_s)\right\|_2^2 - \zeta_2\left\|\mathbf{z}_s - T_{\mathbf{\Psi}}(\mathbf{c}^*(\mathbf{z}_s,\mathbf{x};\zeta_q))f_\phi(\mathbf{x})\right\|_2^2 - \zeta_3\sum_{m=1}^{M}\left|\mathbf{c}_m^*(\mathbf{z}_s,\mathbf{x};\zeta_q)\right|$$
$$-\log\sum_{i=1}^{N_a}\exp\left(-\zeta_4\left\|\mathbf{z}_s - T_{\mathbf{\Psi}}(\mathbf{c}^*(\mathbf{z}_s,\mathbf{a}_\mathbf{i};\zeta_p))f_\phi(\mathbf{a}_\mathbf{i})\right\|_2^2 - \zeta_5\sum_{m=1}^{M}\left|\mathbf{c}_m^*(\mathbf{z}_s,\mathbf{a}_\mathbf{i};\zeta_p)\right|\right) + \frac{\eta}{2}\sum_{m=1}^{M}\|\mathbf{\Psi}_m\|_F^2. \quad \text{(C.6)}$$

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

C.2 DETAILS ON VAELLS TRAINING PROCEDURE

In this section we provide additional details on the general training procedure for all of the experiments. A detailed view of the VAELLS training steps is provided in algorithm 2. For specific architecture details and parameter selections, see each experiment's respective Appendix section.

Network training To enhance the generative capability of the decoder, in some experiments, we use a warm-up as in Tomczak & Welling (2018) [220]. Our warm-up includes updates to the network weights driven by only the reconstruction loss. During warm up there is no Gaussian sampling in the latent space and the sampled Laplace distribution for the transport operator coefficients has a large ζ_3 parameter which encourages sampling coefficients that are very close to zero . As mentioned in section 6.5, during VAELLS training we alternate between steps where we update the network weights and anchor points while keeping the transport operators fixed and steps where we update the transport operators while keeping the network weights and anchor points fixed. As we alternate between these steps, we vary the weights on the objective function terms. Specifically, we decrease the importance of the prior terms during the steps updating the network weights and decrease the importance of the reconstruction term during the steps updating the transport operators.

Transport operator learning As mentioned in section 6.5, one component of computing the prior and posterior objective is the coefficient inference between the sampled point $\mathbf{z}$ and the neural network encoding $f_\phi(\mathbf{x})$ as well as all the encoded anchor points $f_\phi(\mathbf{a}_\mathbf{i})$.

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

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

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

$$\begin{aligned}\widehat{E}_{\text{transopt}}(\mathbf{x}) = \frac{1}{N_s}\sum_{s=1}^{N_s} -\zeta_2 \left\|\mathbf{z}_s - T_{\mathbf{\Psi}}(c^*(\mathbf{z}_s, \mathbf{x}; \zeta_q)) f_\phi(\mathbf{x})\right\|_2^2 - \zeta_3 \sum_{m=1}^{M} |c_m^*(\mathbf{z}_s, \mathbf{x}; \zeta_q)| \\ - \log \sum_{i=1}^{N_a} \exp\left(-\zeta_4 \left\|\mathbf{z}_s - T_{\mathbf{\Psi}}(c^*(\mathbf{z}_s, \mathbf{a_i}; \zeta_p)) f_\phi(\mathbf{a_i})\right\|_2^2 - \zeta_5 \sum_{m=1}^{M} |c_m^*(\mathbf{z}_s, \mathbf{a_i}; \zeta_p)|\right)\end{aligned} \tag{C.7}$$

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

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

$$p_\theta(\mathbf{z}) = q_\phi(\mathbf{z} \mid \mathbf{a}^*), \tag{C.8}$$

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

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

- The weight on the reconstruction term (ζ_1) - use this in combination with warm-up steps to ensure reasonable reconstruction accuracy from the decoder.

- The posterior coefficient inference weight (ζ_q) - this is the weight on the sparsity regularizer term used in the objective Equation C.4 during coefficient inference between points in the *posterior* term. If this weight is too large, then inference can result in zero coefficients for all the operators which is not informative.

- The prior coefficient inference weight (ζ_p) - this is the weight on the sparsity regularizer term used during coefficient inference between points in the *prior* term.
- Number of restarts used during coefficient inference for transport operator training.
- Starting lr_{Ψ} - As mentioned above, we do vary lr_{Ψ} during transport operator training depending on whether our training steps are successful or not. If this learning rate starts too high, it can result in many unsuccessful steps with no updates on the transport operator dictionaries which greatly slows down training.

Comparison techniques We implemented the hyperspherical VAE [81] using the code provided by the authors: https://github.com/nicola-decao/s-vae-pytorch. For the concentric circle and swiss roll experiments we used the network specified in Table C.1. For MNIST experiments, we used the network architecture from their MNIST experiments, and we dynamically binarized the MNIST inputs as they did. We implemented the VAE with VampPrior [220] model using the code provided by the authors: https://github.com/jmtomczak/vae_vampprior. For the concentric circle and swiss roll experiments, we used the network architecture specified in Table C.1 and 100 pseudoinputs for each experiment. For the MNIST experiments we adapted the network in Table C.4 to add a linear layer between the final convTranspose layer and the sigmoid layer. We used 500 pseudoinputs in each of the MNIST experiments. The VAE with VampPrior MNIST tests were also performed on dynamically binarized MNIST data. We implemented our own VAE code with the same network architectures detailed in Table C.1 and Table C.4.

C.3 Variational Posterior Contours

In order to more intuitively visualize the learned variational posterior we show contour plots of the variational posterior given an input point. The variational posterior (Equation C.5) consists of two terms: the data fidelity term $\left(-\zeta_2 \left\|\mathbf{z} - T_{\Psi}(c^*(\mathbf{z}, \mathbf{x}; \zeta_q))f_{\phi}(\mathbf{x})\right\|_2^2\right)$ and the

Algorithm 2: Training of network weights, transport operators, and anchor points

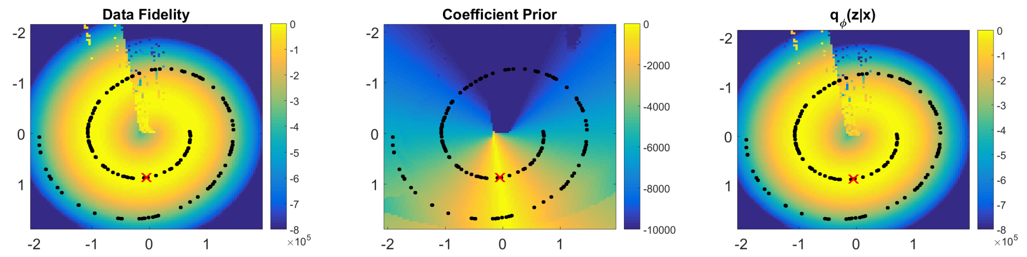
```
**Data:** Training samples $\mathcal{X}$, anchor points $\{a_1, ..., a_{N_a}\}$ selected from the input space
**Result:** Trained transport operator dictionary elements $\{\mathbf{\Psi}_1, ...\mathbf{\Psi}_M\}$, network weights $\phi$ and $\theta$, fine-tuned anchor points $\{\mathbf{a}_1, ..., \mathbf{a}_{N_a}\}$
$\mathbf{\Psi}$, $\phi$, $\theta \leftarrow$ randomly initialize dictionaries, network weights;
for $k = 0, ...., N$ do
    Sample mini-batch from $\mathcal{X}$: $\{\mathbf{x}_1, ..., \mathbf{x}_{N_s}\}$;
    for $s = 0, ...., N_s$ do
        Encode samples: $\mu_s \leftarrow f_\phi(\mathbf{x}_s)$;
        Sample $\mathbf{u}_s \sim$ Unif $\left(-\frac{1}{2}, \frac{1}{2}\right)^M$;
        $\widehat{c}_s \leftarrow l(\mathbf{u}_s)$;
        Sample $\mathbf{z}_s$ from $q_\phi(\mathbf{z}_s \mid \widehat{c}_s, \mathbf{x}_s) \sim \mathbf{T}_{\mathbf{\Psi}}(\widehat{c}_s)\mu_s + \gamma\varepsilon_s$;
        Decode sampled vectors: $\widehat{\mathbf{x}}_s \leftarrow g_\theta(\mathbf{z}_s)$;
        $c^*(\mathbf{z}_s, \mathbf{x}_s) \leftarrow$ Infer_Coefficients$(\mathbf{z}_s, \mathbf{x}_s, d_{\text{start}}, d_{\text{stop}})$;
        for $i = 0, ...., N_a$ do
            for $r = 0, ...., \text{num\_restart}$ do
                $\mathbf{c}^{(r)}(\mathbf{z}_s, \mathbf{a_i}) \leftarrow$ Infer_Coefficients$(\mathbf{z}_s, \mathbf{a_i}, d_{\text{start}}, d_{\text{stop}})$;
                $E_c(\mathbf{c}^{(r)}(\mathbf{z}_s, \mathbf{a_i})) \leftarrow \log\left[q_\phi(\mathbf{z}_s \mid \mathbf{c}^{(r)}(\mathbf{z}_s, \mathbf{a_i}), \mathbf{a_i})q(\mathbf{c}^{(r)}(\mathbf{z}_s, \mathbf{a_i}))\right]$
            $c^*(\mathbf{z}_s, \mathbf{a_i}) \leftarrow \text{argmax}_r\, E_c(\mathbf{c}^{(r)}(\mathbf{z}_s, \mathbf{a_i}))$
    Calculate $\widehat{E}$ in (Equation C.6) using $\widehat{\mathbf{x}}_s$, $c^*(\mathbf{z}_s, \mathbf{x}_s)$, and $c^*(\mathbf{z}_s, \mathbf{a_i})$ for $s = 0, ...., N_s$ and $i = 0, ...., N_a$;
    $\mathbf{\Psi}_{\text{new}} \leftarrow \mathbf{\Psi} - lr_{\mathbf{\Psi}}\frac{\delta\widehat{E}}{\delta\mathbf{\Psi}}$;
    if $\widehat{E}_{\text{transopt}}(\mathbf{\Psi}_{\text{new}}) < \widehat{E}_{\text{transopt}}(\mathbf{\Psi})$ then
        $\mathbf{\Psi} \leftarrow \mathbf{\Psi}_{\text{new}}$;
        $lr_{\mathbf{\Psi}} \leftarrow \frac{lr_{\mathbf{\Psi}}}{\text{decay}}$
    else
        $lr_{\mathbf{\Psi}} \leftarrow lr_{\mathbf{\Psi}} \cdot \text{decay}$
    $\phi \leftarrow \phi - lr_{\text{net}}\frac{\delta\widehat{E}}{\delta\phi}$;
    $\theta \leftarrow \theta - lr_{\text{net}}\frac{\delta\widehat{E}}{\delta\theta}$;
    $\mathbf{a} \leftarrow \mathbf{a} - lr_{\text{anchor}}\frac{\delta\widehat{E}}{\delta a}$;
```

Algorithm 3: Infer_Coefficients$(\mathbf{z}, \mathbf{x}, d_{\text{start}}, d_{\text{stop}})$

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

coefficient prior term $\left(-\zeta_3 \sum_{m=1}^{M} |c_m^*(\mathbf{z}, \mathbf{x}; \zeta_q)|\right)$. The data fidelity term expresses the probability of $\mathbf{z}$ given an input point $\mathbf{x}$. Figure C.1a shows a contour of the data fidelity term. The red x is the location of the encoded point and the black dots are encoded data samples. This shows how the data fidelity term maps out the contour of the latent swiss roll manifold around the encoded point. The coefficient prior is a Laplace distribution which encourages the coefficients to be tightly peaked around zero. Figure C.1b shows the contour of the prior term. There is a high log probability in the slice of the space that can be reached by applying small coefficients to the transport operators and the log probability reduces in parts of the space that require larger coefficient values to reach.

The final variational posterior in Figure C.1c is a combination of these two terms. In our experimental setting, the data fidelity term is the dominant term that characterizes the variational posterior. However, as the weights ζ_2 and ζ_3 vary, the contribution of each term towards the variational posterior will change.



(a) (b) (c)

Figure C.1: Variational posterior contour plots. The red x is the location of the encoded point $\mathbf{z}$. The black dots are encoded data points. (a) Contour of data fidelity term (b) Contour of coefficient prior term (c) Contour of $\log q_\phi(\mathbf{z} \mid \mathbf{x})$

C.4 Estimated Log-Likelihood

We estimate the log-likelihood using the importance sampling from [223] with 500 datapoints and 100 samples per datapoint. We follow the derivation from appendix section C.1 for computing $\log p_\theta(\mathbf{x} \mid \mathbf{z})$, $\log q_\phi(\mathbf{z} \mid \mathbf{x})$, and $\log p_\theta(\mathbf{z})$ and include all the constants defined there. We set ζ_1 to the same value used during training. While we hand-select

ζ_2, ζ_3, ζ_4, and ζ_5 during training, when computing the estimated log-likelihood, we compute each of these values based on the parameters γ and b which are used for sampling the latent vectors as specified in Equation 6.3. To be explicit, $\zeta_2 = \zeta_4 = 2^{-1}\gamma^{-2}$ and $\zeta_3 = \zeta_5 = b^{-1}$. When we infer coefficeints as in Equation C.4, we set $\zeta_2 = 1$ and we set the hyperparameters ζ_q and ζ_p (which are used for the posterior and prior respectively) to the same values used during training which yield successful coefficient inference. As noted by Equation C.3, the approximation of $\log \int_c q_\phi(\mathbf{z}, c \mid \mathbf{x})dc$ will result in an additive constant that we do not include in our estimated log-likelihood computation.

The results presented in Table 6.2 show a wide variation in estimated log-likelihood over the trials. These values are largely dominated by the data fidelity term in the prior which decreases significantly as the number of dictionaries increase from 2 to 8. Because this experiment sets $\gamma = 0.001$, any variation in the data fidelity terms for the $\log q_\phi(\mathbf{z} \mid \mathbf{x})$ and $\log p_\theta(\mathbf{z})$ are magnified when they are multiplied by $\zeta_2 = \zeta_4 = \frac{1}{2(0.001^2)}$.

C.5 Swiss Roll Experiment

Table C.1 and Table C.2 contain the VAELLS network architecture and parameters for the swiss roll experiment. In this experiment, we sample our ground truth 2D data manifold from a swiss roll and then map it to the 20-dimensional input space using a random linear mapping. We use 1000 swiss roll training points and randomly sample swiss roll test points. We initialize anchor points as points that are spaced out around the swiss roll prior to mapping to the 20-dimensional input space. We allow for the anchor points to be updated; however, these updates result in negligible changes to the anchor points. As described in section C.2, in this experiment we define the prior only with respect to the anchor points that are estimated to be closest to each training point.

Table C.1: VAELLS network architecture for swiss roll and concentric circle experiments

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

C.6 Analysis of Sensitivity to Anchor Points

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

We begin by varying the number and location of anchor points. In the initial formulation of the swiss roll experiment, the anchor points are selected to be well distributed around the swiss roll manifold. Figure 6.3b shows the locations of the anchor points used for experiment detailed in the paper. The anchor points are well spaced around the swiss roll manifold. Figure 6.3a shows the learned operator from this experiment which successfully generates paths with the desired swiss roll manifold structure.

While the experiment we presented uses four well-spaced anchor points, we can vary both the number and locations of the anchor points and still learn a mapping to a swiss roll latent structure and a transport operator that generates paths along the swiss roll. Figure C.2 shows examples of tests where VAELLS successfully learns the swiss roll structure in the latent space with different numbers of anchor points. While we achieve success with randomly positioned anchor points, there are tests where the anchor points are relatively evenly spaced over the manifold and the transport operators do not learn the swiss roll structure. Figure C.3 shows examples of tests where VAELLS fails to learn the swiss roll manifold even when the anchor points are relatively evenly spaced on the encoded manifold. These

Table C.2: VAELLS training parameters for swiss roll experiment

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

results indicate that the success of learning transport operators to represent the true latent manifold depends on a factor other than the anchor point locations.

Another possible factor important for successful learning of the latent manifold is the initialization of the dictionary elements. To analyze the impact of dictionary initialization, we train 10 separate instances of VAELLS which we initialize with the same network weights and anchor point locations and a different dictionary element weights. Figure C.4 shows the final training outputs of five of these trials. The trials shown in Figure C.4(a-b) successfully learn a transport operator that traverses a swiss roll manifold and the trials

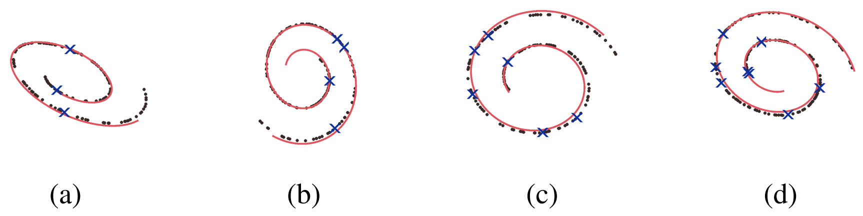


Figure C.2: Outputs of trials where VAELLS learns the correct swiss roll manifold while the number of anchor points is varied and the anchor point locations are randomly selected on the manifold. The black dots are encoded data points, the blue x's are the encoded anchor point locations, and the red line is the generated orbit of the learned transport operator. Each plot shows a result with a different number of anchor points: (a) 3 anchor points (b) 4 anchor points (c) 6 anchor points (d) 8 anchor points.

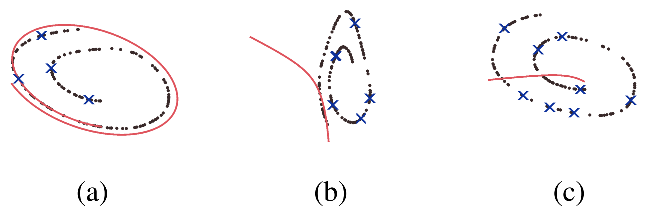


Figure C.3: Outputs of trials where VAELLS fails to learn the correct swiss roll manifold while the number of anchor points is varied and the anchor point locations are randomly selected on the manifold. The black dots are encoded data points, the blue x's are the encoded anchor point locations, and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

shown in Figure C.4(c-e) fail to learn a transport operator that traverses the swiss roll manifold. This experiment shows that the final transport operator orbits and latent space encoding can vary significantly with different initializations of the dictionary element and this indicates that the dictionary initialization is a more important factor for successful training than the anchor point locations.

The transport operator objective is a non-convex optimization surface which can result in the optimization settling in local minima that do not represent the true data manifold. There are a few cues to observe when determining whether a training run is successfully learning the data manifold. First, as mentioned in appendix section C.2, during every transport operator update step, we check whether the update improves the portion of the objective that incorporates the transport operator. If this portion of the objective does not

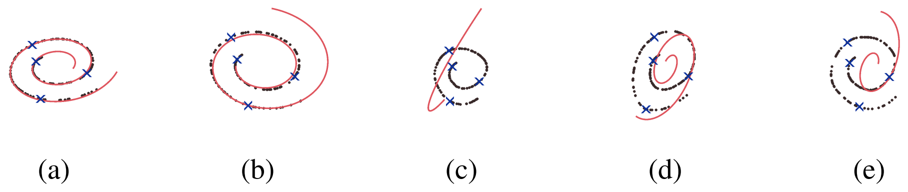


Figure C.4: Outputs of trials that are initialized with the same network weights and anchor point locations and different dictionary weights. The black dots are encoded data points, the blue x's are the encoded anchor point locations, and the red line is the generated orbit of the learned transport operator. (a-b) Trials that successfully learn a transport operator that traverses the swiss roll structure. (c-e) Trials that fail to learn a transport operator that traverses the swiss roll structure.

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

C.7 Concentric Circle Experiment

The concentric circle experiment uses the same network architecture as the swiss roll experiment which is specified in Table C.1. Table C.3 shows the training parameters for the concentric circle experiment. In this experiment, we sample our ground truth 2D data manifold from two concentric circles and then map those sampled points to the 20-dimensional input space using a random linear mapping. We use 400 training points and randomly sample test points.

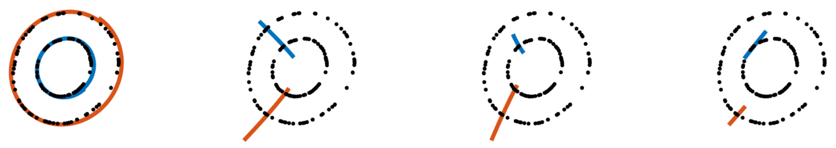


Figure C.5: The orbits of each of the transport operators learned in the concentric circle test case plotted on top of encoded points.

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

Figure C.5 shows the encoded latent points overlaid with the orbits of the learned transport operators. These orbits are generated by selecting one point on each circular manifold and applying a single operator as its trajectory evolves over time. Notice that one of the operators clearly represents the circular structure of the latent space while the other three have much smaller magnitudes and a limited effect on the latent space transformations. The Frobenius norm regularizer in the objective function often aids in model order selection by reducing the magnitudes of operators that are not used to represent transformations between points on the manifold. To see the magnitudes more clearly, Figure C.6 shows the magnitude of each of the transport operators after training the VAELLS model in the concentric circle test case.

Figure C.7a shows latent points sampled from the prior using the sampling described in Equation 6.3 with larger standard deviation and scale parameters to aid in visualization.

Table C.3: VAELLS training parameters for concentric circle experiment

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

Figure C.7(b-c) show two example inferred paths between points encoded on the concentric circle manifold.

C.8 Rotated MNIST Experiment

We split the MNIST dataset into training, validation, and testing sets. The training set contains 50,000 images from the traditional MNIST training set. The validation set is made up of the remaining 10,000 image from the traditional MNIST training set. We use the traditional MNIST testing set for our testing set. The input images are scaled by 255 to keep the pixel values between 0 and 1. To generate a batch of rotated MNIST

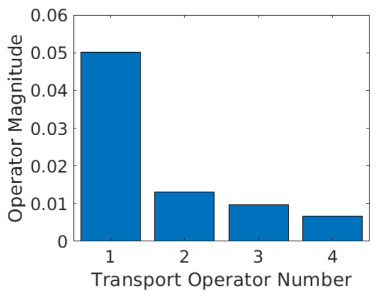
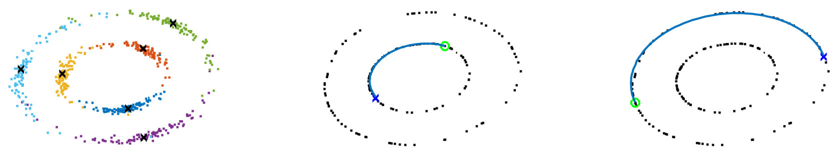


Figure C.6: Magnitude of the operators after training in the 2D concentric circle experiment.



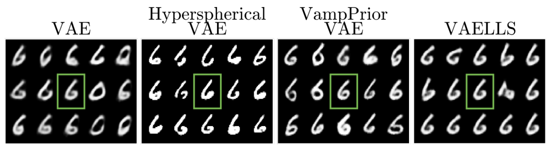
(a) (b) (c)

Figure C.7: Visualization of the components of the VAELLS prior on the concentric circle dataset. (a) Latent points sampled from the anchor points showing the concentric circle structure learned by the VAELLS model. (b-c) Transport operator paths inferred between points on the same concentric circle manifold.

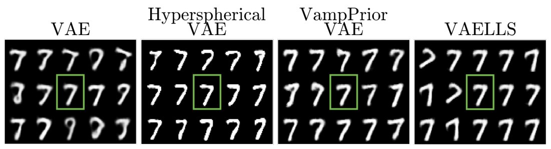
digits, we randomly select points from the MNIST training set and rotate those images to a random angle between 0 and 350 degrees. Separate anchor points are selected for each training example. Anchor points are generated by rotating the original MNIST sample by angles that are evenly spaced between 0 and 360 degrees. Because we have separate anchor points for each training sample, they are not updated during training. Table C.4 and Table C.5 contain the VAELLS network architecture and parameters for the rotated MNIST experiment. As described in appendix section C.2, in this experiment, we define the prior only with respect to the anchor points that are estimated to be closest to each training point. Figure C.8 shows more examples of images decoded from latent vectors sampled from the posterior of the models trained on rotated MNIST digits.

Table C.4: VAELLS network architecture for MNIST experiment

Encoder Network	Decoder Network
Input $\in \mathbb{R}^{28\times 28}$	Input $\in \mathbb{R}^{1}0$
conv: chan: 64 , kern: 4, stride: 2, pad: 1	Linear: 3136 Units
ReLU	ReLU
conv: chan: 64, kern: 4, stride: 2, pad: 1	convTranpose: chan: 64, kern: 4, stride: 1, pad: 1
ReLU	ReLU
conv: chan: 64, kern: 4, stride: 1, pad: 0	convTranpose: chann: 64, kern: 4, stride: 2, pad: 2
ReLU	ReLU
Linear: 10 Units	convTranpose: chan: 1, kernel: 4, stride: 2, pad: 1
	Sigmoid



(a)



(b)

Figure C.8: Examples of images decoded from latent vectors sampled from the posterior of models trained on rotated MNIST digits. In each example, the center digit (in the green box) is the decoded version of the input digit and the surrounding digits are images decoded from the sampled latent vectors. Sampling in the VAELLS latent space results in rotations in the sampled outputs.

C.9 Natural MNIST Experiment

This experiment uses the same training/validation/testing separation as described in the rotated MNIST experiment in section C.8. The only pre-processing step for the digit images is scaling by 255. Our qualitative results use a network that is trained with eight anchor points per digit class. These anchor points are initialized by randomly sampling eight examples of each class at the beginning of training. The anchor points are allowed to update during training but the changes in the anchor points during training are negligible. We use the same network architecture as in the rotated MNIST experiment (shown in Table C.4). Table C.6 shows the training parameters for this experiment. As described in appendix section C.2, in this experiment, we define the prior only with respect to the anchor points that are estimated to be closest to each training point.

Figure C.9 shows more examples of images decoded from latent vectors sampled from

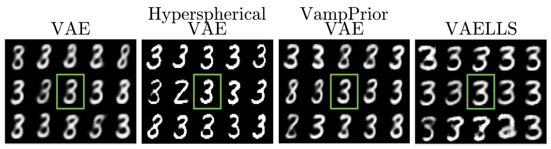
Table C.5: VAELLS training parameters for rotated MNIST experiment

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

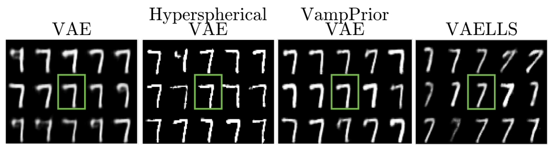
the posterior of the models trained on MNIST digits. Figure C.10 and Figure C.11 show the effect that each of the eight learned transport operators has on digits. To generate each figure, input images randomly selected from each class are encoded into the latent space and a single learned operator is applied to each of those latent vectors. The decoded version of the input image is shown in the middle column (in a green box). The images to the left of the middle column show the result of applying the operator with a negative coefficient and the images to the right of the middle column show the result of applying the operator with a positive coefficient.

Table C.6: VAELLS training parameters for natural MNIST experiment

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

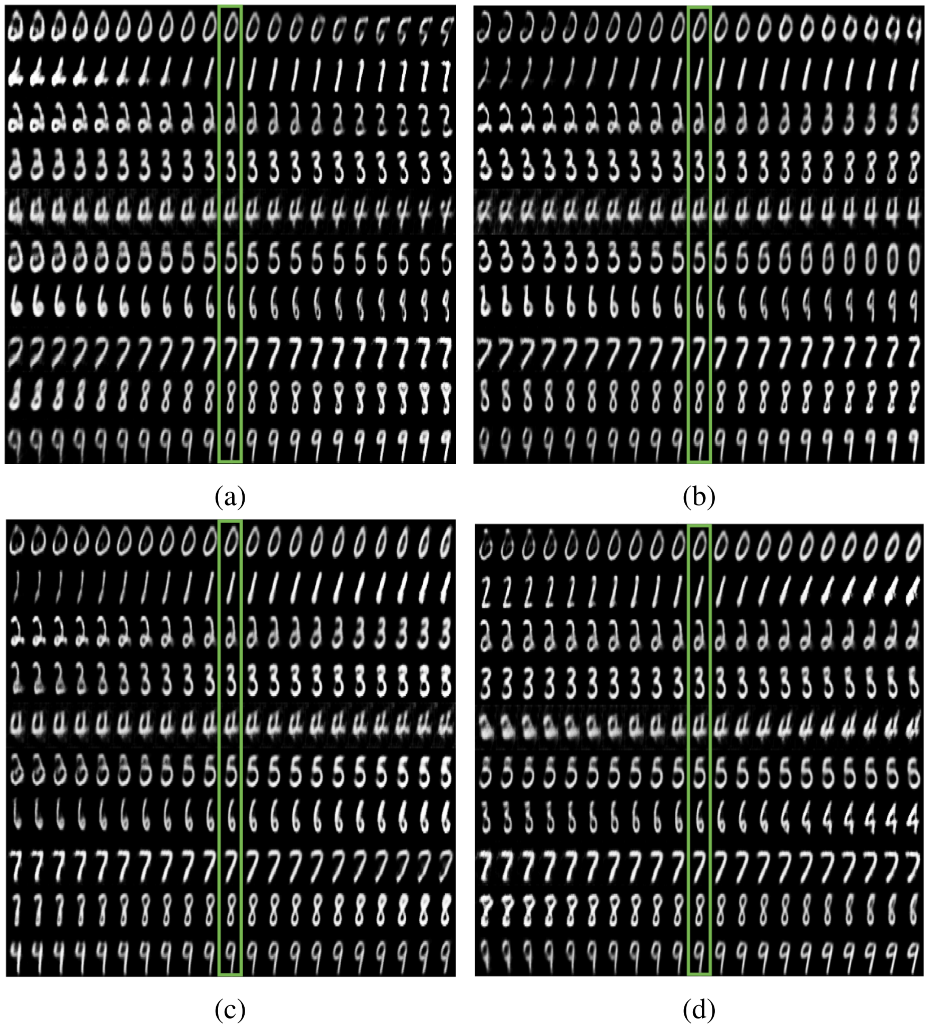


(a)



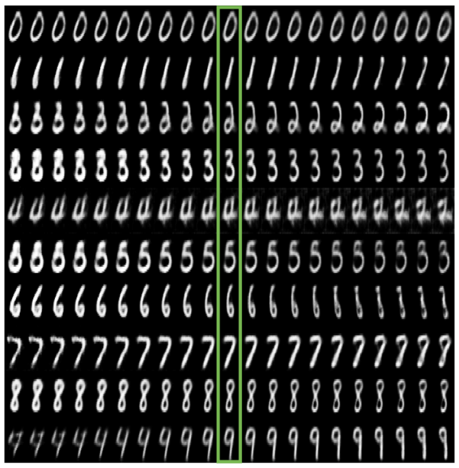
(b)

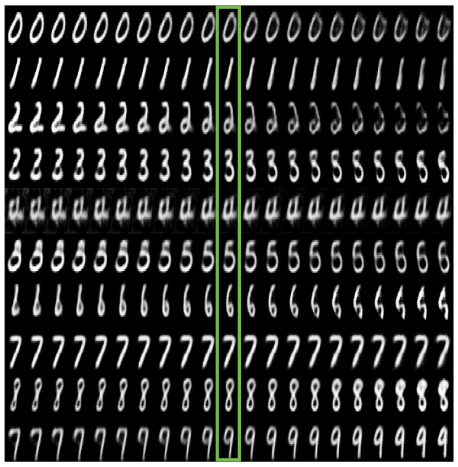
Figure C.9: Examples of images decoded from latent vectors sampled from the posterior of models trained on natural MNIST digits. In each example, the center digit (in the green box) is the decoded version of the input digit and the surrounding digits are images decoded from the sampled latent vectors.

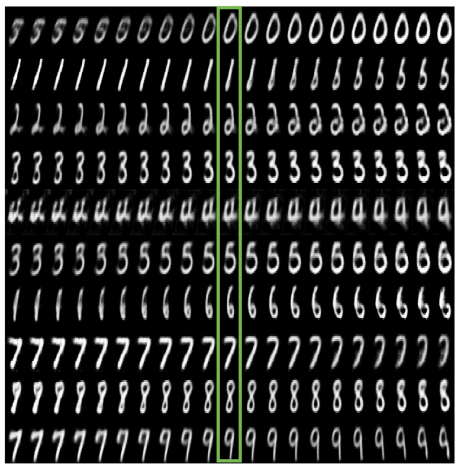


(a) (b) (c) (d)

Figure C.10: Extrapolated paths using the first four transport operators learned on natural MNIST digit variations.


(a)


(b)


(c)

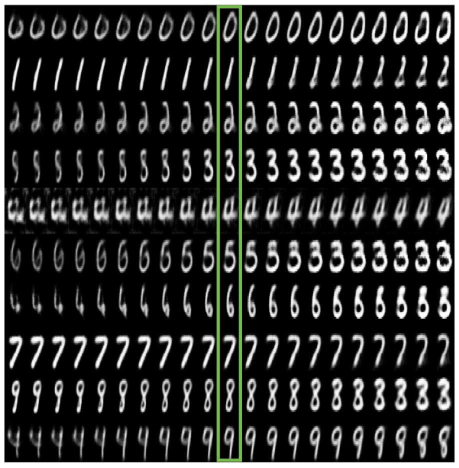

(d)

Figure C.11: Extrapolated paths using the second four transport operators learned on natural MNIST digit variations.

REFERENCES

[1] M. Connor and C. Rozell, “Representing closed transformation paths in encoded network latent space,” in *AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 3666–3675.

[2] M. Connor, K. Fallah, and C. Rozell, *Learning identity-preserving transformations on data manifolds*, 2021. arXiv: 2106.12096 `[cs.LG]`.

[3] M. Connor, G. Canal, and C. Rozell, “Variational autoencoder with learned latent structure,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2021, pp. 2359–2367.

[4] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference for Learning Representation*, 2054.

[5] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 6105–6114.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *International Conference for Learning Representation*, 2021.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Ieee, 2009, pp. 248–255.

[8] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 843–852.

[9] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, “Scaling vision transformers,” *arXiv preprint arXiv:2106.04560*, 2021.

[10] S. Gurumurthy, R. K. Sarvadevabhatla, and R. V. Babu, “Deligan: Generative adversarial networks for diverse and limited data.,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4941–4949.

[11] S. Hauberg, O. Freifeld, A. B. L. Larsen, J. Fisher, and L. Hansen, “Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data

augmentation," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2016, pp. 342–350.

[12] B. Liu, X. Wang, M. Dixit, R. Kwitt, and N. Vasconcelos, "Feature space transfer for data augmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2018.

[13] C. Fefferman, S. Mitter, and H. Narayanan, "Testing the manifold hypothesis," *Journal of the American Mathematical Society*, vol. 29, no. 4, pp. 983–1049, 2016.

[14] S. Rifai, Y. N. Dauphin, P. Vincent, Y. Bengio, and X. Muller, "The manifold tangent classifier," *Advances in Neural Information Processing Systems*, vol. 24, pp. 2294–2302, 2011.

[15] R. N. Shepard and L. A. Cooper, *Mental images and their transformations.* The MIT Press, 1986.

[16] C. Lamm, C. Windischberger, E. Moser, and H. Bauer, "The functional role of dorso-lateral premotor cortex during mental rotation: An event-related fmri study separating cognitive processing steps using a novel task paradigm," *NeuroImage*, vol. 36, no. 4, pp. 1374–1386, 2007.

[17] R. N. Shepard and J. Metzler, "Mental rotation of three-dimensional objects," *Science*, vol. 171, no. 3972, pp. 701–703, 1971.

[18] L. A. Cooper and R. N. Shepard, "Chronometric studies of the rotation of mental images," in *Visual information processing*, Elsevier, 1973, pp. 75–176.

[19] M. A. Just and P. A. Carpenter, "Cognitive coordinate systems: Accounts of mental rotation and individual differences in spatial ability.," *Psychological review*, vol. 92, no. 2, p. 137, 1985.

[20] J. J. DiCarlo and D. D. Cox, "Untangling invariant object recognition," *Trends in cognitive sciences*, vol. 11, no. 8, pp. 333–341, 2007.

[21] J. J. DiCarlo, D. Zoccolan, and N. C. Rust, "How does the brain solve visual object recognition?" *Neuron*, vol. 73, no. 3, pp. 415–434, 2012.

[22] E. J. Ward, L. Isik, and M. M. Chun, "General transformations of object representations in human visual cortex," *Journal of Neuroscience*, vol. 38, no. 40, pp. 8526–8537, 2018.

[23] B. J. Culpepper and B. A. Olshausen, "Learning transport operators for image manifolds.," in *Advances in Neural Information Processing Systems*, 2009, pp. 423–431.

[24] R. P. Rao and D. L. Ruderman, "Learning lie groups for invariant visual perception," in *Advances in Neural Information Processing Systems*, 1999, pp. 810–816.

[25] X. Miao and R. P. Rao, "Learning the lie groups of visual invariance," *Neural computation*, vol. 19, no. 10, pp. 2665–2693, 2007.

[26] J. Sohl-Dickstein, C. M. Wang, and B. A. Olshausen, "An unsupervised algorithm for learning lie group transformations," *arXiv preprint arXiv:1001.1027*, 2010.

[27] T. Cohen and M. Welling, "Learning the irreducible representations of commutative lie groups," in *International Conference on Machine Learning*, PMLR, 2014, pp. 1755–1763.

[28] W. C. Hoffman, "The lie algebra of visual perception," *Journal of mathematical Psychology*, vol. 3, no. 1, pp. 65–98, 1966.

[29] P. C. Dodwell, "The lie transformation group model of visual perception," *Perception & Psychophysics*, vol. 34, no. 1, pp. 1–16, 1983.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2013.

[31] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International Conference on Machine Learning*, PMLR, 2014, pp. 1278–1286.

[32] W. M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Academic press, 1986, vol. 120.

[33] G. Carlsson, T. Ishkhanov, V. De Silva, and A. Zomorodian, "On the local behavior of spaces of natural images," *International journal of computer vision*, vol. 76, no. 1, pp. 1–12, 2008.

[34] J. A. Gallego, M. G. Perich, L. E. Miller, and S. A. Solla, "Neural manifolds for the control of movement," *Neuron*, vol. 94, no. 5, pp. 978–984, 2017.

[35] R. J. Low, S. Lewallen, D. Aronov, R. Nevers, and D. W. Tank, "Probing variability in a cognitive map using manifold inference from neural dynamics," *BioRxiv*, p. 418 939, 2018.

[36] A. S. Charles, B. A. Olshausen, and C. J. Rozell, "Learning sparse codes for hyperspectral imagery," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 963–978, 2011.

[37] S. Kadoury, W. Mandel, M. Roy-Beaudry, M.-L. Nault, and S. Parent, “3-d morphology prediction of progressive spinal deformities from probabilistic modeling of discriminant manifolds,” *IEEE transactions on medical imaging*, vol. 36, no. 5, pp. 1194–1204, 2017.

[38] K. Pearson, “Liii. on lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[39] H. Hotelling, “Analysis of a complex of statistical variables into principal components.,” *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[40] P. Comon, “Independent component analysis, a new concept?” *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[41] J. B. Tenenbaum, V. De Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[42] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[43] K. Q. Weinberger and L. K. Saul, “An introduction to nonlinear dimensionality reduction by maximum variance unfolding,” in *AAAI Conference on Artificial Intelligence*, vol. 6, 2006, pp. 1683–1686.

[44] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[45] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[46] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” *arXiv preprint arXiv:1802.03426*, 2018.

[47] P. Dollár, V. Rabaud, and S. Belongie, “Non-isometric manifold learning: Analysis and an algorithm,” in *International Conference on Machine Learning*, ACM, 2007, pp. 241–248.

[48] P. Dollár, V. Rabaud, and S. J. Belongie, “Learning to traverse image manifolds,” in *Advances in Neural Information Processing Systems*, 2007, pp. 361–368.

[49] Y. Bengio and M. Monperrus, “Non-local manifold tangent learning,” *Advances in Neural Information Processing Systems*, vol. 17, pp. 129–136, 2005.

[50] M. Park, W. Jitkrittum, A. Qamar, Z. Szabó, L. Buesing, and M. Sahani, “Bayesian manifold learning: The locally linear latent variable model (ll-lvm),” in *Advances in Neural Information Processing Systems*, 2015, pp. 154–162.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[52] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the devil in the details: Delving deep into convolutional nets,” in *Proceedings of the British Machine Vision Conference*, BMVA Press, 2014.

[53] A. J. Ratner, H. Ehrenberg, Z. Hussain, J. Dunnmon, and C. Ré, “Learning to compose domain-specific transformations for data augmentation,” in *Advances in Neural Information Processing Systems*, 2017, pp. 3236–3246.

[54] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation strategies from data,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 113–123.

[55] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel, “Population based augmentation: Efficient learning of augmentation policy schedules,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 2731–2741.

[56] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, “Fast autoaugment,” *Advances in Neural Information Processing Systems*, 2019.

[57] G. Rogez and C. Schmid, “Mocap-guided data augmentation for 3d pose estimation in the wild,” in *Advances in Neural Information Processing Systems*, 2016, pp. 3108–3116.

[58] E. G. Miller, N. E. Matsakis, and P. A. Viola, “Learning from one example through shared densities on transforms,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, vol. 1, 2000, pp. 464–471.

[59] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond empirical risk minimization,” in *International Conference on Learning Representations*, 2018.

[60] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a few examples: A survey on few-shot learning,” *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[61] B. Hariharan and R. Girshick, “Low-shot visual recognition by shrinking and hallucinating features,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3018–3027.

[62] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, “Deep Visual Analogy-Making,” in *Advances in Neural Information Processing Systems 28*, 2015, pp. 1252–1260.

[63] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, A. Kumar, R. Feris, R. Giryes, and A. Bronstein, “Delta-encoder: An effective sample synthesis method for few-shot object recognition,” in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[64] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, “Spatial transformer networks,” in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[65] C.-H. Lin and S. Lucey, “Inverse compositional spatial transformer networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2568–2576.

[66] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, “Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision,” in *Advances in Neural Information Processing Systems*, 2016, pp. 1704–1712.

[67] A. Bas, P. Huber, W. A. Smith, M. Awais, and J. Kittler, “3d morphable models as spatial transformer networks,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 904–912.

[68] N. S. Detlefsen, O. Freifeld, and S. Hauberg, “Deep diffeomorphic transformer networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4403–4412.

[69] K. S. Tai, P. Bailis, and G. Valiant, “Equivariant transformer networks,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 6086–6095.

[70] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 764–773.

[71] T. Cohen and M. Welling, “Group equivariant convolutional networks,” in *International Conference on Machine Learning*, PMLR, 2016, pp. 2990–2999.

[72] S. Dieleman, J. De Fauw, and K. Kavukcuoglu, “Exploiting cyclic symmetry in convolutional neural networks,” in *International Conference on Machine Learning*, PMLR, 2016, pp. 1889–1898.

[73] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Harmonic networks: Deep translation and rotation equivariance," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5028–5037.

[74] D. Marcos, M. Volpi, N. Komodakis, and D. Tuia, "Rotation equivariant vector field networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5048–5057.

[75] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical cnns," in *International Conference on Learning Representations*, 2018.

[76] T. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, "Gauge equivariant convolutional networks and the icosahedral cnn," in *International Conference on Machine Learning*, PMLR, 2019, pp. 1321–1330.

[77] E. J. Bekkers, "B-spline cnns on lie groups," in *International Conference on Learning Representations*, 2019.

[78] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4, pp. 291–294, 1988.

[79] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length, and helmholtz free energy," *Advances in Neural Information Processing Systems*, vol. 6, pp. 3–10, 1994.

[80] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[81] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak, "Hyperspherical variational auto-encoders," *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)*, 2018.

[82] L. A. P. Rey, V. Menkovski, and J. W. Portegies, "Diffusion variational autoencoders," *arXiv preprint arXiv:1901.08991*, 2019.

[83] L. Falorsi, P. de Haan, T. R. Davidson, and P. Forré, "Reparameterizing distributions on lie groups," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2019, pp. 3244–3253.

[84] E. Mathieu, C. Le Lan, C. J. Maddison, R. Tomioka, and Y. W. Teh, "Continuous hierarchical representations with poincaré variational auto-encoders," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[85] Y. Nagano, S. Yamaguchi, Y. Fujita, and M. Koyama, "A wrapped normal distribution on hyperbolic space for gradient-based learning," in *International Conference on Machine Learning*, 2019, pp. 4693–4702.

[86] O. Skopek, O.-E. Ganea, and G. Bécigneul, "Mixed-curvature variational autoencoders," in *International Conference for Learning Representation*, 2020.

[87] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *International Conference on Machine Learning*, 2011.

[88] S. Rifai, G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot, "Higher order contractive auto-encoder," in *Joint European conference on machine learning and knowledge discovery in databases*, Springer, 2011, pp. 645–660.

[89] A. Kumar, P. Sattigeri, and P. T. Fletcher, "Semi-supervised learning with gans: Manifold invariance with improved inference," *Advances in Neural Information Processing Systems*, 2017.

[90] G. Arvanitidis, L. K. Hansen, and S. Hauberg, "Latent space oddity: On the curvature of deep generative models," in *International Conference on Learning Representations*, 2018.

[91] N. Chen, A. Klushyn, R. Kurle, X. Jiang, J. Bayer, and P. Smagt, "Metrics for deep generative models," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2018, pp. 1540–1550.

[92] H. Shao, A. 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*, 2018, pp. 315–323.

[93] T. Yang, G. Arvanitidis, D. Fu, X. Li, and S. Hauberg, "Geodesic clustering in deep generative models," *arXiv preprint arXiv:1809.04747*, 2018.

[94] G. Arvanitidis, S. Hauberg, P. Hennig, and M. Schober, "Fast and robust shortest paths on manifolds learned from data," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2019, pp. 1506–1515.

[95] G. Arvanitidis, B. Georgiev, and B. Schölkopf, "A prior-based approximate latent riemannian metric," *arXiv preprint arXiv:2103.05290*, 2021.

[96] D. Kalatzis, D. Eklund, G. Arvanitidis, and S. Hauberg, "Variational autoencoders with riemannian brownian motion priors," in *International Conference on Machine Learning*, 2020.

[97] S. Reichelt, R. Häussler, G. Fütterer, and N. Leister, "Depth cues in human visual perception and their realization in 3d displays," in *Three-Dimensional Imaging, Visualization, and Display 2010 and Display Technologies and Applications for Defense, Security, and Avionics IV*, International Society for Optics and Photonics, vol. 7690, 2010, 76900B.

[98] J. T. Petersik, "Three-dimensional object constancy: Coherence of a simulated rotating sphere in noise," *Perception & psychophysics*, vol. 25, no. 4, pp. 328–335, 1979.

[99] M. L. Braunstein, D. D. Hoffman, L. R. Shapiro, G. J. Andersen, and B. M. Bennett, "Minimum points and views for the recovery of three-dimensional structure.," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 13, no. 3, p. 335, 1987.

[100] J. T. Todd, R. A. Akerstrom, F. D. Reichel, and W. Hayes, "Apparent rotation in three-dimensional space: Effects of temporal, spatial, and structural factors," *Perception & Psychophysics*, vol. 43, no. 2, pp. 179–188, 1988.

[101] B. A. Dosher, M. S. Landy, and G. Sperling, "Ratings of kinetic depth in multidot displays.," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 15, no. 4, p. 816, 1989.

[102] G. Sperling, M. S. Landy, B. A. Dosher, and M. E. Perkins, "Kinetic depth effect and identification of shape.," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 15, no. 4, p. 826, 1989.

[103] M. L. Braunstein, *Depth perception through motion*. Academic Press, 2014.

[104] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[105] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[106] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: A factorization method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[107] D. Nistér, "Preemptive ransac for live structure and motion estimation," *Machine Vision and Applications*, vol. 16, no. 5, pp. 321–329, 2005.

[108] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, *et al.*, "Detailed real-time urban 3d re-

construction from video," *International Journal of Computer Vision*, vol. 78, no. 2, pp. 143–167, 2008.

[109] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *ACM SIGGRAPH 2006 Papers*, 2006, pp. 835–846.

[110] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 270–279.

[111] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European conference on computer vision*, Springer, 2016, pp. 740–756.

[112] J. Xie, R. Girshick, and A. Farhadi, "Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks," in *European conference on computer vision*, Springer, 2016, pp. 842–857.

[113] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017, p. 7.

[114] J. M. Zacks and P. Michelon, "Transformations of visuospatial images," *Behavioral and cognitive neuroscience reviews*, vol. 4, no. 2, pp. 96–118, 2005.

[115] S. P. Johnson, "Development of visual perception," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 2, no. 5, pp. 515–528, 2011.

[116] D. D. Hoffman, M. Singh, and C. Prakash, "The interface theory of perception," *Psychonomic bulletin & review*, vol. 22, no. 6, pp. 1480–1506, 2015.

[117] B. L. Anderson, "Can computational goals inform theories of vision?" *Topics in Cognitive Science*, vol. 7, no. 2, pp. 274–286, 2015.

[118] G. Perry, E. Rolls, and S. Stringer, "Continuous transformation learning of translation invariant representations," *Experimental brain research*, vol. 204, no. 2, pp. 255–270, 2010.

[119] N. Li and J. J. DiCarlo, "Unsupervised natural experience rapidly alters invariant object representation in visual cortex," *Science*, vol. 321, no. 5895, pp. 1502–1507, 2008.

[120] ——, "Unsupervised natural visual experience rapidly reshapes size-invariant object representation in inferior temporal cortex," *Neuron*, vol. 67, no. 6, pp. 1062–1075, 2010.

[121] H. Wallach and D. O'connell, "The kinetic depth effect.," *Journal of experimental psychology*, vol. 45, no. 4, p. 205, 1953.

[122] S. Ullman, "Maximizing rigidity: The incremental recovery of 3-d structure from rigid and nonrigid motion," *Perception*, vol. 13, no. 3, pp. 255–274, 1984.

[123] R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces," *IEEE Transactions on pattern analysis and machine intelligence*, no. 1, pp. 13–27, 1984.

[124] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 6, pp. 580–593, 1997.

[125] R. I. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[126] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second. Cambridge University Press, ISBN: 0521540518, 2004.

[127] T. Kanade and D. D. Morris, "Factorization methods for structure from motion," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 356, no. 1740, pp. 1153–1173, 1998.

[128] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Proceedings of the IEEE International Conference on Computer Vision*, IEEE, 2011, pp. 2320–2327.

[129] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *arXiv preprint arXiv:1406.2283*, 2014.

[130] L. Ladicky, J. Shi, and M. Pollefeys, "Pulling things out of perspective," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 89–96.

[131] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2024–2039, 2015.

[132] M. Kozhevnikov and M. Hegarty, "A dissociation between object manipulation spatial ability and spatial orientation ability," *Memory & cognition*, vol. 29, no. 5, pp. 745–756, 2001.

[133] J. Money, D. Alexander, and H. Walker, *A standardized road-map test of direction sense: Manual*. Johns Hopkins Press, 1965.

[134] J. M. Zacks, J. Mires, B. Tversky, and E. Hazeltine, "Mental spatial transformations of objects and perspective," *Spatial Cognition and Computation*, vol. 2, no. 4, pp. 315–332, 2000.

[135] M. Hegarty and D. Waller, "A dissociation between mental rotation and perspective-taking spatial abilities," *Intelligence*, vol. 32, no. 2, pp. 175–191, 2004.

[136] L. A. Cooperau and R. N. Shepard, "The time required to prepare for a rotated stimulus," *Memory & cognition*, vol. 1, no. 3, pp. 246–250, 1973.

[137] R. B. Ekstrom and H. H. Harman, *Manual for kit of factor-referenced cognitive tests, 1976*. Educational testing service, 1976.

[138] L. L. Thurstone and T. G. Thurstone, "Factorial studies of intelligence.," *Psychometric monographs*, 1941.

[139] S. G. Vandenberg and A. R. Kuse, "Mental rotations, a group test of three-dimensional spatial visualization," *Perceptual and motor skills*, vol. 47, no. 2, pp. 599–604, 1978.

[140] M.-A. Amorim, B. Isableu, and M. Jarraya, "Embodied spatial transformations:" body analogy" for the mental rotation of objects.," *Journal of Experimental Psychology: General*, vol. 135, no. 3, p. 327, 2006.

[141] L. A. Cooper, "Mental rotation of random two-dimensional shapes," *Cognitive psychology*, vol. 7, no. 1, pp. 20–43, 1975.

[142] ——, "Demonstration of a mental analog of an external rotation," *Perception & Psychophysics*, vol. 19, no. 4, pp. 296–302, 1976.

[143] M. C. Corballis and R. McLaren, "Interaction between perceived and imagined rotation.," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 8, no. 2, p. 215, 1982.

[144] M. Wexler, S. M. Kosslyn, and A. Berthoz, "Motor processes in mental rotation," *Cognition*, vol. 68, no. 1, pp. 77–94, 1998.

[145] A. Wohlschläger, "Mental object rotation and the planning of hand movements," *Perception & psychophysics*, vol. 63, no. 4, pp. 709–718, 2001.

[146] M. van Elk and O. Blanke, "Imagined own-body transformations during passive self-motion," *Psychological research*, vol. 78, no. 1, pp. 18–27, 2014.

[147] Z. W. Pylyshyn, "What the mind's eye tells the mind's brain: A critique of mental imagery.," *Psychological bulletin*, vol. 80, no. 1, p. 1, 1973.

[148] S. M. Kosslyn, *Image and brain: The resolution of the imagery debate*. MIT press, 1996.

[149] S. M. Kosslyn and J. R. Pomerantz, “Imagery, propositions, and the form of internal representations,” *Cognitive psychology*, vol. 9, no. 1, pp. 52–76, 1977.

[150] S. M. Kosslyn, *Image and mind*. Harvard University Press, 1980.

[151] S. M. Kosslyn, “The medium and the message in mental imagery: A theory.,” *Psychological review*, vol. 88, no. 1, p. 46, 1981.

[152] Z. W. Pylyshyn, “The imagery debate: Analogue media versus tacit knowledge.,” *Psychological review*, vol. 88, no. 1, p. 16, 1981.

[153] R. A. Finke, “Theories relating mental imagery to perception.,” *Psychological bulletin*, vol. 98, no. 2, p. 236, 1985.

[154] R. A. Finke and R. N. Shepard, “Visual functions of mental imagery.,” 1986.

[155] Z. W. Pylyshyn, “Mental imagery: In search of a theory,” *Behavioral and brain sciences*, vol. 25, no. 2, pp. 157–182, 2002.

[156] Z. Pylyshyn, “Return of the mental image: Are there really pictures in the brain?” *Trends in cognitive sciences*, vol. 7, no. 3, pp. 113–118, 2003.

[157] A. Larsen, “Deconstructing mental rotation.,” *Journal of experimental psychology: Human perception and performance*, vol. 40, no. 3, p. 1072, 2014.

[158] M. Wraga, S. H. Creem, and D. R. Proffitt, “The influence of spatial reference frames on imagined object-and viewer rotations,” *Acta Psychologica*, vol. 102, no. 2-3, pp. 247–264, 1999.

[159] ——, “Updating displays after imagined object and viewer rotations.,” *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 26, no. 1, p. 151, 2000.

[160] M. Keehner, S. A. Guerin, M. B. Miller, D. J. Turk, and M. Hegarty, “Modulation of neural activity by angle of rotation during imagined spatial transformations,” *Neuroimage*, vol. 33, no. 1, pp. 391–398, 2006.

[161] E. S. Barratt, “An analysis of verbal reports of solving spatial problems as an aid in defining spatial factors,” *The Journal of Psychology*, vol. 36, no. 1, pp. 17–25, 1953.

[162] K. Schultz, “The contribution of solution strategy to spatial performance.,” *Canadian Journal of Psychology/Revue canadienne de psychologie*, vol. 45, no. 4, p. 474, 1991.

[163] B. V. Funt, “A parallel-process model of mental rotation,” *Cognitive Science*, vol. 7, no. 1, pp. 67–93, 1983.

[164] M. Fukumi, S. Omatu, and Y. Nishikawa, “Rotation-invariant neural pattern recognition system estimating a rotation angle,” *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 568–581, 1997.

[165] T. Inui and M. Ashizawa, “Temporo-parietal network model for 3d mental rotation,” in *Advances in cognitive neurodynamics (II)*, Springer, 2011, pp. 91–95.

[166] K. Seepanomwan, D. Caligiore, A. Cangelosi, and G. Baldassarre, “Generalisation, decision making, and embodiment effects in mental rotation: A neurorobotic architecture tested with a humanoid robot,” *Neural Networks*, vol. 72, pp. 31–47, 2015.

[167] L. Wiskott and T. J. Sejnowski, “Slow feature analysis: Unsupervised learning of invariances,” *Neural computation*, vol. 14, no. 4, pp. 715–770, 2002.

[168] B. Hall, *Lie groups, Lie algebras, and representations: an elementary introduction*. Springer, 2015, vol. 222.

[169] E. C. Hildreth, N. M. Grzywacz, E. H. Adelson, and V. K. Inada, “The perceptual buildup of three-dimensional structure from motion,” *Perception & Psychophysics*, vol. 48, no. 1, pp. 19–36, 1990.

[170] M. G. Kendall, “A new measure of rank correlation,” *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[171] H. W. Kuhn, “The hungarian method for the assignment problem,” *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[172] R. A. Andersen and D. C. Bradley, “Perception of three-dimensional structure from motion,” *Trends in cognitive sciences*, vol. 2, no. 6, pp. 222–228, 1998.

[173] S. Ullman, “The interpretation of structure from motion,” *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979.

[174] J. Zbontar, Y. LeCun, *et al.*, “Stereo matching by training a convolutional neural network to compare image patches.,” *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2287–2318, 2016.

[175] W. Luo, A. G. Schwing, and R. Urtasun, “Efficient deep learning for stereo matching,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5695–5703.

[176] J. J. Hull, “A database for handwritten text recognition research,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 5, pp. 550–554, 1994.

[177] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[178] N. Aifanti, C. Papachristou, and A. Delopoulos, “The mug facial expression database,” in *Image analysis for multimedia interactive services (WIAMIS), 2010 11th international workshop on*, IEEE, 2010, pp. 1–4.

[179] T. Kanade, J. Cohn, and Y. Tian, “Comprehensive database for facial expression analysis,” in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, 2000, pp. 46–53.

[180] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, “The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression,” in *2010 ieee computer society conference on computer vision and pattern recognition workshops*, IEEE, 2010, pp. 94–101.

[181] D. E. King, “Dlib-ml: A machine learning toolkit,” *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[182] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” in *International Conference on Learning Representations*, 2018.

[183] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proceedings of International Conference on Computer Vision (ICCV)*, Dec. 2015.

[184] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” in *International Conference on Learning Representations*, 2016.

[185] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[186] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, vol. 1, 2005, pp. 539–546.

[187] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[188] X. Lin, H. Wang, Z. Li, Y. Zhang, A. Yuille, and T. S. Lee, “Transfer of view-manifold learning to similarity perception of novel objects,” in *International Conference on Learning Representations*, 2017.

[189] N. Frosst, N. Papernot, and G. Hinton, “Analyzing and improving representations with the soft nearest neighbor loss,” in *International Conference on Machine Learning*, PMLR, 2019, pp. 2012–2020.

[190] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International Conference on Machine Learning*, PMLR, 2016, pp. 478–487.

[191] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, “Towards k-means-friendly spaces: Simultaneous deep learning and clustering,” in *International Conference on Machine Learning*, PMLR, 2017, pp. 3861–3870.

[192] K. Han, A. Vedaldi, and A. Zisserman, “Learning to discover novel visual categories via deep transfer clustering,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8401–8409.

[193] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner, “Towards a definition of disentangled representations,” *arXiv preprint arXiv:1812.02230*, 2018.

[194] J. B. Tenenbaum and W. T. Freeman, “Separating style and content with bilinear models,” *Neural computation*, vol. 12, no. 6, pp. 1247–1283, 2000.

[195] S. Reed, K. Sohn, Y. Zhang, and H. Lee, “Learning to disentangle factors of variation with manifold interaction,” in *International Conference on Machine Learning*, PMLR, 2014, pp. 1431–1439.

[196] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun, “Disentangling factors of variation in deep representation using adversarial training,” in *Advances in Neural Information Processing Systems*, 2016, pp. 5040–5048.

[197] D. Bouchacourt, R. Tomioka, and S. Nowozin, “Multi-level variational autoencoder: Learning disentangled representations from grouped observations,” in *AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[198] B. Cheung, J. A. Livezey, A. K. Bansal, and B. A. Olshausen, “Discovering hidden factors of variation in deep networks,” in *International Conference on Learn-*

ing Representations, ICLR Workshop Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2015.

[199] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International Conference on Artificial Neural Networks*, Springer, 2011, pp. 44–51.

[200] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015.

[201] J. Yang, S. E. Reed, M.-H. Yang, and H. Lee, "Weakly-supervised disentangling with recurrent transformations for 3d view synthesis," *Advances in Neural Information Processing Systems*, vol. 28, pp. 1099–1107, 2015.

[202] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2016.

[203] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "Beta-vae: Learning basic visual concepts with a constrained variational framework.," *International Conference on Learning Representations*, vol. 2, no. 5, p. 6, 2017.

[204] H. Kim and A. Mnih, "Disentangling by factorising," in *International Conference on Machine Learning*, PMLR, 2018, pp. 2649–2658.

[205] A. Kumar, P. Sattigeri, and A. Balakrishnan, "Variational inference of disentangled latent concepts from unlabeled observations," in *International Conference on Learning Representations*, 2018.

[206] R. T. Chen, X. Li, R. Grosse, and D. Duvenaud, "Isolating sources of disentanglement in vaes," in *Advances in Neural Information Processing Systems*, 2018, pp. 2615–2625.

[207] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*, Springer, 2016, pp. 694–711.

[208] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *International Conference on Machine Learning Deep Learning Workshop*, 2015.

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

[210] R. Bergmann, “MVIRT, a toolbox for manifold-valued image restoration,” in *IEEE International Conference on Image Processing*, 2017, pp. 201–205.

[211] N. Chen, J. Bayer, S. Urban, and P. Van Der Smagt, “Efficient movement representation by embedding dynamic movement primitives in deep autoencoders,” in *International Conference on Humanoid Robots (Humanoids)*, IEEE, 2015, pp. 434–440.

[212] G. W. Taylor, G. E. Hinton, and S. T. Roweis, “Modeling human motion using binary latent variables,” in *Advances in Neural Information Processing Systems*, 2007, pp. 1345–1352.

[213] Q. Hu, A. Szabó, T. Portenier, P. Favaro, and M. Zwicker, “Disentangling factors of variation by mixing them,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3399–3407.

[214] J. Lin, Z. Chen, Y. Xia, S. Liu, T. Qin, and J. Luo, “Exploring explicit domain supervision for latent space disentanglement in unpaired image-to-image translation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 4, pp. 1254–1266, 2019.

[215] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[216] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[217] L. Mao, Y. Yan, J.-H. Xue, and H. Wang, “Deep multi-task multi-label cnn for effective facial attribute classification,” *IEEE Transactions on Affective Computing*, 2020.

[218] P. Bachman, O. Alsharif, and D. Precup, “Learning with pseudo-ensembles,” *Advances in neural information processing systems*, vol. 27, pp. 3365–3373, 2014.

[219] B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: A strategy employed by v1?” *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[220] J. Tomczak and M. Welling, "Vae with a vampprior," in *International Conference on Artificial Intelligence and Statistics*, A. Storkey and F. Perez-Cruz, Eds., vol. 84, PMLR, 2018, pp. 1214–1223.

[221] H. Li, O. Lindenbaum, X. Cheng, and A. Cloninger, "Variational diffusion autoencoders with random walk sampling," in *European Conference on Computer Vision*, Springer, 2020, pp. 362–378.

[222] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference for Learning Representations*, 2015.

[223] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance weighted autoencoders," in *International Conference on Learning Representations*, 2015.

[224] PACE, *Partnership for an Advanced Computing Environment (PACE)*, 2017.

[225] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, Jan. 2009.

[226] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014.

[227] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[228] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*, PMLR, 2015, pp. 1530–1538.

[229] M. Gil, "On rényi divergence measures for continuous alphabet sources," Ph.D. dissertation, Citeseer, 2011.