공학석사 학위논문

시연 학습을 위한 군 대칭 오토인코더

Group Symmetric Autoencoders for Learning from Demonstration

2023년 2월

서울대학교 대학원

기계공학부

손 민 준

시연 학습을 위한 군 대칭 오토인코더

Group Symmetric Autoencoders for Learning from Demonstration

지도교수 박 종 우

이 논문을 공학석사 학위논문으로 제출함

2022 년 10 월

서울대학교 대학원

기계공학부

손 민 준

손민준의 공학석사 학위논문을 인준함

2022 년 12 월

위 원 장 : _____ (인)

부위원장 : _____ (인)

위 원 : _____ (인)

ABSTRACT

Group Symmetric Autoencoders for Learning from Demonstration

by

Minjun Son

Department of Mechanical Engineering
Seoul National University

Learning from Demonstration(LfD) is a powerful motion-planning framework to resolve limitation of classical algorithm, such as optimization or sampling method. Without designing explicit cost function or sampling in high-dimensional space, motion planner can learn demonstration of human expert to generate accurate and human-like motions. Demonstration data is often given in the form of end-effector trajectory so that the data is often high-dimensional. To deal with the dimensionality, high-dimensional data are assumed to be embedded on low-dimensional manifold, which is called manifold hypothesis. Autoencoder is widely used deep

generative model to learn a data manifold. However, since demonstration is time-consuming, the number of training data is inevitably small. Lack of training data can cause over-fitting of neural network model and failure of generating accurate motion. Since translated or rotated trajectories are indeed identical, endowing translational or rotational symmetry to model can improve data efficiency and prevent over-fitting. In this paper, we first formulate the natural symmetry of robotic task as group action. We then propose group symmetric variational autoencoder (GSVAE) which has group-invariant encoder and group-equivariant decoder about symmetry group. We show that GSVAE can learn data manifold and generate motion more accurately than baseline model with water pouring dataset.

Keywords: Learning from Demonstration, manifold learning, representation learning, autoencoders, symmetry of task, group invariance/equivariance

Student Number: 2021-21058

Contents

List of Figures

1 Introduction

Motion planning is a process of determining a path from an initial state to a goal state while avoiding obstacles and satisfying certain constraints such as joint limits or torque limits [2]. Motion planning has been widely studied and is one of the most essential parts of robotics. Popular approaches for motion planning are (i) optimization-based algorithms [3] and (ii) sampling-based algorithms [4, 5]. Optimization-based algorithms find a trajectory that minimizes the cost function while satisfying some constraints. The design of cost function and constraints are crucial components in optimization, where cost function and constraints should be designed so that the optimal solution is unique and the optimization process is stable. However, designing cost functions and constraints which satisfy the above condition is not a trivial problem. On the other hand, sampling-based algorithms find a trajectory by sampling on configuration space or state space. Sufficient samples to cover high-dimensional configuration space or state space are required, however generating samples to cover high-dimensional space is time-consuming and intractable. Pouring water, pegging in a hole, and scooping are examples of robotic

tasks which are hard to implement with the aforementioned motion planning algorithms. Learning from demonstration (LfD) method is one of the most successful approaches to avoid these problems [6, 7, 8]. Rather than specifying cost function analytically or sampling in high-dimensional space, observations of a human expert (demonstrations) are used to train a motion planner and then the trained motion planner can generate accurate and human-like motions.

A Motion planner model for LfD should satisfy two properties: (i) generalizability and (ii) learning data manifold. (i) Just replaying the given human demonstration is not sufficient and the motion planner should be able to generate the motion which is not observed in demonstration data. This property is called generalizability of a model. (ii) Demonstration is often given as trajectory of end-effector and as a result, demonstration data is high-dimensional. We note that high-dimensional trajectory data do not occupy the whole trajectory space due to smoothness of trajectory and constraints of the given tasks. To find a low-dimensional representation of high-dimensional data, we adopt a manifold hypothesis [9, 10]. That is, high-dimensional data are embedded on a low-dimensional manifold. The motion planner model should be able to learn the manifold structure of trajectories and generate a trajectory from low-dimensional parameters.

In this paper, we divide the low-dimensional parameters into two categories: task parameter and manner parameter [11]. Let's start with pouring water example. In the pouring water example, a user aims to train the motion planner which generates the trajectory that pours water in a bottle to a cup. Water pouring trajectories are given by human experts and each trajectory is represented by some low-dimensional parameters. Some low-dimensional parameters should specify goals of the trajectories such as the position of the cup or the initial configuration of the bottle. These parameters are called task parameters. The rest of

the low-dimensional parameters indicates the information except the goal of the trajectory such as shape of the trajectory or speed of the trajectory. These parameters are called manner parameters. The model should be able to represent high-dimensional trajectory into task and manner parameters separately and generate a trajectory that executes the given task parameter accurately.

An autoencoder is a widely used deep generative model to generate a generalized trajectory and to learn the manifold structure of given human demonstration [11, 12]. Autoencoder consists of two neural networks: encoder and decoder. An encoder is a mapping from high-dimensional space to low-dimensional space and a decoder is a mapping from low-dimensional space to high-dimensional space. The low-dimensional space is called a latent space or coordinate space. Encoder and decoder are trained to minimize the reconstruction loss of the given training data and become a coordinate chart of the trained manifold. After training, the decoder can be used to generate a new trajectory. First, sample some low-dimensional parameters in the latent space and then decode the sampled parameters to high-dimensional trajectories.

Training deep generative models often requires a large amount of dataset. If the number of data is insufficient, the model is highly overfitted to the training dataset and the generalizability of the model is not guaranteed [13]. However, gathering demonstrations of human experts would be time-consuming and this prevents LfD framework from being applied to high-dimensional trajectory space. Autoencoders are often regularized to combine prior knowledge of tasks or to endow better inductive bias [10, 14, 15]. To leverage the difficulty in gathering demonstration, we focus on the natural symmetry contained in robotic tasks. For the pouring water example, there are translational symmetry on the planar position of a cup and rotational symmetry on the relative position of a bottle. If we consider the

natural symmetry of tasks, all symmetric trajectories from a single demonstration are known so that we can train the model with much fewer demonstrations. In this paper, we represent the natural symmetry of task by group action and then we formalize neural networks which are invariant and equivariant for the given group. Using these invariant and equivariant neural networks, we propose a Group Symmetric Varational Autoencoder (GSVAE) which has an invariant encoder and an equivariant decoder to learn a group symmetric representation of trajectory demonstration.

Various experimental results on water pouring dataset verify the performance of our model. The accuracy of manifold learning and generalizability of LfD models are compared by reconstruction error on training dataset and validation dataset. We have shown that our GSVAE model can learn the more accurate manifold with fewer neural network parameters and have better generalization capacity than a baseline model. In addition, trajectories which are generated from linearly interpolated latent variables are visualized to compare the generation power of LfD models. Our GSVAE model generates more smoothly varying trajectories by modulating manner parameters, that is GSVAE can learn more smooth manifolds than the baseline model. Lastly, robot experiments are implemented to compare the models in the real world situation. The success rates of the generated trajectories are compared and GSVAE shows a higher success rate.

The paper is structured in the following way. Chapter 2 contains mathematical preliminaries which are essential to formulate group symmetric variational autoencoder. Autoencoder models for manifold learning are explained and definitions for group action, invariance and equivariance are given to represent the natural symmetry of task. Chapter 3 constructs an invariant neural network and an equivariant neural network about a given symmetry group. Chapter 4 proposes group

symmetric autoencoder. Chapter 5 contains experiments for water pouring dataset. Group action of water pouring dataset is formulated in this chapter and the performances of models are proposed. Chapter 6 concludes the paper with a summary of contributions and experimental results.

2

Preliminaries

In this chapter, autoencoder models and group action are introduced. First, the structure and training process of vanilla autoencoder are introduced and other variants of vanilla autoencoder such as variational autoencoder (VAE) and task-conditioned autoencoder (TCVAE) are also introduced. The concepts of group action, invariance, and equivariance are formulated to represent the natural symmetry of tasks in mathematical form. Let's start with autoencoder models.

2.1 Autoencoders

2.1.1 Vanilla Autoencoders

Autoencoder is a neural network model to reconstruct high-dimensional data with a bottle-neck structure in the middle as shown in Figure 2.1. Autoencoder consists of two networks: an encoder network and a decoder network. The encoder

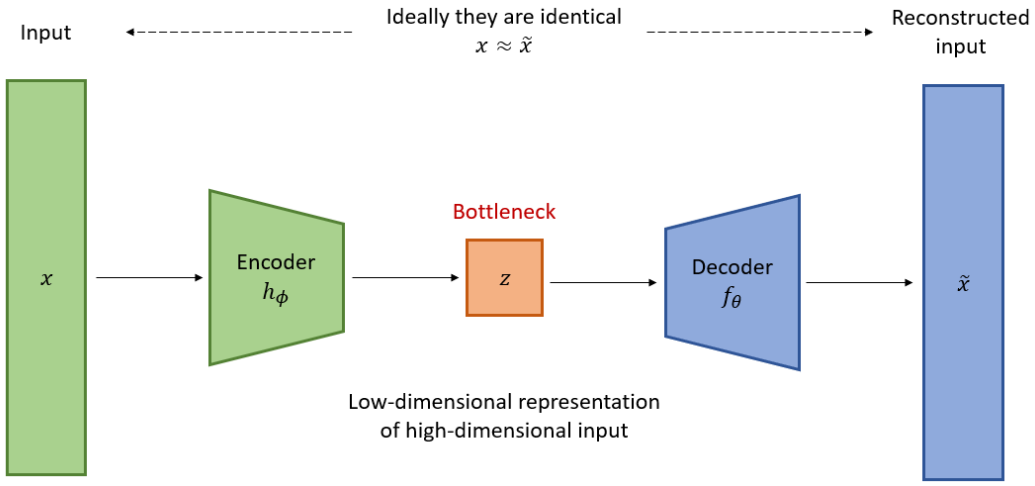

Figure 2.1: Structure of a vanilla autoencoder [1]. Encoder encodes a high-dimensional input x into a low-dimensional latent variable z and decoder recovers the latent variable back into the high-dimensional input $\tilde{x}$. The encoder and decoder are parameterized by neural network parameters θ, ϕ and trained to minimize the reconstruction error so that the original input x and the reconstructed input $\tilde{x}$ are identical.

network encodes the high-dimensional input into a low-dimensional latent variable. The low-dimensional latent variable is a "compressed code" for the high-dimensional input x which contains essential information for reconstruction and other non-necessary information is wiped out. The decoder network recovers the low-dimensional latent variable back into the high-dimensional data. The encoder function $g(\cdot)$ and the decoder function $f(\cdot)$ are parameterized by ϕ and θ. The compressed latent variable for input x is $z = g_\phi(x)$ and the reconstructed input is $\tilde{x} = f_\theta(z) = f_\theta(g_\phi(x))$. Neural network parameters ϕ and θ are trained to minimize the difference between the original input x and the reconstructed input $\tilde{x}$.

The difference measure for reconstruction is often selected as MSE loss:

$$\mathcal{L}(\theta, \phi) = \frac{1}{N} \sum_{i=1}^{N} (x - f_\theta(g_\phi(x_i)))^2 \tag{2.1.1}$$

where $\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$ is a training dataset.

2.1.2 Manifold Interpretation of Autoencoder

A manifold is a topological space that resembles Euclidean space locally. More specifically, an abstract manifold is a Hausdorff, second-countable topological space that is locally homeomorphic to Euclidean space of some finite dimension, known as the dimension of the manifold [16]. This is called the intrinsic definition of a manifold. Meanwhile, there is another way to define a manifold, which is called the extrinsic definition of a manifold. In the extrinsic view, a manifold is considered embedded in some high-dimensional Euclidean space. An $n-$dimensional manifold embedded in $\mathbb{R}^m$ $(n < m)$ is a subset of $\mathbb{R}^m$ such that each point in $\mathbb{R}^m$ has a neighborhood that is homeomorphic to an open subset of $\mathbb{R}^n$. When a manifold is viewed in the extrinsic view, it is easy to define concepts of tangent or normal using intuition from high-dimensional Euclidean space. Since an $n-$manifold locally resembles $\mathbb{R}^n$, there is an invertible mapping between the manifold and $\mathbb{R}^n$ and this mapping is called a local chart of the manifold. We can describe high-dimensional points on a manifold by n coordinates using a local chart. For example, the globe is a $2-$dimensional manifold embedded in $\mathbb{R}^3$ and a world map is a local chart of the globe.

In machine learning, there are various types of data such as images and video and these data are often high-dimensional. For example, a 200×200 sized image is represented by a $40,000$-dimensional vector and representing color video data requires an even higher-dimensional vector. If we randomly sample vectors in $\mathbb{R}^{40000}$

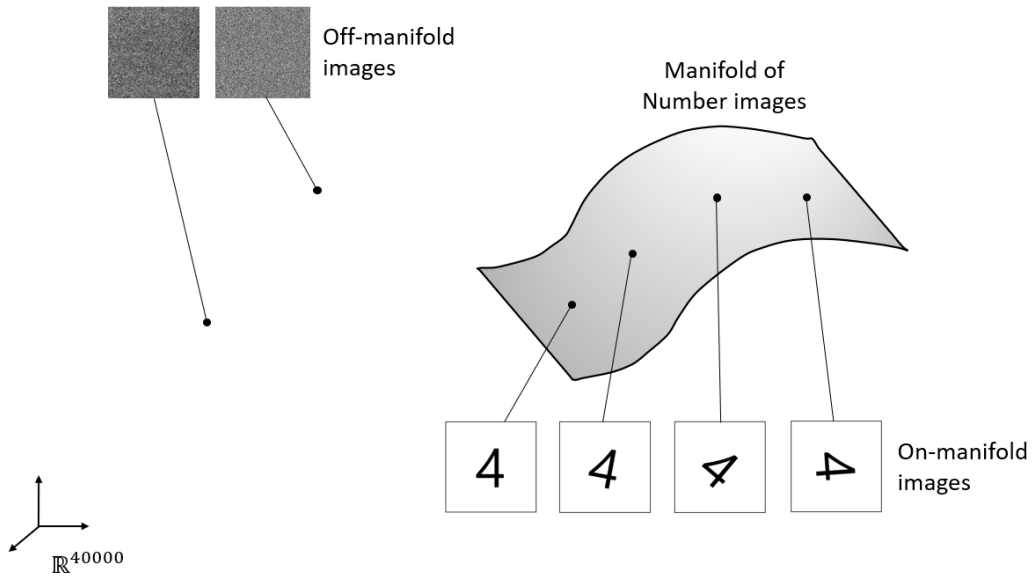

Figure 2.2: Number image data are on a low-dimensional manifold rather than occupying the whole high-dimensional image space. Samples on the low-dimensional manifold are normal number image data while samples off the manifold are noisy images.

and visualize these vectors, most of the vectors represent just noisy images. Thus, natural images such as number images or human face images are concentrated in a small region rather than occupying the whole high-dimensional image space, that is, natural images form a low-dimensional manifold embedded in high-dimensional space as shown in Figure 2.2. This is called the manifold hypothesis [9, 10]. If a model can learn a local chart of the data manifold, high-dimensional data can be represented using much fewer variables. Autoencoder is indeed an exact model to learn the manifold structure of high-dimensional data. In Figure 2.3, the image of decoder $\mathcal{M} = f_\theta(\mathbb{R}^n)$ is the learned data manifold and latent variable z is a low-dimensional coordinate of input x. The encoder and the decoder act like a local

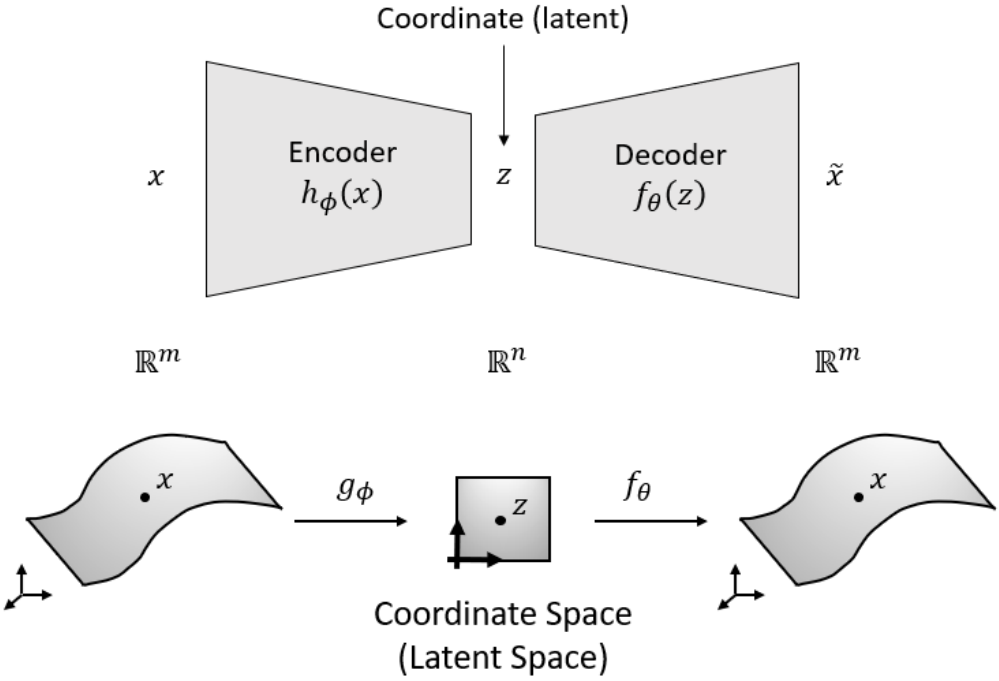

Figure 2.3: Autoencoder learns the manifold structure of high-dimensional dataset. Encoder and decoder become a local chart of the data manifold and latent variable become a coordinate. The trained data manifold is a image of latent space by decoder, that is $f_\theta(\mathbb{R}^n)$.

chart of the data manifold. By modulating latent variable z and mapping by the decoder, we can generate smoothly varying high-dimensional data.

2.1.3 Variational Autoencoder

Vanilla autoencoder maps the input into a fixed vector so that a distribution of dataset cannot be captured. Rather than mapping the input into a fixed vector, variational autoencoder (VAE) [12] maps a vector into a distribution. Let's assume

that the dataset $\mathcal{D} = \{x_1, x_2, \cdots, x_N\}$ is sampled by the following sampling process: (i) a latent variable z_i is sampled from some prior distribution $p(z) = \mathcal{N}(0, I)$, (ii) a data x_i is generated from some conditional distribution $p(x|z)$. We can model the conditional distribution $p(x|z)$ as a parameterized distribution $p_\theta(x|z)$.

In estimation theory, maximum likelihood estimation (MLE) or maximum a posteriori (MAP) estimation are commonly used to estimate the best parameter θ. For MLE and MAP, the marginal likelihood $p_\theta(x) = \int p_\theta(x|z)p(z)dz$ and the posterior density $p_\theta(z|x) = p_\theta(x|z)p(z)/p_\theta(x)$ should be evaluated but these evaluations are often intractable in case of the complicated likelihood functions $p_\theta(x|z)$ such as a deep neural network.

Rather than directly estimating the evidence of the posterior, we introduce a parametric model $q_\phi(z|x)$ to estimate the true posterior $p_\theta(z|x)$. The difference between the true posterior and the model posterior can be measured with Kullback-Leibler divergence (KL-divergence):

$$D_{KL}(q_\phi(z|x)||p_\theta(z|x)) = -\int q_\phi(z|x) \log \frac{p_\theta(z|x)}{q_\phi(z|x)} dz \tag{2.1.2}$$

If we expand the KL-divergence, we can obtain:

$$D_{KL}(q_\phi(z|x)||p_\theta(z|x)) = \log p_\theta(x) + D_{KL}(q_\phi(z|x)||p(z)) - \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) \tag{2.1.3}$$

By rearranging the equation, we can obtain the evidence lower bound (ELBO) for the marginal likelihood $p_\theta(x)$:

$$\log p_\theta(x) \geq \log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \tag{2.1.4}$$

$$= \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) - D_{KL}(q_\phi(z|x)||p(z)) \tag{2.1.5}$$

$$= \mathcal{L}_{ELBO}(\theta, \phi; x) \tag{2.1.6}$$

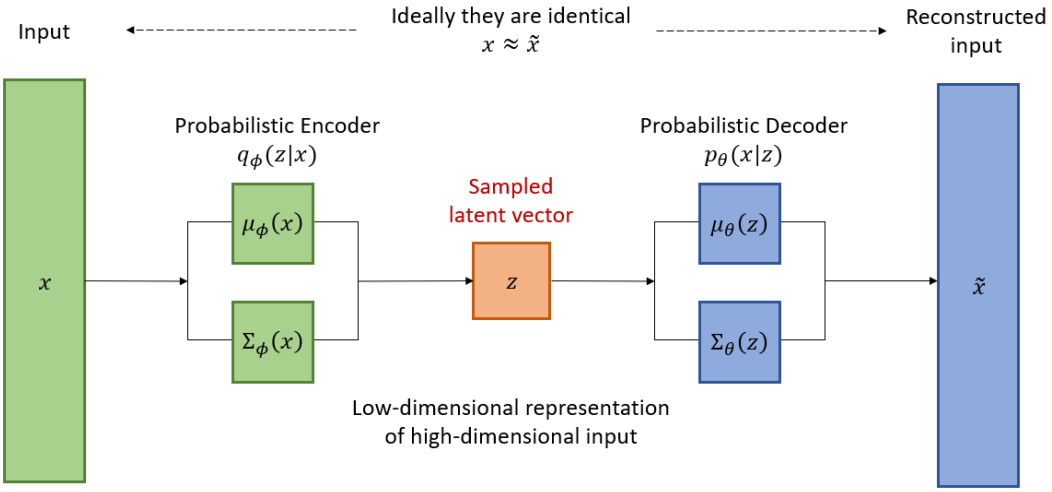

Figure 2.4: Structure of a variational autoencoder. Variational autoencoder encodes and decodes the high-dimensional input in a probabilistic manner. Latent variable z is sampled from a probabilistic encoder $q_\phi(z|x)$ given the input x and reconstructed input $\tilde{x}$ is sampled from a probabilistic decoder $p_\theta(x|z)$ given the latent z. Gaussian prior $p(z) = \mathcal{N}(0, I)$ is assumed and parameters θ, ϕ are trained with ELBO loss.

Maximizing $\mathcal{L}_{ELBO}$ about θ and ϕ is identical to maximizing the log-likelihood and to minimizing the KL-divergence between the model posterior and the true posterior. Thus, the objective function is:

$$\theta^\star, \phi^\star = \arg\min_{\theta,\phi} -\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{ELBO}(\theta, \phi; x_i) \tag{2.1.7}$$

Likelihood $p_\theta(x|z)$ and posterior $q_\phi(z|x)$ construct an autoencoder structure. Since two functions are probability distributions, the likelihood function is called a probabilistic encoder and the posterior function is called a probabilistic decoder. Two distributions are often selected as Gaussians whose means and covariances are modeled by neural networks as shown in Figure 2.4.

2.1.4 Task-Conditioned Variational Autoencoder

Variational autoencoder can be trained for Learning from Demonstration (LfD) applications. Human expert's demonstration is often given in the form of end-effector trajectory and trajectory for robotic tasks has some constraints. For example, suppose that a human expert demonstrates pouring water into a cup. A demonstration is a continuous trajectory and a water bottle should be upright not to spill water until reaching near the cup. These constraints restrict the valid trajectories and trajectory demonstration data occupy the whole high-dimensional trajectory space. Thus, demonstration data is often assumed to form a manifold structure. By learning the demonstration manifold, we can easily generate a new trajectory that satisfies the above constraints without specifying the constraints analytically.

Task-conditioned variational autoencoder (TCVAE) [11] is proposed to learn a manifold structure of trajectory demonstration and generate a new trajectory. Figure 2.5 describes the structure of TCVAE. The input x and the reconstructed input $\tilde{x}$ of TCVAE are high-dimensional trajectory data. The difference between VAE and TCVAE is that the latent variable of TCVAE is split into two variables: task parameter w and manner parameter z. Task parameter w describes the goal

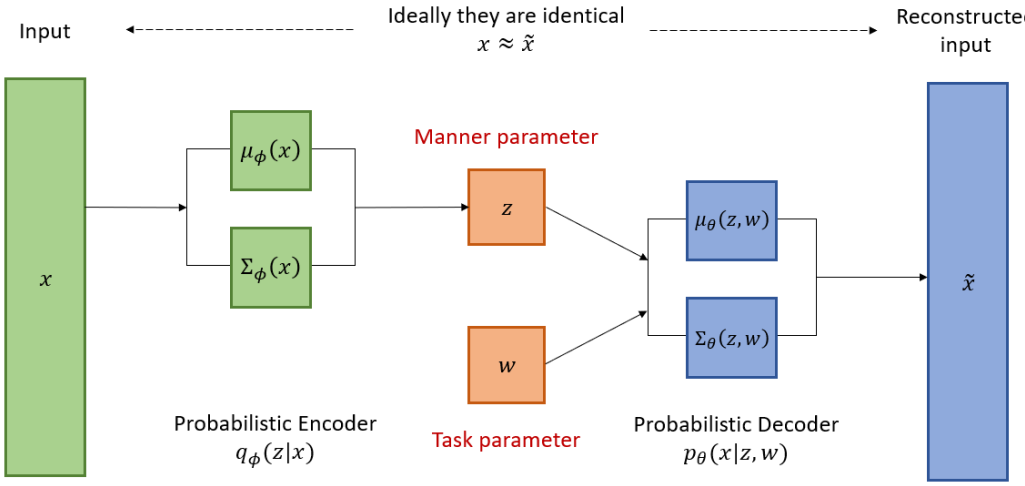

Figure 2.5: Structure of a task-conditioned autoencoder. The difference between task-conditioned autoencoder and variational autoencoder is a task parameter. For learning from demonstration problem, task of given demonstration is represented by a single parameter w. Probabilistic decoder gets latent variable (manner parameter) together with task parameter as an input to generate a trajectory which achieves the given task.

of trajectories such as the position of a cup or the initial configuration of a bottle. Manner parameter z describes other characteristics of the trajectory such as the shape of the trajectory or speed of the trajectory. A probabilistic encoder $q_\phi(z|x)$ encodes characteristics of the trajectory input x into manner parameter z, which is identical to the encoder of VAE. On the other hand, an input of the probabilistic decoder is augmented by concatenating the task parameter w with the manner parameter z. The decoder translates the latent manner parameter back into trajectory data which targets the given task parameter. A dataset for TCVAE should

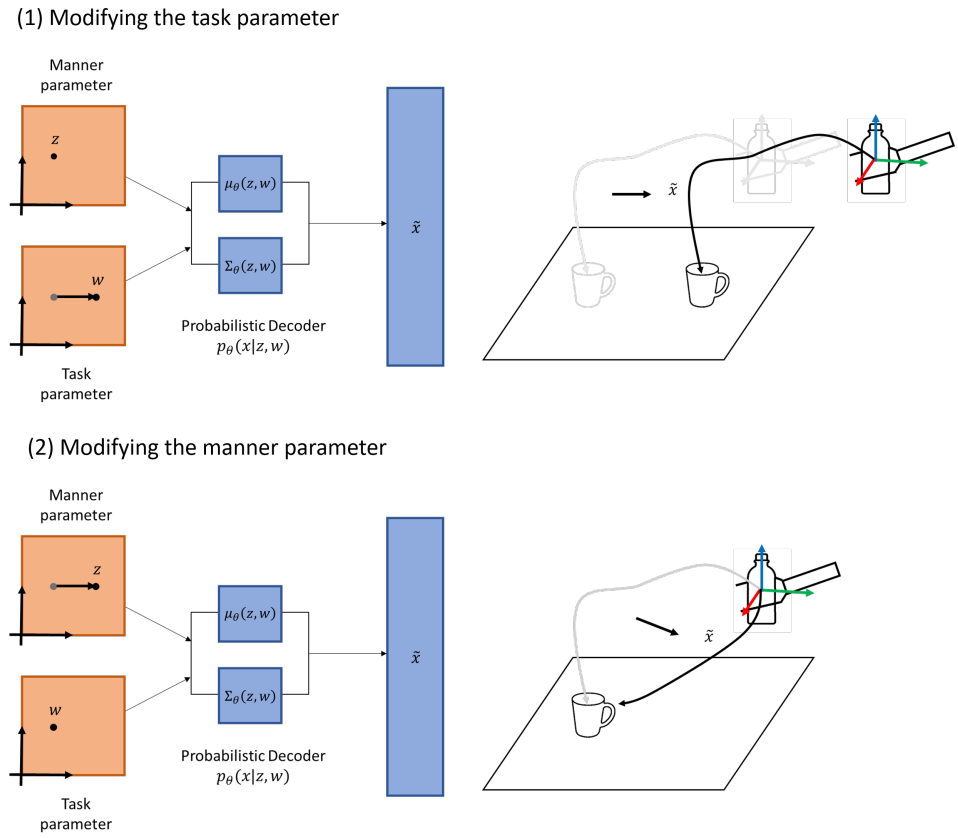

Figure 2.6: Decoder of task-conditioned autoencoder can generate a new trajectory. If we modulate the task parameter w, the decoder generates a new trajectory which aims a different goal while other characteristics such as shape or speed remain unchanged. If we modulate the manner parameter z, the decoder generates a new trajectory which has different shape while the goal remains unchanged.

have more information about task parameters. A pair of trajectory x and the corresponding task parameter w should be collected together and these pairs compose a training dataset $\mathcal{D}$:

$$\mathcal{D} = \{(x_1, w_1), (x_2, w_2), \cdots, (x_N, w_N)\} \tag{2.1.8}$$

and the evidence lower bound for TCVAE is:

$$\mathcal{L}_{ELBO}(\theta, \phi; x, w) = \mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x|z, w) - D_{KL}(q_\phi(z|x)||p(z)) \tag{2.1.9}$$

By the definitions of task and manner parameters, we want to decouple both parameters, that is, information about the goal of a trajectory cannot be inferred from the manner parameter. If not, the goal of the trajectory will change also by modifying the manner parameter. However, task-conditioned variational autoencoder does not ensure such decoupling properties and additional network and regularization terms are required for decoupling. An auxiliary neural network f_η is added which is used to predict the task parameter w from the manner parameter z and the auxiliary network f_η is trained by the following optimization:

$$\min_\eta \max_\phi \mathcal{L}_{aux} = \min_\eta \max_\phi L_1(f_\eta(\mu_\phi(x)), w) \tag{2.1.10}$$

By minimizing this auxiliary loss, the auxiliary network f_η becomes more skilled for inferring task parameters from manner parameters. Since our goal is to decouple the task parameter from the manner parameter, TCVAE can be adversarially optimized by the following alternating optimization problem:

$$\min_{\theta,\phi} \mathcal{L}_{ELBO} - \alpha \mathcal{L}_{aux} \quad \min_\eta \mathcal{L}_{aux} \tag{2.1.11}$$

Despite this auxiliary regularization, we observe that the regularization does not significantly improve the performance of TCVAE model and does not guarantee the decoupling. In this paper, we train the TCVAE model without the additional regularization term.

After training TCVAE, the decoder of TCVAE can generate a new trajectory with the given task. Figure 2.6 shows the generation process of the trajectory using

the decoder. Select a manner parameter z and a task parameter w and then decoding by the trained decoder generates the corresponding trajectory. If we modify the task parameter, the generated trajectory is transformed to execute the modified task parameter while maintaining the shape or speed of the trajectory. On the other hand, if we modify the manner parameter, the shape of the trajectory is changed while the objective of the trajectory is unchanged.

In LfD, collecting demonstration data is a highly time-consuming process and the number of training data often lacks for training a deep neural network without over-fitting. TCVAE also suffers from this lack of training data and over-fitting and often fails for the task parameter which is unseen in the training phase. This over-fitting problem can be resolved by considering the natural symmetry of the robotic task. In the next section, the natural symmetry of the task is introduced in the sense of group action, invariance and equivariance.

2.2 Symmetry and Group Action

The goal of this section is to define symmetry of robotic task mathematically. Symmetry is often used to refer to an object or a system that is invariant under some transformation such as translation, reflection, or rotation. The definitions of transformation and invariance are given in the following subsections.

2.2.1 Group Action

A group action is a way for a group to act on a set, it is defined as a function that assigns a transformation to each element of the group, preserving the group operation. In other words, a set of transformations on a set forms a group structure so that transformations can be composited and inverted like group elements

[17]. Let G be a group with identity element e and X be a set. Then a group action is a binary operation $\cdot : G \times X \to X$ satisfies the following two axioms:

$$\text{(Identity)}\ e \cdot x = x \tag{2.2.12}$$

$$\text{(Compatibility)}\ g_1 \cdot (g_2 \cdot x) = (g_1 g_2) \cdot x \tag{2.2.13}$$

where $x \in X$ and $g_1, g_2 \in G$. Under these axioms, the set X together with an action of G is called a $G-$set.

2.2.2 Invariance and Equivariance

To describe symmetry, invariance should be defined. Let X be a $G-$set and $\cdot : G \times X \to X$ be a group action of G on X. A function $f : X \to Y$ is said to be invariant about G if

$$f(g \cdot x) = f(x)\ \forall x \in X, g \in G \tag{2.2.14}$$

Since an object (function f) defined on a space X is not changed by the transformation (group action G), we often say that there exists a symmetry in the space X.

Meanwhile, we introduce another important concept about symmetry, equivariance. Assume that Y is also a $G-$set and $* : G \times Y \to Y$ be a group action of G on Y. A function $h : X \to Y$ is said to be equivariant about G if

$$h(g \cdot x) = g * h(x)\ \forall x \in X, g \in G \tag{2.2.15}$$

While an invariant function maps an element of X to the same element regardless of group action g, an equivariant function preserves the group action before and after mapping.

3

Group Invariant and Group Equivariant Neural Networks

In this chapter, we construct an invariant neural network and an equivariant neural network about a given symmetry group G. Invariant neural networks and equivariant neural networks can be constructed by transforming every input into a standard input. To define the standard input.

3.1 Group Invariant Neural Networks

A group-invariant neural network can be constructed as follows. Let $\tilde{f}_\theta : X \to Y$ be any function modeled by a neural network where θ is the parameter of the neural network. Group invariant function $f_\theta : X \to Y$ can be constructed by replacing all points that can be transformed to each other by some group action into some standard input x_0. More formally, define an equivalence relation $\sim$ by:

$$x_1 \sim x_2 \text{ if there exists } g \in G \text{ such that } x_1 = g \cdot x_2 \tag{3.1.1}$$

The equivalence relation $\sim$ partitions the set X into subsets which are called equivalence classes. For each equivalence class, pick a fixed point x_0 that represents all points in the equivalence class. With abuse of notation, let x_0 be a function that maps every point $x \in [x_0]$ to the standard x_0, that is $x_0(x) = x_0$. And define $g_0(x)$ as the group element in G such that $x_0 = g_0(x) \cdot x$ and $g_0(g \cdot x) = g_0(x)g^{-1}$. Then the function $f_\theta : X \to Y$ defined as

$$f_\theta(x) = \tilde{f}_\theta(g_0(x) \cdot x) = \tilde{f}_\theta(x_0(x)) \tag{3.1.2}$$

is invariant about the group G.

Proof.

$$\begin{aligned} f_\theta(g \cdot x) &= \tilde{f}_\theta(g_0(g \cdot x) \cdot (g \cdot x)) \\ &= \tilde{f}_\theta(g_0(x)g^{-1} \cdot (g \cdot x)) \\ &= \tilde{f}_\theta(g_0(x) \cdot x) \\ &= f_\theta(x) \end{aligned}$$

□

3.2 Group Equivariant Neural Networks

A group equivariant neural network can be constructed as follows. Let $\tilde{h}_\phi : X \to Y$ be any function modeled by a neural network where ϕ is the parameter of the neural network. Then the function $h_\phi : X \to Y$ defined as

$$h_\phi(x) = g_0(x)^{-1} * \tilde{h}_\phi(g_0(x) \cdot x) = g_0(x)^{-1} * \tilde{h}_\phi(x_0(x)) \tag{3.2.3}$$

is equivariant about the group G.

Proof.

$$\begin{aligned} h_\phi(g \cdot x) &= g_0(g \cdot x)^{-1} * \tilde{h}_\phi(g_0(g \cdot x) \cdot (g \cdot x)) \\ &= (g_0(x)g^{-1})^{-1} * \tilde{h}_\phi(g_0(x)g^{-1} \cdot (g \cdot x)) \\ &= g g_0(x) * \tilde{h}_\phi(g_0(x) \cdot x) \\ &= g * (g_0(x) * \tilde{h}_\phi(g_0(x) \cdot x)) \\ &= g * h_\phi(x) \end{aligned}$$

□

A group invariant neural network and a group equivariant neural network are used to construct an autoencoder model that considers the natural symmetry of robotic tasks in the following chapter.

4

Group Symmetric Autoencoder

In this chapter, we propose a group symmetric autoencoder model (i) which learns the manifold structure of trajectory data conditioned by task and (ii) which has an invariant encoder and equivariant decoder about the natural symmetry of the given task. To make a notation simple, we first begin with deterministic task-conditioned autoencoder rather than the variational form. In other words, encoder $h_\phi(x)$ and decoder $f_\theta(z, w)$ are mappings between vectors instead of distributions.

4.1 Construction of Group Symmetric Autoencoder

We argue that the encoder should be invariant and the decoder should be equivariant about the symmetry group to consider a natural symmetry of robotic task and improve data efficiency. Figure 4.1 shows invariance of an encoder. When trajectory x is transformed by group action g, that is $g \cdot x$, the characteristics of trajectory such as shape or speed are not changed. Since these characteristics are

encoded into manner parameter, encoder $h_\phi(z)$ should encode x and $g \cdot x$ invariantly, that is $h_\phi(z) = h_\phi(g \cdot x)$.

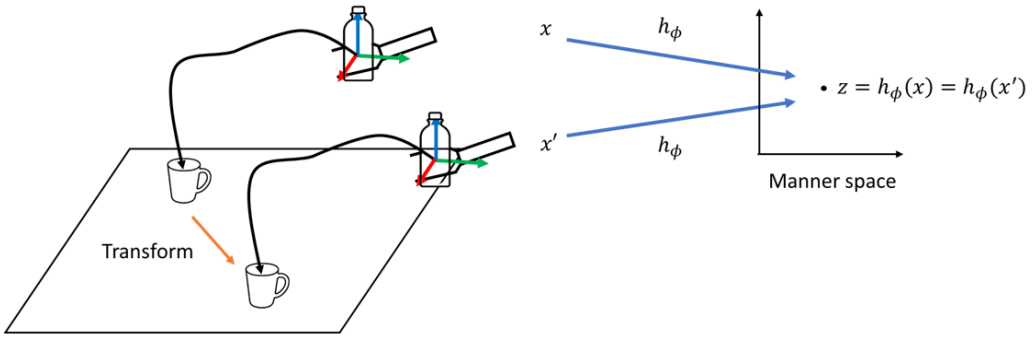

Figure 4.1: Encoder which is invariant to the given symmetry group. The invariant encoder maps the original trajectory x and the transformed trajectory $x' = g \cdot x$ into the same manner z, that is $h_\phi(x) = h_\phi(g \cdot x)$.

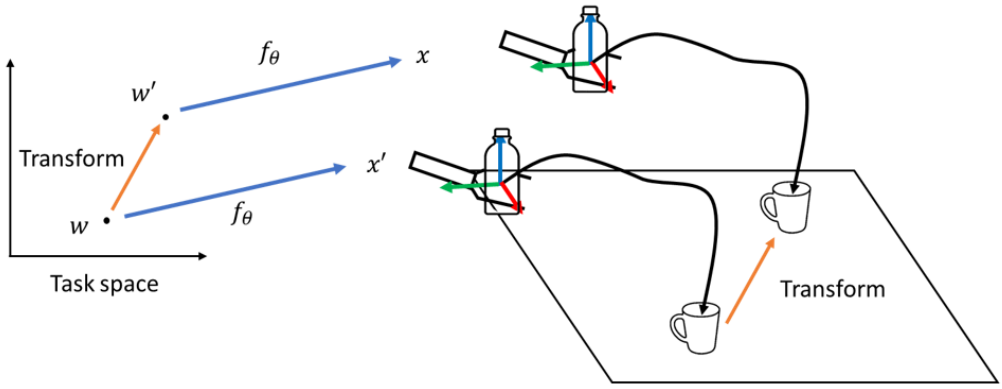

Figure 4.2: Decoder which is equivariant to the given symmetry group. If task parameter w is transformed to w', the outputs of w and w' by the decoder also differ by the same transformation, that is $f_\theta(z, g \cdot w) = g \cdot f_\theta(z, w)$.

Figure 4.2 shows equivariance of the decoder. By modulating task parameters and decoding through the decoder, we can generate a new trajectory that aims for

a generalized task. When we transform task parameter w by group action g, that is $g \cdot w$, two decoded trajectories $f_\theta(z, w)$ and $f_\theta(z, g \cdot w)$ have the same characteristics but execute the different tasks which are exactly transformed by g. Therefore, decoded trajectories $f_\theta(z, w)$ and $f_\theta(z, g \cdot w)$ should be also in the transformation of one to another, that is $f_\theta(z, g \cdot w) = g \cdot f_\theta(z, w)$.

Using invariant and equivariant neural networks introduced in Chapter 3, we can construct an invariant encoder and an equivariant decoder. Let $\cdot : G \times X \to X$ be a group action of G on X and $* : G \times W \to W$ be a group action of G on W. An invariant encoder can be constructed as the following steps:

(i) Model a neural network $\tilde{h}_\phi : X \to Z$

(ii) Define the standard input x_0 for each equivalent class in $X/\sim$.

(iii) Find a group action $g_0(x)$ that transforms x into the standard input x_0, that is $x_0 = g_0(x) \cdot x$.

(iv) Construct an invariant encoder:

$$h_\phi(x) = \tilde{h}_\phi(g_0(x) \cdot x)$$

An equivariant decoder can be constructed as the following steps:

(i) Model a neural network $\tilde{f}_\theta : (Z \times W) \to X$

(ii) Define the standard task w_0 for each equivalent class in $W/\sim$.

(iii) Find a group action $g_0(w)$ that transforms w into the standard input w_0, that is $w_0 = g_0(w) \cdot w$.

(iv) Construct an equivariant decoder:

$$f_\theta(z, w) = g_0(w)^{-1} \cdot \tilde{f}_\theta(z, g_0(w) * w)$$

So far, a group symmetric autoencoder is proposed, which has a deterministic encoder and decoder. If we generalize the model to a probabilistic model, the mean function and covariance function of the encoder and decoder should be invariant and equivariant. This modification is straightforward and we call this variational model as group symmetric variational autoencoder (GSVAE). In the experiment chapter, we will use the GSVAE model instead of the deterministic group symmetric autoencoder.

4.2 Measures for Invariance and Equivariance

While GSVAE has a completely invariant encoder and equivariant decoder, other models are not explicitly regularized for the encoder and decoder to be invariant and equivariant. Therefore, measures to quantify invariance and equivariance are required. Invariance measure $\mathcal{I}$ for encoder f_θ is defined as:

$$\mathcal{I}(f_\theta)(x) = \frac{1}{m}\sum_{i=1}^{m} \|f_\theta(x) - f_\theta(g_i \cdot x)\|_2^2 \tag{4.2.1}$$

where g_i $(i = 1, 2, \cdots, m)$ is randomly sampled group action from G. Since the manner space is a Euclidean space, we choose the distance between two manner parameters as L_2-norm $\|\cdot\|_2$. Equivariance measure $\mathcal{E}$ for decoder h_ϕ is defined as:

$$\mathcal{E}(h_\phi)(z, w) = \frac{1}{m}\sum_{i=1}^{m} \text{dist}\left(h_\phi(z, g_i * w), g_i \cdot h_\phi(z, w)\right) \tag{4.2.2}$$

We need to define a distance function $\text{dist}(\cdot,\cdot)$ between two trajectories of configuration. Since a demonstration trajectory is a sequence of $SE(3)$ for water pouring dataset, the distance between two sequences of $SE(3)$ should be defined. We defined the distance function $\text{dist}(\cdot,\cdot)$ as an average of geodesic distance between two

$SE(3)$ for every time step. The geodesic distance between two $SE(3)$ is defined as:

$$d(T_1, T_2) = \| \log(R_1^{-1} R_2) \|_2^2 + \|p_1 - p_2\|_2^2 \tag{4.2.3}$$

where $R_i \in SO(3)$ and $p_i \in \mathbb{R}^3$ are the rotation matrix and the position vector of $T_i \in SE(3)$.

5

Experiments

In this section, we will compare TCVAE and group symmetric variational autoencoder (GSVAE, our model) on water pouring dataset [7]. Reconstruction loss and invariance/equivariance measures will be provided for quantitative comparison of two models. Reconstruction visualization, manner parameter modulation, and task parameter modulation will be provided for qualitative comparison of two models.

5.1 Water Pouring Dataset

We demonstrate GSVAE and TCVAE to real world LfD problem, water pouring. Configuration of water pouring problem is shown in Figure 5.1. A trajectory $x = [T_1, T_2, \cdots, T_{N_t}] \in SE(3)^{N_t} = X$ consists of N_t samples of $SE(3)$ which are expressed about the fixed frame $\{s\}$. A task parameter $w = [x_c, y_c, z_b, y_b, z_b, \theta_{EE}] \in \mathbb{R}^5 \times \mathbb{S}^1 = W$ is consist of 6 parameters where (x_c, y_c) is a planar position of the cup, (x_b, y_b, z_b) is a spatial initial position of the bottle, and θ_{EE} is a rotation angle of bottle frame $\{b\}$ about bottle axis $\{z_b\}$.

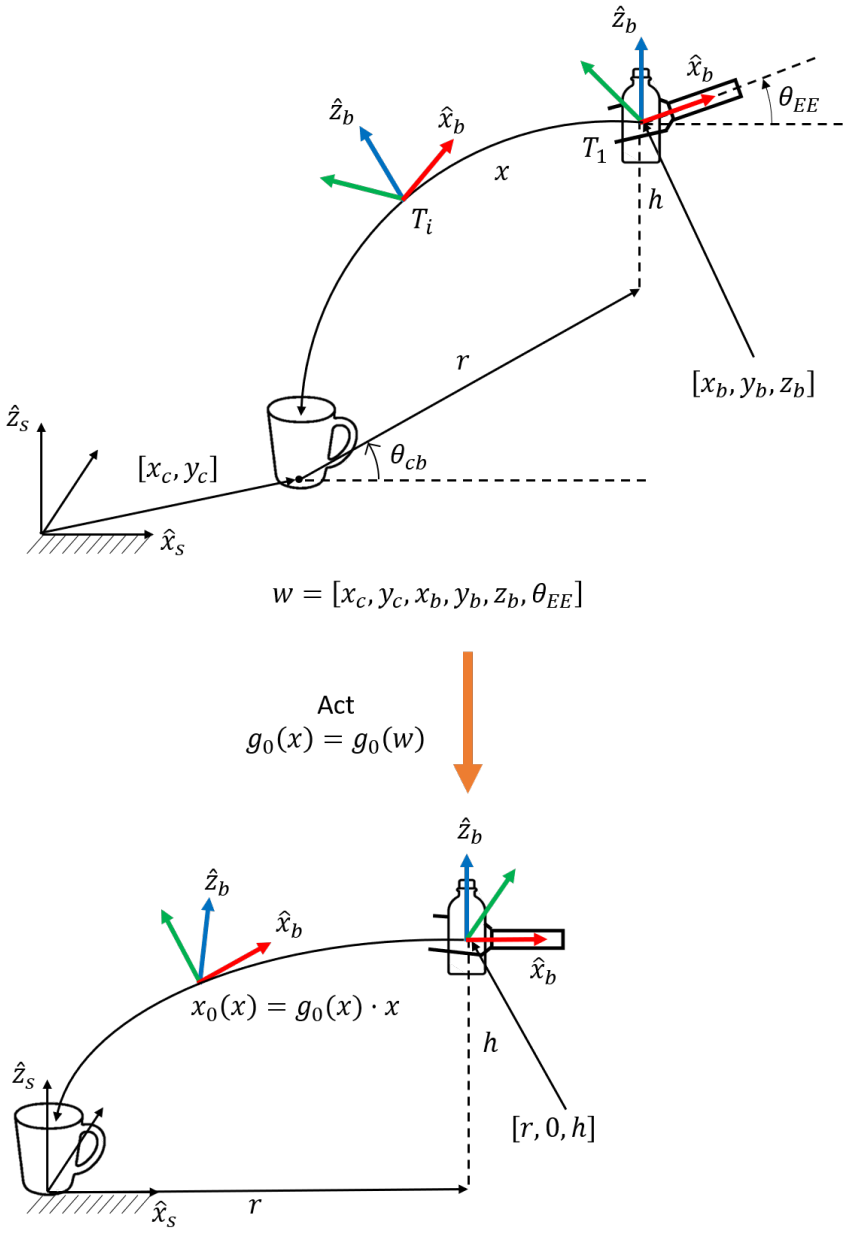

Figure 5.1: Configuration of water pouring problem. Task parameter is identified as $w = [x_c, y_c, x_b, y_b, z_b, \theta_{EE}]$ and each trajectory is transformed to the standard trajectory whose pouring direction is aligned with $-\hat{x}_s$ and initial configuration of bottle is also aligned with $\hat{x}_s$.

To implement GSVAE, an user should specify a symmetry of task and group action. There are three different symmetry in water pouring problem: (i) translation about plane, (ii) rotation about axis of cup, and (iii) rotation of bottle frame $\{b\}$ about axis of bottle. We set the symmetry group of water pouring problem as $G = \mathbb{R}^2 \times \mathbb{T}^2$ and define the group actions of G on X and $Z \times W$ as follows:

Group action of G on X

For $g = [\Delta x, \Delta y, \Delta\theta_{cb}, \Delta\theta_{EE}] \in \mathbb{R}^2 \times \mathbb{T}^2 = G$, $\cdot : G \times X \to X$ is defined as follows:

$$g \cdot x = g \cdot [T_1, \cdots, T_{N_t}] = \left[T'_1, \cdots, T'_{N_t}\right], \tag{5.1.1}$$

where $T'_i = Rot\left(\hat{z}, \Delta\theta_{cb}\right) T_i Rot\left(\hat{z}, \Delta\theta_{EE}\right) + Trans\left([\Delta x, \Delta y, 0]\right)$

Group action of G on W

For $g = [\Delta x, \Delta y, \Delta\theta_{cb}, \Delta\theta_{EE}] \in \mathbb{R}^2 \times \mathbb{T}^2 = G$, $* : G \times W \to W$ is defined as follows:

$$g * w = g * \begin{bmatrix} x_c \\ y_c \\ x_b \\ y_b \\ z_b \\ \theta_{EE} \end{bmatrix} = \begin{bmatrix} x_c + \Delta x \\ y_c + \Delta y \\ x_b + \Delta x + (d_x \cos\Delta\theta_{cb} - d_y \sin\Delta\theta_{cb}) \\ y_b + \Delta y + (d_x \sin\Delta\theta_{cb} + d_y \cos\Delta\theta_{cb}) \\ z_b \\ \theta_{EE} + \Delta\theta_{cb} + \Delta\theta_{EE} \end{bmatrix} \tag{5.1.2}$$

where $d_x = x_b - x_c$ and $d_y = y_b - y_c$.

For group symmetric representation learning, we need to specify standard trajectory x_0, standard task w_0, and group actions $g_0(x), g_0(w)$. In water pouring problem, we set the standard input as the trajectory where the cup is on the origin and the bottle frame $\{b\}$ is aligned with $\{s\}$ as shown in the bottom of

Figure 5.1. Formal definitions of $g_0(x)$, $g_0(w)$, and $w_0(w)$ is given as:

$$g_0(x) = g_0(w) = g_0\left(\begin{bmatrix} x_c \\ y_c \\ x_b \\ y_b \\ z_b \\ \theta_{EE} \end{bmatrix}\right) = \begin{bmatrix} -x_c \\ -y_c \\ -\operatorname{atan2}(d_y, d_x) \\ \operatorname{atan2}(d_y, d_x) - \theta_{EE} \end{bmatrix} \tag{5.1.3}$$

$$w_0(w) = \left[0, 0, \sqrt{d_x^2 + d_y^2}, 0, z_b, 0\right] \tag{5.1.4}$$

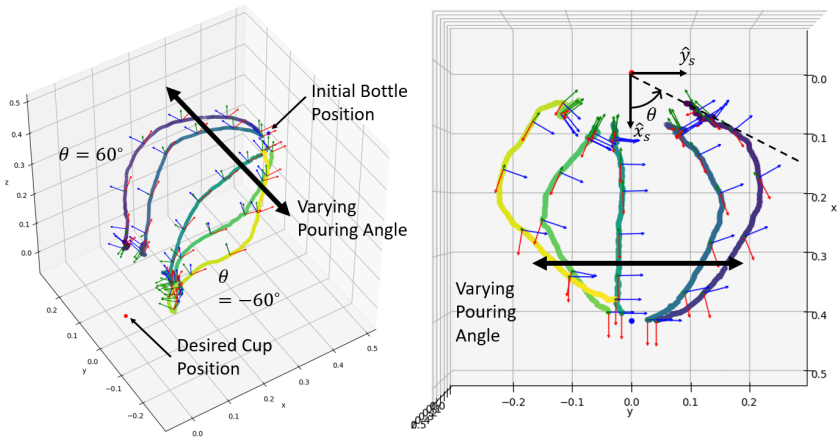

Figure 5.2: Pouring angle θ is the angle between $\hat{x}_s$ and the pouring direction. We collect various pouring trajectory with 5 different pouring angle, $\{-60°, -30°, 0°, 30°, 60°\}$. Varying pouring angle will form 1-dimensional manner space.

5.2 Implementation Detail

For Learning from Demonstration, human expert should collect trajectory data. We collect the end-effector trajectory by AprilTag SE(3) detection algorithm as shown in Figure 5.3. AprilTag algorithm segments the attached tag on the bottle and infers the configuration of the tag. For autoencoder training, we collect $75 = 5 \times 3 \times 5$ standard trajectories with varying $r \in \{0.4\text{ m}, 0.5\text{ m}, 0.6\text{ m}, 0.7\text{ m}, 0.8\text{ m}\}$, $h = \{0.4\text{ m}, 0.5\text{ m}, 0.6\text{ m}\}$ and pouring angle $\theta = \{-60°, -30°, 0°, 30°, 60°\}$ where the pouring angle θ is the angle between $\hat{x}_s$ and the pouring direction as shown in Figure 5.2. Varying pouring angle from $-60°$ to $60°$ will form 1-dimensional manner space.

Figure 5.3: Demonstration of end-effector trajectory which pours water into the desired cup position. AprilTag SE(3) detection algorithm infers the configuration of attached tag.

In addition, for validation and test, we collect 60 standard trajectories with randomly sampled r, h, and θ. Trajectory data is transformed by randomly sampled group action $g \in G$ in every batch. Since sampling rate of trajectory is 30 fps and total time is 9 s, size of a trajectory data is $3240 = 270 \times 12$ where 12 is the number of elements in $SE(3)$ matrix except the last row.

We use fully connected neural network for $\tilde{f}_\theta$ and $\tilde{h}_\phi$ and set latent space (manner space) dimension as 1. Autoencoder is trained with Adam optimizer and 0.0001 step size.

5.3 Results

5.3.1 Reconstruction Quality

Reconstruction loss for training dataset shows the accuracy of manifold learning and reconstruction loss for test dataset shows the generalizabilty of the model. Reconstruction losses for GSVAE and TCVAE are shown in Table 5.1.

Networks size is the number of hidden nodes and the number of hidden layers of MLP networks that model $\tilde{f}_\theta$ and $\tilde{h}_\phi$. Reconstruction loss is L_2-norm between data and reconstructed data. In Table 5.1, the larger network size of model is, the smaller reconstruction loss for training is since the network become flexible. In contrast, the reconstruction loss for test dataset does not have same tendency. If the network size is too large, the reconstruction loss for test dataset saturates due to the over-fitting. To compare the accuracy of manifold learning and generalizability of two models, we selected TCVAE (512×2) and GSVAE (32×2) which are colored red. Test reconstructions for two models are nearly identical while train reconstructions and the number of parameters for GSVAE is much smaller than

Table 5.1: Reconstruction error and the number of parameters of TCVAE and GSVAE. Train reconstruction error shows the accuracy of manifold learning and test reconstruction error shows the generalizability of each model. Reconstruction error of GSVAE is lower than that of TCVAE with much less number of parameters.

Model (Net size)	**Train recon.**	**Test recon.**	**Number of parameters**
TCVAE (32×2)	1.88×10^{-2}	2.62×10^{-2}	213,099
TCVAE (128×2)	5.91×10^{-3}	5.38×10^{-3}	867,243
TCVAE (512×2)	3.30×10^{-3}	5.70×10^{-3}	**3,852,459**
GSVAE (32×2)	1.14×10^{-3}	5.71×10^{-3}	**212,939**
GSVAE (128×2)	2.25×10^{-4}	6.22×10^{-3}	866,603
GSVAE (512×2)	2.25×10^{-5}	5.74×10^{-3}	3,849,899

TCVAE. Therefore, GSVAE can learn more accurate data manifold with less number of parameters.

We also qualitatively compare the reconstruction qualities of two models in Figure 5.4. Blue line is an original demonstration trajectory and orange line is a reconstructed trajectory by each model. A cup position is indicated as a black dot. Reconstructed trajectory of TCVAE has some variance with the original demonstration trajectory while that of GSVAE is identical with the original demonstration trajectory. In addition, the reconstructed trajectory of TCVAE is noisy so that the trajectory should be filtered to be used for real robot implementation.

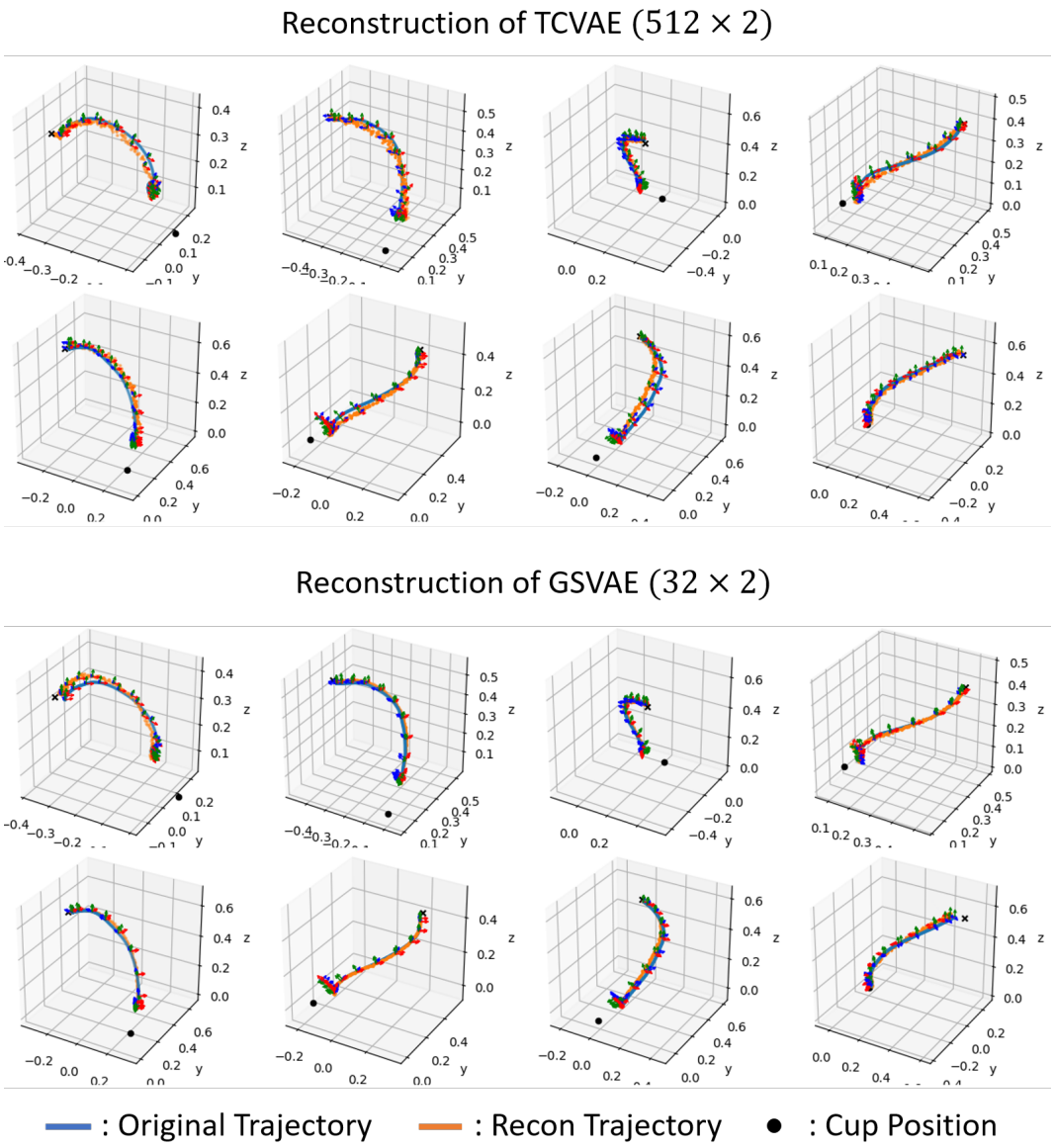

Figure 5.4: Reconstruction quality of each model. Blue line is the original trajectory and orange line is the reconstructed trajectory. Cup position is marked with a black dot. The trajectory reconstructed by TCVAE is far more deviated from the original trajectory than that of GSVAE. In addition, the trajectory reconstructed by TCVAE is highly noisy which is not adequate for real robot implementation.

Table 5.2: Encoder invariance and decoder equivariance are measured for each model. Invariance and equivariance of GSVAE is numerically zero while those of TCVAE is not zero. This result shows that invariance and equivariance cannot be achieved only by the data augmentation.

Model (Net size)	**Encoder Invariance**	**Decoder Equivariance**
TCVAE (32×2)	9.11×10^{0}	7.10×10^{2}
TCVAE (128×2)	3.78×10^{0}	1.55×10^{2}
TCVAE (512×2)	7.40×10^{0}	4.15×10^{2}
GSVAE (128×2)	1.80×10^{-12}	3.88×10^{-13}
GSVAE (512×2)	9.28×10^{-12}	3.95×10^{-13}
GSVAE (512×2)	2.10×10^{-12}	3.76×10^{-12}

5.3.2 Invariance and Equivariance

Since we augment the standard trajectory by randomly selected group action, TCVAE might be ale to learn invariance and equivariance of task symmetry without the explicit regularization of network. To evaluate the invariance and equivariance of the model, we evaluate two models with metrics defined in Section 4.2. Table 5.2 shows invariance and equivariance of two models. we observe that TCVAE model cannot learn an invariant encoder and an equivariant decoder although training data are augmented by group action. Also, we can check our invariant and equivariant neural network designs are valid from invariance and equivariance measure of GSVAE. This shows that data augmentation does not ensure the invariance and equivariance about group symmetry.

For other downstream tasks such as obstacle avoidance, LfD model should be

able to generate various trajectories with different shapes. To generate a trajectory with new characteristics, we can modulate latent variable (manner parameter). Figure 5.5 and Figure 5.6 show a manner modulation performance of TCVAE and GSVAE, respectively. We select a trajectory from training dataset and modify the manner parameter. More precisely, encode the training trajectory, fix the task parameter, modify the task parameter to linearly interpolated manner parameters $\{-1, -0.5, 0, 0.5, 1\}$, and then decode by trained decoder. In each figure, cup position (goal) is marked with black dot and bottle position (initial) is marked with black cross. pouring directions are aligned to the vertical direction. Trajectory shape (left, straight, or right with respect to the pouring direction) is marked with black arrow. Figure 5.5 shows the manner modulation results of TCVAE. Trajectory shapes are randomly varying while manner parameter is monotonically increasing. For example, the trajectory shapes of Task 1 are varying in the order of right, left, left, left, and center. This shows that trajectories are not smoothly varying and the learned trajectory manifold by TCVAE is heavily twisted. In addition, each manifold with different task does not have aligned latent coordinates. For example, the manner parameters with straight pouring direction are 1, 0, 0, 1, 0 in Task $1 \sim 5$. This shows that manifolds with different task parameters are not aligned.

5.3.3 Manner Modulation

Figure 5.6 shows the manner modulation results of GSVAE. Trajectory shapes are smoothly varying from left to right while manner parameter is monotonically increasing. Furthermore, straight pouring direction is aligned with manner 0. This shows that GSVAE can learn smooth manifold which is easy to modulate latent variable and aligned manifolds which have different task parameter. We suggest

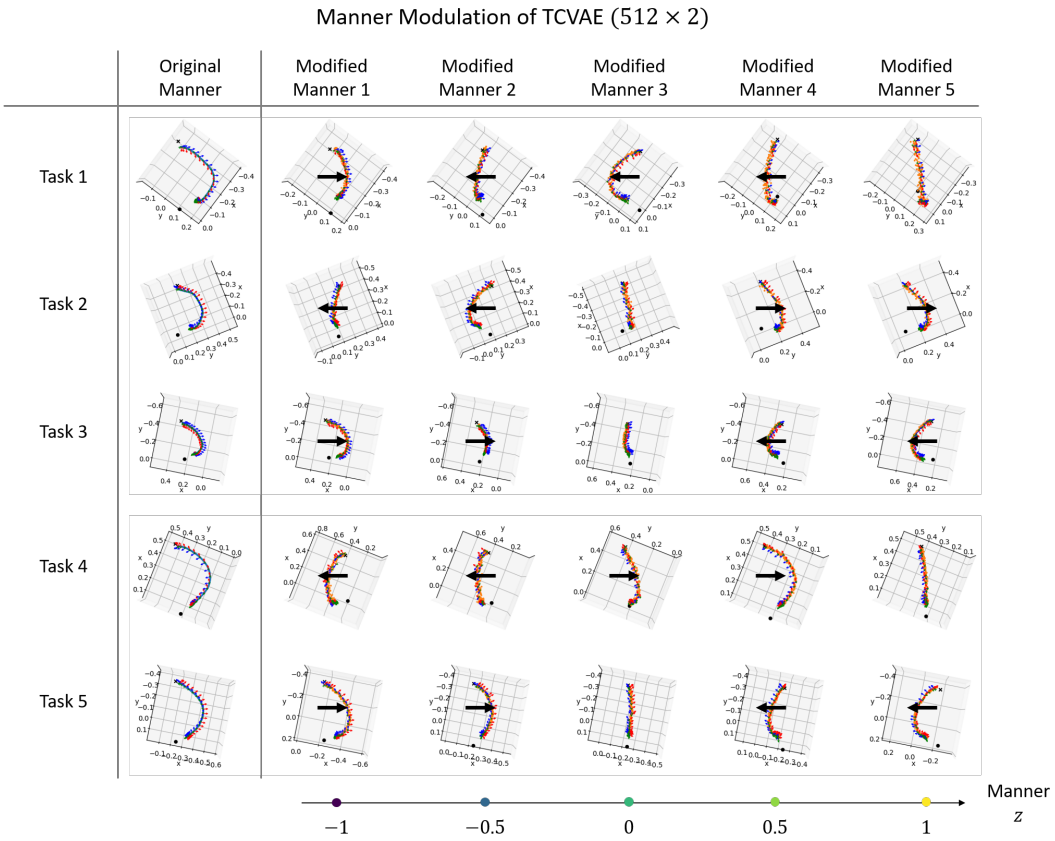

Figure 5.5: Manner modulation of TCVAE. We randomly select 5 training data and change each manner parameter to linearly interpolated manner parameter $\{-1, -0.5, 0, 0.5, 1\}$. Trajectories are aligned to the vertical direction for visualization. Manner modulation results in new trajectory with different shape. However, trajectories generated by TCVAE do not smoothly vary with monotonically increasing manner parameter. This shows that the data manifold learned by TCVAE is highly twisted. In addition, manifolds with different task parameters do not have a aligned latent space. The manifolds with task 2, 3, 5 have straight pouring trajectory at manner zero but the manifolds with task 1, 4 have straight pouring trajectory at manner 1.

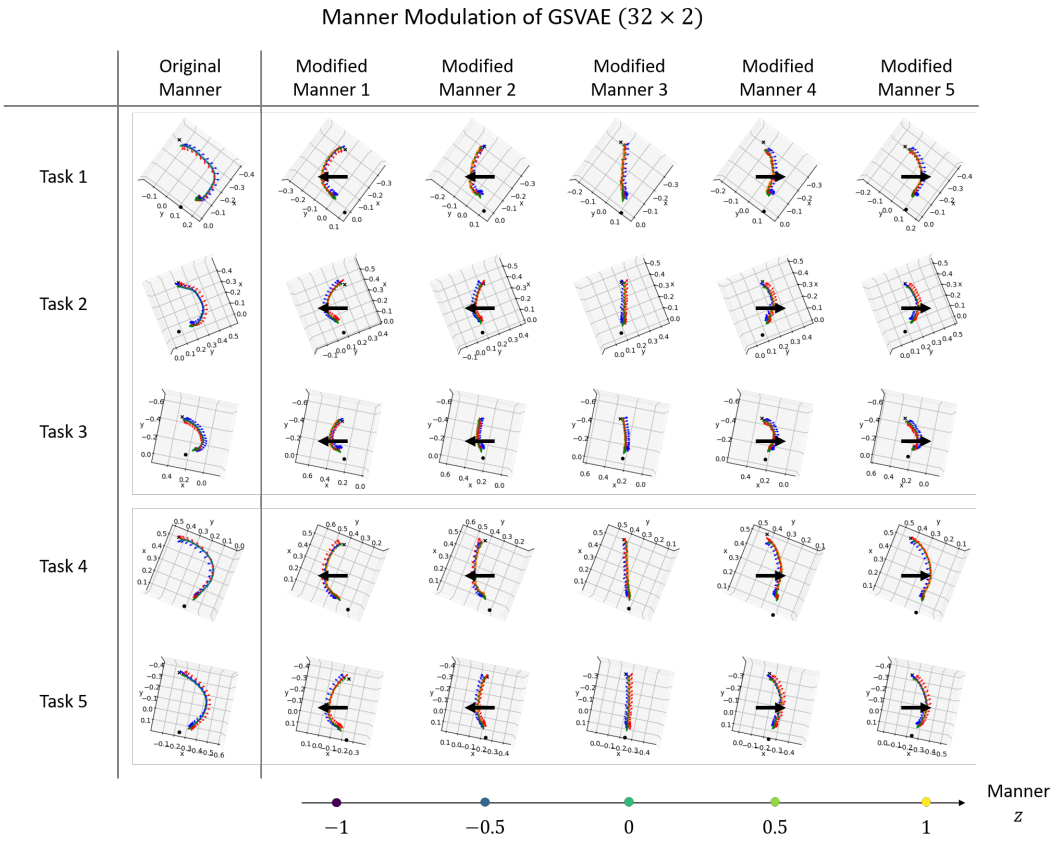

Figure 5.6: Manner modulation of GSVAE. The trajectories generated by GSVAE smoothly vary with monotonically increasing manner parameter. This shows that the data manifold learned by GSVAE is smooth. In addition, manifolds with different task parameters are also aligned. For every task, trajectories with straight pouring direction are consistently at manner zero.

that the reason why GSVAE can smooth and aligned manifolds is because every training data is aligned and trajectories with the same manner become close.

5.3.4 Task Modulation

For robot to pour water into different goals, LfD model should be able to generate various trajectories with different tasks. To generate a trajectory with a new goal, we can modulate task parameter. Figure 5.7 shows the task modulation results of TCVAE. Each generated trajectory successfully follow the given task parameters. However, although the manner parameter is fixed and only task parameters are varying, pouring directions of generated trajectories are also varying. This shows that learned manner parameter and task parameter are not decoupled by TCVAE and this is the key limitation of TCVAE model.

Figure 5.8 shows that task modulation results of GSVAE. Each generated trajectory successfully follow the given task parameters. In addition, trajectory directions are remain fixed and only tasks of trajectories are varying. GSVAE can learn decoupled task parameter and manner parameter.

5.3.5 Robot Implementation

To evaluate the trajectories generated by GSVAE and TCVAE, we implement the generated trajectories with FRANKA EMIKA robot, which is a 7DoF robot arm. We generate trajectories by decoding randomly sampled manner parameter and task parameter. Inverse kinematics should be solved for robot to follow the generated end-effector trajectory and we use a numerical inverse kinematics algorithms introduced in [2]. Since the work space of the FRANKA robot is limited, we transform the generated trajectories to the standard trajectory and then solve inverse

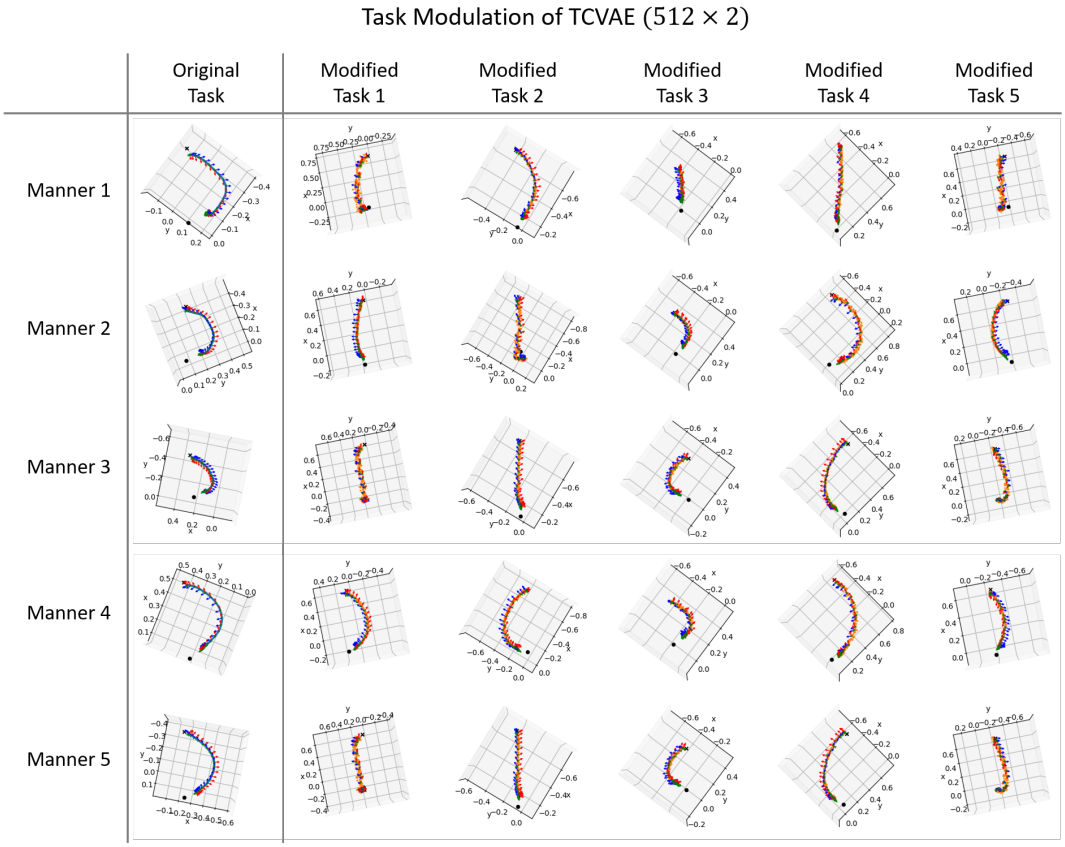

Figure 5.7: Task modulation of TCVAE. We randomly select 5 training data and change each task parameter to randomly selected cup positions. Task modulation results in new trajectory with different goal position. However, trajectories generated by TCVAE do not preserve the shape of trajectory although manner parameter remain fixed. This result shows that task parameter and manner parameter trained by TCVAE are coupled.

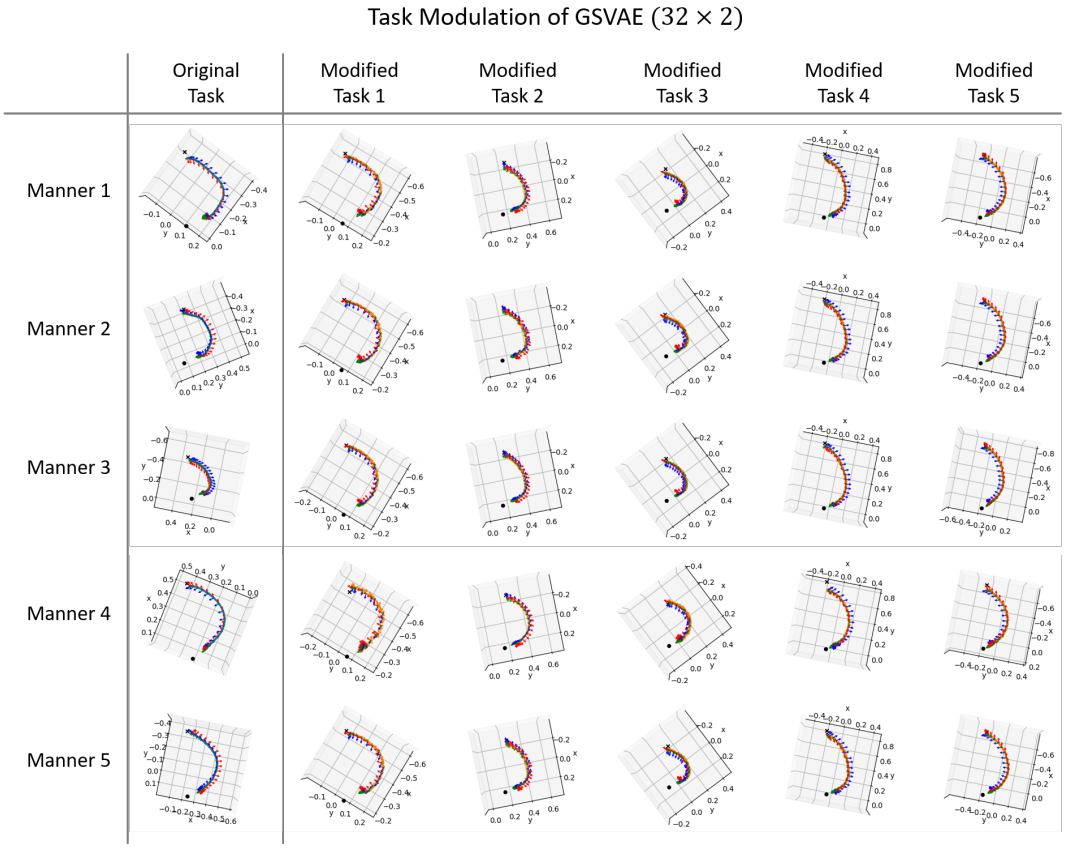

Figure 5.8: Task modulation of GSVAE. Trajectories generated by GSVAE preserve the shape of trajectory. This result shows that task parameter and manner parameter trained by GSVAE are decoupled.

kinematics. We smoothen the noisy trajectories by moving average with window size 15 and upsample the solved joint angle trajectories to fps 1000 since time step between each joint angle should be 0.001 s for position control of the FRANKA robot. Before implementing to the real robot, we check the validity of generated joint trajectory on simulation environment as shown in Figure 5.9.

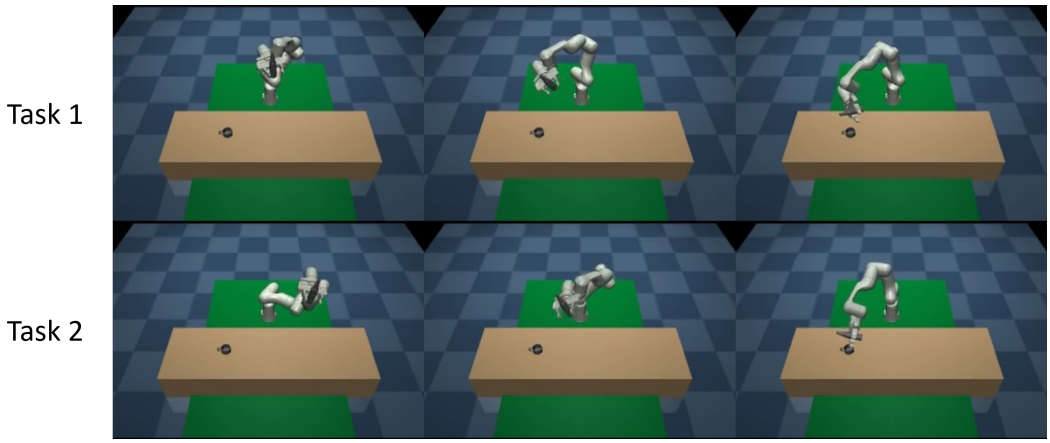

Figure 5.9: Check validity of generated trajectories on mujoco simulation environment.

Figure 5.11 shows the robot pouring water with the trajectories randomly generated by TCVAE and GSVAE. Two failure modes are observed in the trajectories generated by TCVAE. The trajectories do not pour water into the cup since the pouring angle of bottle is insufficient or the trajectories spill the water out of the desired cup position as shown in Figure 5.10. The trajectories of TCVAE are not smooth so that inverse kinematics cannot be easily solved. Meanwhile, GSVAE outputs sufficiently smooth and accurate trajectories.

We also count the success rate of water pouring of each model and the result is given in Table 5.3. We generate 14 trajectories with randomly chosen task parameters and manner parameters and then implement them with the robot. Success

rate of TCVAE is $7/14 = 50\%$ and that of GSVAE is $12/14 = 86\%$.

Table 5.3: Success rate of water pouring. The trajectories generated by GSVAE are more accurate and smooth to achieve the given tasks.

Model (Net size)	**TCVAE** (512×2)	**GSVAE** (32×2)
Success rate	7/14	12/14

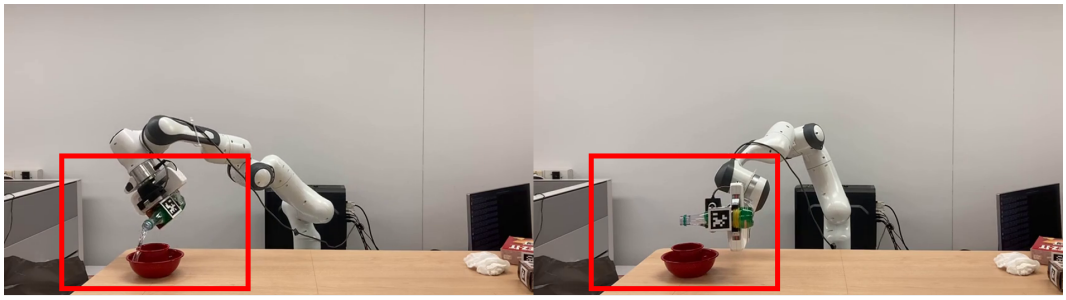

Figure 5.10: Failure modes of TCVAE. The trajectories genereated by TCVAE often spill the water out of the cup or does not pour the water at all.

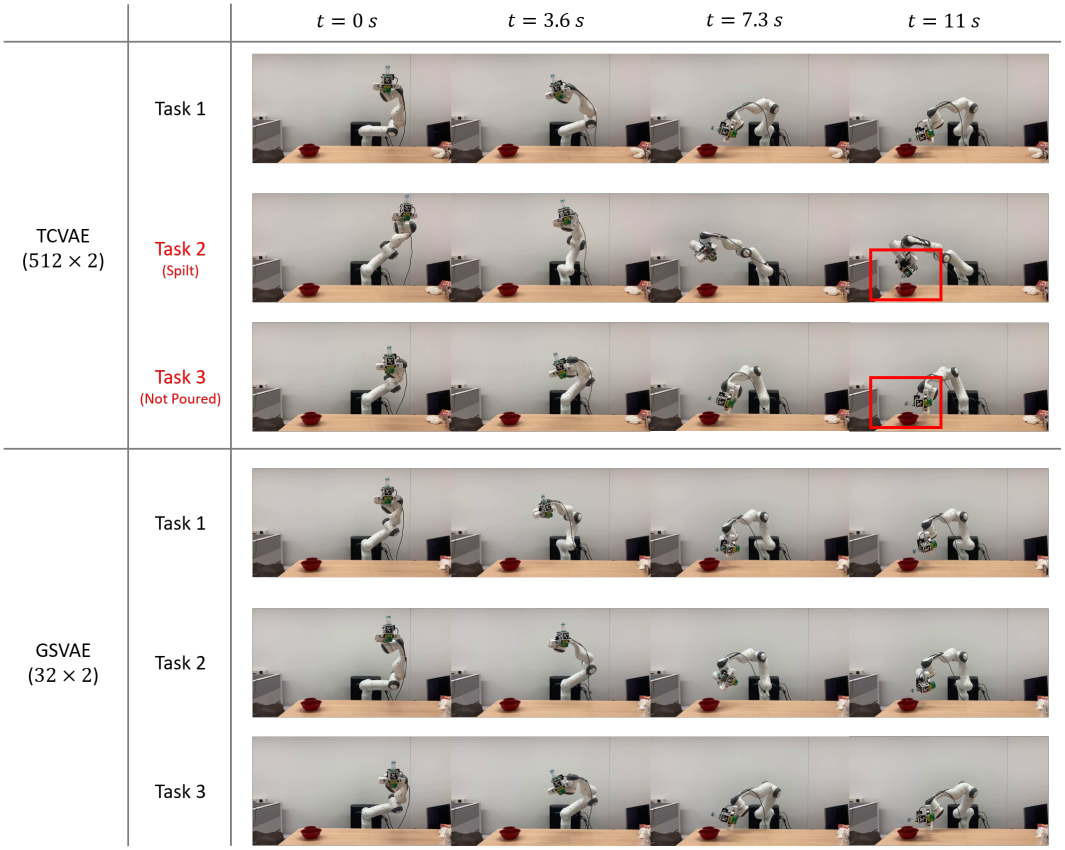

Figure 5.11: Robot implementation of the trajectories generated by each model. We randomly sample manner and task parameters and decode to trajectories. In the case of GSVAE, most of the trajectories are smooth so that robot successfully pour water into the desired cup position. However, in the case of TCVAE, two failure modes are observed. In Task 2, the trajectory is far away from the cup position so that the robot spill water. In Task 3, wrist angle are too small so that the robot do not pour water into the cup.

6

Conclusion

We have proposed group symmetric variational autoencoder (GSVAE) to learn a demonstration trajectory of a human expert with considering manifold structure in trajectory space and natural symmetry of robotic task. We have introduced group action and invariance/equivariance to formulate the symmetry of task and developed invariant and equivariant neural networks which are used for a group-invariant encoder and a group-equivariant decoder of GSVAE. Two algorithms, TCVAE and GSVAE, are compared with water pouring experiments. We have verified the efficacy of GSVAE with diverse quantitative and qualitative measures such as reconstruction loss, invariance/equivariance measure, latent modulation, and success rate of robot implementation. We argue that our GSVAE model has three contributions. By considering the natural symmetry of the given task, GSVAE can (i) learn more accurate and more smooth data manifold with fewer neural network parameters, (ii) ensure the invariance and equivariance about group action without any data augmentation, and (iii) align and decoupled task and manner.

Bibliography

[1] Lilian Weng. From autoencoder to beta-vae. *lilianweng.github.io*, 2018.

[2] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[3] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494. IEEE, 2009.

[4] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[5] Byungchul An, Jinkyu Kim, and Frank C Park. An adaptive stepsize rrt planning algorithm for open-chain robots. *IEEE Robotics and Automation Letters*, 3(1):312–319, 2017.

[6] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

[7] Mingshan Chi, Yufeng Yao, Yaxin Liu, Yiqian Teng, and Ming Zhong. Learning motion primitives from demonstration. *Advances in Mechanical Engineering*, 9(12):1687814017737260, 2017.

[8] Takayuki Osa, Amir M Ghalamzan Esfahani, Rustam Stolkin, Rudolf Lioutikov, Jan Peters, and Gerhard Neumann. Guiding trajectory optimization by demonstrated distributions. *IEEE Robotics and Automation Letters*, 2(2):819–826, 2017.

[9] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[10] LEE Yonghyeon, Sangwoong Yoon, MinJun Son, and Frank C Park. Regularized autoencoders for isometric representation learning. In *International Conference on Learning Representations*, 2021.

[11] Michael Noseworthy, Rohan Paul, Subhro Roy, Daehyung Park, and Nicholas Roy. Task-conditioned variational autoencoders for learning movement primitives. In *Conference on robot learning*, pages 933–944. PMLR, 2020.

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

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[14] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[15] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[16] John M Lee. Smooth manifolds. In *Introduction to smooth manifolds*, pages 1–31. Springer, 2013.

[17] John B Fraleigh. *A first course in abstract algebra*. Pearson Education India, 2003.

국문초록

시연 학습은 최적화나 샘플링 기반의 고전적인 동작 계획 알고리즘들의 한계를 해결하는 강력한 방법론이다. 비용 함수를 명시적으로 정의하거나 고차원 공간에서 샘플링하지 않고도 동작 계획기는 인간 전문가의 시연을 학습하여 사람과 비슷하고 정확한 동작을 생성할 수 있다. 시연 데이터는 흔히 엔드 이팩터 경로의 형태로 주어지기 때문에 고차원이다. 이러한 고차원 데이터를 다루기 위해 고차원 데이터가 저차원의 다양체 위에 포함되어 있다고 가정하고, 이를 다양체 가정이라 부른다. 오토인코더는 데이터의 다양체 구조를 학습하기 위해 널리 사용되는 심층 생성 모델 중 하나이다. 그러나 시연 과정은 시간을 크게 소모하기 때문에 학습 데이터는 필연적으로 적다. 학습 데이터의 부족은 인공 신경망 모델이 과적합 되도록 하고, 정확한 동작을 생성해낼 수 없게 한다. 평행이동이나 회전이동된 경로는 실제로는 같은 것이기 때문에, 평행이동과 회전이동에 대한 대칭성을 모델에 반영하면 데이터의 효율성을 증가시킬 수 있고 과적합을 방지할 수 있다. 이 논문에서는 먼저 로봇 작업에 내재된 대칭성을 군의 작용으로 표현한다. 그리고 대칭성을 나타내는 군에 대해 불변인 인코더와 등변인 디코더를 가지는 군 대칭 오토인코더를 제안한다. 군 대칭 오토인코더가 다른 비교 모델에 비해 더 정확한 다양체를 학습하며 더 정확한 동작을 생성해낼 수 있음을 물붓기 데이터셋을 통해 입증하였다.

주요어: 시연 학습, 다양체 학습, 표현 학습, 오토인코더, 작업의 대칭성, 군에 대한 불변성/가변성

학번: 2021-21058