博士論文

Representation Learning with Neural Networks for Structural World
(ニューラルネットワークによる構造的外界の表現学習)

Yoshihiro Nagano
長野祥大

ABSTRACT

Deep learning is an algorithm that solves machine learning problems with differentiable computational graphs and gradient-based learning methods. While standard machine learning techniques focus on feature engineering, deep learning made it possible to extract essential features from complex datasets automatically. A key to the extraction of a good representation is a model of an object of appropriate granularity. A number of network architectures have been proposed to capture the structure of datasets in various domains.

However, most deep learning research focuses on the correlation between dimensions of the data. Real-world datasets often have specific correlation structures between data points, although many algorithms typically assume the data distribution is i.i.d. An assumption on the instance-wise structure behind the dataset in such a situation is useful from both the engineering and scientific perspectives. For engineering, the structural assumption constrains the space to be explored and improves performance. For science, the reasonable constraints on an external world will give us the reason why efficient information processing systems is in its present form.

Based on the aforementioned motivations, we present a numerical analysis and model proposal of a deep learning algorithm from the perspective of instance-wise structure. By using the representative representation learning algorithm, called variational autoencoder (VAE), as a basic technique, we focus on three types of instance-wise structures: cluster structure, hierarchical structure, and local structure. First, on the inference of VAE, the sequential application of VAE to a noisy image data gradually removes the noise from the image. We numerically reveal that such a denoising phenomenon is caused by approaching the cluster center for the dataset which has a clear cluster structure. Next, we construct a practical probabilistic distribution for training VAEs with a hyperbolic latent space against a hierarchical dataset. The hierarchical structure can be effectively embedded in hyperbolic space, and our approach enables us to incorporate uncertainty into the embedding. Last, we show that a meta-learning algorithm called model-agnostic meta-learning (MAML) is useful for representation learning of local structures.

This thesis aims to reveal why the information processing systems work well by focusing on the relationship between the structure behind the dataset and the learning algorithm or the model. Neural network studies are capable of using arbitrary differentiable computational graphs, allowing for a wide variety of experimental studies with this kind of scientific and engineering motivation. Therefore, our approach is widely applicable to other subjects of representation learning, and we expect that this thesis will be a part of such future intelligence research.

ACKNOWLEDGMENTS

First of all, I would like to express sincere appreciation to my supervisor, Professor Masato Okada, who has encouraged me throughout my graduate school days. Thanks to his perseverance and tremendous efforts, I was able to write this thesis. Besides my supervisor, I express my gratitude to the rest of my Ph.D. committee, Professor Hiroshi Kori, Associate Professor Taiji Suzuki, Associate Professor Issei Sato and Associate Professor Yasutoshi Makino for providing valuable comments on my drafts and presentation for this thesis.

My sincere appreciation also goes to my collaborator, Ph.D. Ryo Karakida, who gave a lot about academic research, beyond deep discussions on research. His theoretical insights into neural networks have had a major impact on my perspective.

I express my deep gratitude to my mentors and collaborators, Mr. Shoichiro Yamaguchi, Mr. Yasuhiro Fujita, Ph.D. Masanori Koyama, and Mr. Masaki Watanabe, during my internship at Preferred Networks, Inc. My time at Preferred Networks, Inc. has been a great experience in which I have been able to follow my own motivations for research and have had excellent mentors to discuss. Through my internship, I learned the importance of researching in a team, and I was able to expand my limits.

I am gratefully thankful to all members in Okada laboratory. There have been too many to count, but I would like to mention the names of Mr. Itsushi Sakata, M.D. Yuki Yoshida and Mr. Shiro Takagi. They gave me a chance to work together and made my daily life in the lab enjoyable. Thank you.

The enormous contribution of my mentors during my undergraduate years made me realize the fascination of fundamental research and gave me the knowledge and skills to carry out my research. These experiences led me to enter the Ph.D. program. In particular, I would like to thank Professor Shun Ishizaki, Associate Professor Atsushi Aoyama, Associate Professor Norifumi Watanabe, Assistant Professor Koji Iwayama, Lecturer Nao Tatsumi and Lecturer Kensuke Miyamoto.

Let me remark that my Ph.D. course was financially supported entirely by Japan Society for the Promotion of Science (JSPS), and partly by Masason Foundation. I was proud of working as a research fellow of these foundations. In particular, discussions with my friends at the Masason Foundation, including Mr. Liyuan Xu, Ph.D. Maho Shinoda, and Ms. Airi Takeuchi, brought me much more than just financial support.

Finally, I would like to express my profound gratitude to my family, especially for my parents Tadashi, Keiko, and my sister Kyoko. They have always trusted my choice and encouraged me. I have given a lot of trouble and burden to my family. I think words cannot express how grateful I am to my family. The accomplishment of this thesis would not have been possible without their selfless love.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION AND BACKGROUND

Machine learning is my favorite branch of philosophy.
"Experimental philosophy" if there's even such a thing.

— *David Ha* (*hardmaru*, 2019*a*,*b*)

Why and how can we think? What makes us intelligent? This is one of the ultimate questions in the study of artificial intelligence. This thesis attempts to approach representational learning of neural networks from both scientific and engineering perspectives as a first step towards answering the above challenging question. In particular, we focus on the structure of the external world, which is essential for the formation of the intelligent system. By studying the relationship between machine learning models and the structure of various datasets, we reveal the behavior of the system and suitable learning algorithms in some particular structures.

1.1 ARTIFICIAL NEURAL NETWORKS

There are many approaches from various fields for intelligence research. The most direct approach to the elucidation of biological intelligence would be the experimental sciences for animals and humans, such as neuroscience and cognitive science. In these approaches, researchers conduct experiments and data analysis based on the scientific motivation of understanding the behavior and neural activity of organisms and humans. On the other hand, research on artificial neural networks, which aims to model the brain, attempts to construct an abstracted mathematical model of the underlying mechanism and to explain complex phenomena with a small number of equations and parameter relationships (Dayan et al., 1995; Hebb, 1949; McCulloch and Pitts, 1943; Rosenblatt, 1958).

Research that originated in artificial neural networks has now grown into two major fields: computational neuroscience and machine learning. In the field of computational neuroscience, researchers have been working on the elaboration of mathematical models and the reflection of new experimental facts to construct biologically plausible mathematical models. On the other hand, in the field of machine learning, researchers use the simplified neural network models for engineering applications. Machine learning research focuses on proposing models and learning algorithms to solve practical problems in the real world. For this purpose, the explainability of biological systems and other phenomena in nature is not necessarily the goal. Such changes in constraints sometimes lead to significant research developments. One of these developments is deep learning.

Deep learning (LeCun, Bengio, and Hinton, 2015) is a general term for techniques that model machine learning problems using neural networks and train the model parameters by a gradient-based method. Research on artificial neural networks has started with the aim of intelligent information processing that mimics neural circuits of living organisms. In recent years, neural networks are widely used as approximators of nonlinear functions. These algorithms outperform standard machine learning techniques and humans in various practical applications such as image classification (Krizhevsky, Sutskever, and Hinton, 2012), audio generation (Oord et al., 2016), and agents of games (Mnih et al., 2015; Silver et al., 2017). The theoretical aspects behind them are also gradually being elucidated (Arora et al., 2018; Jacot, Gabriel, and Hongler, 2018; Poole et al., 2016; Schoenholz et al., 2017). Research in deep learning has progressed rapidly with the background of substantial computational resources and large datasets. Especially, a differentiable computational graph, which is the primary and common component of neural networks, makes it possible to interchange technologies that have been cultivated in various domains.

1.2 ADVANTAGES OF ARTIFICIAL NEURAL NETWORK RESEARCH

Most of the deep learning models and learning algorithms proposed in recent years no longer have the aspect of the imitation of biological systems. For example, while deep learning often uses back propagation-based optimization, there is still a debate on whether or not the biological brain is capable of performing back propagation (Lillicrap et al., 2020; Whittington and Bogacz, 2019). However, systems that do not necessarily have a strong resemblance to living organisms and can solve the same problems as biological organisms are rather highly useful for the study of intelligence. It is possible to compare the similarities with organisms such as humans, monkeys, and mice by considering a mathematical model as an experimental subject (Leibo et al., 2018; Marblestone, Wayne, and Kording, 2016; Yamins and DiCarlo, 2016). Revealing abstract compatibilities at the level of the phenomenon, rather than the hardware, has important implications for the understanding of intelligence. Also, if the computational models solve the problem in a completely different way than we organisms do, they reveal that the biological system may not be optimal for the problem. Deep learning is one of the most promising mathematical models for engineering research in modern machine learning and is a suitable subject for this motivation.

One of the advantages of using mathematical models as experimental subjects is their flexibility. There are limitations on the metrics that can be measured with human and animal subjects, and it is impossible to perform arbitrary operations on them from an ethical and technical point of view. Besides, it has been pointed out that it is challenging to analyze high-dimensional observations without knowing the guiding principles behind them (Jonas and Kording, 2017). In contrast, mathematical models such as neural networks enable us to conduct flexible experiments with arbitrary metrics. For example, there are studies that compare the internal representation of a trained network with the neural activity of an animal (Yamins and DiCarlo, 2016); compare representations between networks trained with different algorithms and/or architectures (Kornblith et al., 2019); examine the impact of removing a part of the network (Morcos et al., 2018); examine the behavior of the model in an environment similar to a cognitive experiment on an animal (Leibo et al., 2018). It is possible to take correspondence between the observed behavior and the principles behind it for a machine learning model since we know the learning algorithm that serves as the guiding principle. Also, deep learning research has the separability of objective functions, models, and learning algorithms, as will be discussed below. This property contributes to the flexibility of an experimental setup.

1.3 REPRESENTATION LEARNING WITH NEURAL NETWORKS

In this thesis, we focus on the representation learning with deep neural networks. One of the most notable features of deep learning is representation learning of complex datasets. The performance of machine learning methods is heavily dependent on the choice of data representation or features (Bengio, Courville, and Vincent, 2013). Standard machine learning techniques have focused on feature engineering: the design of preprocessing pipelines and data transformations. In contrast, deep learning allows us to automatically extract useful features for a task from data with only simple preprocessing. For instance, Le et al. (2012) showed that unsupervised learning of deep neural networks for unlabeled data allows high-level representations, such as those known as "grandmother cells" (Quiroga et al., 2005), to emerge.

Representation learning for unlabeled data can be considered in the following framework. Let $\boldsymbol{x} \in \mathcal{X}$ be a data and $g : \mathcal{X} \to \mathcal{Z}$ be a function that projects the data to a latent (representation) space. In machine learning, this function is expressed with some parameters ϕ. We can model this function g_ϕ by a neural network. Given $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}\}$ as a dataset, the goal of representation learning is to acquire a suitable model for the dataset by minimizing some objective function:

$$\min_{\phi} \mathcal{L}(g_\phi; \mathcal{D}). \tag{1.1}$$

In this case, the "goodness" of the model g_ϕ depends on the design of the objective function $\mathcal{L}(\cdot)$. Typically, it is necessary to design an appropriate training (optimization) algorithm to solve this minimization problem, depending on the objective function $\mathcal{L}(\cdot)$ and model g_ϕ. Appropriate design of training algorithms can reduce the computation time and guarantee convergence to the optimal solution. On the other hand, deep learning research uses gradient-based optimization methods regardless of these constraints. Although gradient-based training on models that are nonlinear to parameters is generally not guaranteed to be optimal, deep learning is empirically known to achieve good performance. In addition, its theoretical aspects have also been revealed in recent years. Since gradient-based optimization requires only gradients with respect to the parameters of Equation 1.1, we can use any differentiable computational graphs in g_ϕ. In other words, *we can manipulate the objective function or model independently without being aware of the training algorithm* in deep learning research. This feature has made it possible to interchange the techniques developed in various fields on the foundation of a differentiable computational graph. As a result, it leads to rapid progress in research and the achievement of engineering motivation.

In addition to the aforementioned engineering advantages, this feature has a significant contribution to a scientific studies. Since each element is independently interchangeable, it can serve as a basis for a reductive study of the contribution of each element. We mentioned above that models acquired by machine learning can be used as experimental subjects for intelligence research as well as living organisms. Examining the properties of well-performing models can lead to an understanding of factors that are important to intelligence. By treating the pair of datasets, models, and objective functions as some sort of sandbox environment, we can take an "understanding by construct" approach to intelligence research.

So *what is essential to acquire good representation?* A good representation should be one which does not include factors that affect subsequent tasks, such as noise and spurious correlation in the data (Arjovsky et al., 2019; Bengio, Courville, and Vincent, 2013; Tishby, Pereira, and Bialek, 2000). It is essential to have a model that represents the object with appropriate granularity. Although neural networks have the expressive power to be any nonlinear feature extractor, their capacity is too large without proper assumptions. For example, deep learning studies for natural images mostly use convolutional neural networks (Fukushima and Miyake, 1982; LeCun et al., 1989). Such a convolutional layer takes advantage of shift- and location-invariance in an image to reduce the model's search space. Other layers that exploit structural constraints of the data have been extensively studied. However, most deep learning research focuses on the correlation between dimensions of the data. Many algorithms typically assume the data distribution is i.i.d. On the other hand, real-world datasets often have specific correlation structures between data points. In such situations, the assumption of a structure between data points can provide a better representation.

1.4 STRUCTURE BEHIND DATASETS AND THE INFERENCE MODEL

An assumption on the structure behind the dataset is useful for a practical situation; it constrains the space to be explored and improves performance. Such an assumption is also important in terms of elucidating the mechanisms of intelligent information processing.

From the constraints of the external world, we can understand the reason why human information processing is as its current form. In this section, we summarize the structural constraints used for neural networks from two perspectives: dimension-wise and instance-wise structures.

1.4.1 *Dimension-Wise Structure*

A typical approach of structural constraints to datasets is interdimensional structure, which has been the subject of a tremendous amount of neural network research. A number of layers of networks have been proposed for high-dimensional datasets, assuming correlations between dimensions. The most standard layer, the fully-connected layer, imposes no assumptions about the inter-dimensional correlations of the input. The convolutional layer (Fukushima and Miyake, 1982; LeCun et al., 1989) is widely used for datasets where correlations can be assumed between adjacent dimensions, such as image and time series. In particular, for natural images, this layer reflects the empirical fact that adjacent pixels are correlated, and objects contained in the image have invariance to coordinate shifts. One-dimensional convolution can utilize temporal correlations of time-varying data, such as audio. Furthermore, there are various layers, such as recurrent connection, attention (Cho et al., 2014; Graves, 2013; Sutskever, Vinyals, and Le, 2014; Vaswani et al., 2017), and dilated convolution (Chen et al., 2014, 2017), to capture periodic or long-range correlations that cannot be achieved by convolution alone.

1.4.2 *Instance-Wise Structure*

The structure between data points has been widely studied in the field of manifold learning. Manifold learning is an approach that aims to assign a low-dimensional coordinate to a high-dimensional observation. The algorithms of manifold learning model the interinstance-wise structure mainly by adopting a non-parametric approach based on nearest neighbor graphs. A number of methods have been proposed to model nonlinear manifolds using a patchwork of locally flat tangent spaces (Brand, 2003; Roweis and Saul, 2000; Tenenbaum, Silva, and Langford, 2000; Vincent and Bengio, 2003). The locally linear embedding (Roweis and Saul, 2000) maps the original data point to a low-dimensional space by computing the weighted sum of its neighbor points. The Isomap (Tenenbaum, Silva, and Langford, 2000) constructs a distance matrix between data points using k-neighborhood graphs and embeds the data in a low-dimensional space with multidimensional scaling so that the distances are preserved.

Although many machine learning methods, including deep learning, make i.i.d. assumptions about data distribution, real-world datasets often have specific correlation structures. For instance, the object classification task is typically a problem for machine learning, and there is a hierarchical structure to the concept of things that humans perceive. Other than such an explicit situation, varying times and environments in which data are collected can affect their observations. Therefore, focusing on the structure between data points is useful for learning representations using neural networks.

1.5 RESEARCH QUESTIONS

This thesis's primary focus is the relationship between the structure of the external world and representation learning. Most machine learning studies assume that the dataset is obtained by i.i.d. In other words, consider the situation $p(\mathcal{D}) = \prod_i p(\boldsymbol{x}^{(i)})$. On the other hand, there are many different structures in the real world, and we cannot guarantee that such an assumption holds true. We can consider the following research questions in this situation.

QUESTION 1 As mentioned above, most models proposed in existing machine learning research assume i.i.d. for the data distribution. Then, how would the resulting model g_ϕ, trained under those assumptions, behave for a dataset with a particular structure? Are there any principles or unique properties in its behavior, or is there any correlation with the biological information processing?

QUESTION 2 We can assume a typical structure of a real-world dataset by narrowing it down to a certain domain. Can we design a model g_ϕ that is more appropriate than existing methods when assuming a specific structure for the data distribution $p(\mathcal{D})$?

QUESTION 3 Under an assumption of a particular structure for the data distribution $p(\mathcal{D})$, is it possible to consider a training algorithm that achieves better performance, without changing the model g_ϕ?

We study the above questions individually, assuming the structure of a typical dataset. If the structure of the external world is different, the information processing system suitable for it should be different. By studying inference models that assume the structure of a dataset, we aim to gain insight into the relationship between intelligent systems and the external world with which they are in contact.

In addition to the aforementioned questions, there is also room to consider an objective function suitable for the purpose. However, in this study, we follow the Bayesian principle to maximize the log likelihood. We also assumed several explicit structures as data distributions throughout the study. On the other hand, a good data distribution to replicate the real world is not inherently self-evident and could be a challenging and essential question. Although there is a possibility of approaching this issue using deep learning, we do not go into it in this thesis.

1.6 CONTRIBUTION OF THIS THESIS

As remarked before, deep learning studies less focused on a structure between data points than dimension-wise structure. The instance-wise structure has been incorporated by a non-parametric approach in the context of manifold learning. In this thesis, we present a numerical analysis and model proposal of a deep learning algorithm from the perspective of instance-wise structure. In particular, we discuss the relationship between instance-wise structures and their learning algorithms, not only from an engineering motive but also from a scientific one. Machine learning is a practical discipline that aims to solve real-world problems by finding the rules behind data rather than writing an explicit code. On the other hand, we emphasize that we can also consider machine learning and statistical learning problems as scientific objects. A model or algorithm that has been shown to be useful for a given task can itself capture a part of the nature of information processing for intelligent agents, including humans. Also, insights into objects revealed by scientific motivation lead to the discovery of practical engineering methods. As such, *both scientific and engineering motives can complement each other*. In Chapter 3, we investigate the behavior of a trained model from a scientific perspective, and in Chapter 4 and 5, we propose practical models from an engineering perspective.

The target of this thesis is representation learning for a dataset in which the structure between instances can be assumed. We focus on three types of instance-wise structures: cluster structure, hierarchical structure, and local structure.

In Chapter 2, we briefly explain the representative methods of training generative models with deep neural networks (called deep generative models). First, we explain the latent variable-based generative model, which is the typical graphical model for representation learning in neural networks. Then, we show three methods of deep generative models, called variational autoencoder (VAE), generative adversarial network, and normalizing flow. We summarize the properties of each model and explain that the VAE is suitable for our research motivation.

In Chapter 3, we verify the characteristics of the existing VAE inference for cluster structures from numerical experiments. VAE can remove noise from image data by using an encoder and a decoder. It is known that the noise is gradually reduced by repetition of encoding and decoding. However, the effect of this repeated inference is only empirically reported. In this chapter, we analyze the inference process of VAE numerically using a dataset with clustered structures that are widely available in the real world to understand the mechanism of such phenomena.

In Chapter 4, we propose useful probability distributions for datasets with a hierarchical structure and a representation learning method using them. Since there are many phenomena that have a hierarchical structure in nature, a generative model specific to this situation is valuable. In recent years, embedding hierarchical datasets into hyperbolic spaces has been a hot topic of research in the field of natural language processing. These studies provide a method to deterministically embed observations into hyperbolic space. In this chapter, we construct a probability distribution for embedding datasets in hyperbolic space with uncertainty and propose a deep generative model using this distribution.

In Chapter 5, we propose a training method for deep generative models that is useful for datasets with local structures. Here, we also discuss disentanglement under the assumption of local structure. Since many of the datasets targeted by machine learning are governed by the consistent rules of the physical world, it is reasonable to assume that high-dimensional observations are represented using a small number of control variables. Disentanglement representation, which aims to extract these control variables, has attracted a lot of attention in recent years and has been intensively studied. Although it has been shown that disentanglement representation cannot be achieved without some kind of inductive bias to the dataset or model, the specific form of inductive bias has not been actively discussed so far. In this chapter, we show that meta-learning can be a useful inductive bias by considering locality in the dataset.

Finally, we summarize these studies in Chapter 6 and discuss future prospects.

2 PRELIMINARIES: DEEP GENERATIVE MODELS

In this chapter, we first explain the latent variable-based generative model, which is the typical graphical model for representation learning in neural networks. Then, we briefly review the algorithms to train a generative model with neural networks, called deep generative models. After comparing the properties of each model, we explain the suitability of using the model called Variational Autoencoder for this study.

2.1 LATENT VARIABLE-BASED GENERATIVE MODELS

Conditional distributions with latent variables are often used when modeling high-dimensional data such as images, video, and audio. For example, observations of natural images take a pixel-by-pixel intensity vector, so the higher the resolution, the higher the dimension of the observation. In contrast, the content of an image can be represented by a small number of control parameters, obviously independent of resolution. In other words, we consider the probability distribution to be in a dimension smaller than the dimension of $\boldsymbol{x}$ (Rifai et al., 2011). Conceptually, let the data be $\boldsymbol{x}$, we give its probability distribution

$$\boldsymbol{x} \sim p(\boldsymbol{x}) \tag{2.1}$$

as

$$z \sim p(z), \tag{2.2}$$

$$\boldsymbol{x} \sim p(\boldsymbol{x}|z) \tag{2.3}$$

using the latent variable z. Once we get conditional distributions, we can calculate the posterior distribution

$$p(z|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|z)p(z)}{\int p(\boldsymbol{x}|z)p(z)\mathrm{d}z} \tag{2.4}$$

of the latent variable by Bayes' theorem. This posterior distribution can be used to estimate the low-dimensional control parameters from the high-dimensional data. Such an operation is called recognition, and its opposite, the inference of data from latent variables, is called generation (Figure 2.1). Note that, if the conditional distribution from z to $\boldsymbol{x}$ is complex, the integration of the denominator of Equation 2.4 is not analytically tractable in general. This is exactly the case with the parameterization using neural networks described below.

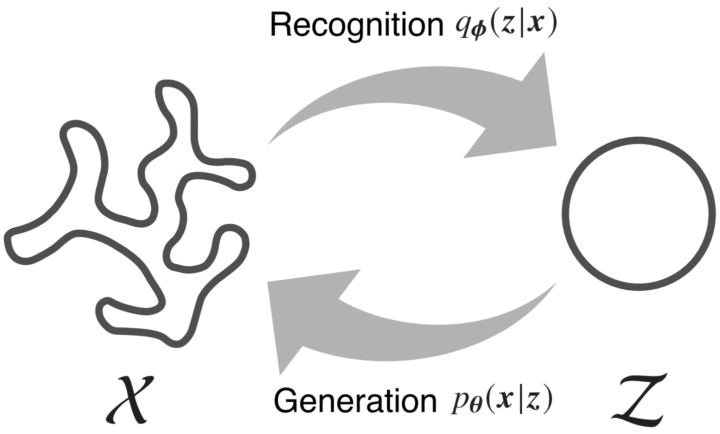


Figure 2.1: Schematic diagram of recognition and generation

Here the representation power of the entire model depends on the model and parametrization of $p(x|z)$. As mentioned earlier, real-world datasets are high-dimensional and complex, thus we require tools to model such probability distributions. Neural networks are widely used to model high-dimensional and nonlinear probability distributions. Models such as the Helmholtz machine (Dayan et al., 1995) and the restricted Boltzmann machine (Hinton, 2002; Smolensky, 1986) have been proposed so far, but they are known to be challenging to scale to large datasets. In the following, we describe methods for modeling conditional distributions using neural networks, with particular focus on those that have been developed in recent years.

2.2 PARAMETERIZING GENERATIVE MODELS WITH NEURAL NETWORKS

The deep learning technology has made it possible to learn a complex probability distribution with high dimensional observation and high nonlinearity. By using the deep generative model, we can generate large-scale datasets such as natural images (Brock, Donahue, and Simonyan, 2019; Kingma and Dhariwal, 2018; Oord and Vinyals, 2017) and audio (Oord et al., 2016) with high quality. Variational autoencoder (VAE) (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014) maximizes the variational lower bound of the log-likelihood. It extracts the latent representation of a dataset using an encoder that projects from the space of the data to the space of the latent variables and a decoder that gives an inverse mapping. Generative adversarial network (GAN) (Goodfellow et al., 2014) learns a probability density that spans in dimensions lower than the dimensions of the observation space by using likelihood-free adversarial learning. Normalizing flow (NF) (Dinh, Sohl-Dickstein, and Bengio, 2017; Rezende and Mohamed, 2015) allows us to train a complex probability distribution with an exact log-likelihood by using an invertible network and change-of-variable trick. Here, we briefly summarize the latent variable-based deep generative model.

2.2.1 *Variational Autoencoder*

VAE (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014) is one of the typical approaches to model the probabilistic distribution for high-dimensional observation. As described above, the simplest, and most common, graphical model with latent variables is one that is specified as factorization with the following structure (Kingma, 2017):

$$p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{z}) p_\theta(\boldsymbol{x}|\boldsymbol{z}). \tag{2.5}$$

By modeling the conditional distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ with a deep neural network, we can sample $\boldsymbol{x}$ efficiently even when the data space $\mathcal{X}$ is high-dimensional. However, training in such a model is challenging because of the intractability of the marginal distribution. When we optimize the model parater θ for the dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N$, we maximize the log-likelihood under an i.i.d. assumption:

$$\max_\theta \log p_\theta(\mathcal{D}) = \max_\theta \sum_{i=1}^N \log p_\theta(\boldsymbol{x}^{(i)}). \tag{2.6}$$

Although the posterior distribution $p_\theta(z|\boldsymbol{x})$ is necessary to calculate the above marginal distribution, this posterior distribution is generally intractable when a neural network is used for $p_\theta(\boldsymbol{x}|\boldsymbol{z})$.

VAE solves the above problem by introducing an additional parameter ϕ and approximate the posterior distribution of z to evaluate the lower bound on the variation of the log-likelihood.

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}^{(i)}) =& \log \int p_\theta(\boldsymbol{x}^{(i)}, z) \mathrm{d}z = \log \int p_\theta(\boldsymbol{x}^{(i)}|z) p_\theta(z) \mathrm{d}z \\
=& \log \int \frac{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})} p_\theta(\boldsymbol{x}^{(i)}|z) p_\theta(z) \mathrm{d}z \\
\geq& \int q_{\boldsymbol{\phi}}(z|\boldsymbol{x}) \log \frac{p_\theta(\boldsymbol{x}^{(i)}|z) p_\theta(z)}{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})} \mathrm{d}z \\
=& \int q_{\boldsymbol{\phi}}(z|\boldsymbol{x}) \left\{ \log p_\theta(\boldsymbol{x}^{(i)}|z) - \log \frac{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}{p_\theta(z)} \right\} \mathrm{d}z \\
=& \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(\boldsymbol{x}^{(i)} \mid z)\right] - D_{\mathrm{KL}}\left(q_\phi(z \mid \boldsymbol{x}^{(i)}) \middle\| p_\theta(z)\right) = \mathcal{L}(\theta, \phi; \boldsymbol{x}^{(i)}),
\end{aligned} \tag{2.7}$$

where the third line is derived through Jensen's inequality. In the above equation, $D_{\mathrm{KL}}(q\|p)$ is the Kullback–Leibler divergence of probability distributions q and p. The first term of the objective function corresponds to the reconstruction error, and the second term corresponds to the regularization. This objective function is called as the variational lower bound or the evidence lower bound (ELBO). In summary, we can evaluate the objective function without a true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$.

The VAE models conditional distributions $p_\theta(\boldsymbol{x}^{(i)}|\boldsymbol{z})$ and $q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})$ using neural networks: a decoder and an encoder. Therefore, the encoder gives a mapping of data, such as natural images to a latent variable space, and the decoder gives an inverse mapping. Conceptually, the conditional distribution with the encoder neural network are given by following equation:

$$\xi_e = \mathrm{encoder}_\phi(\boldsymbol{x}), \tag{2.8}$$

$$\boldsymbol{z} \sim p(\boldsymbol{z}|\xi_e), \tag{2.9}$$

and the same for the decoder network as:

$$\xi_d = \mathrm{decoder}_\theta(\boldsymbol{z}), \tag{2.10}$$

$$\boldsymbol{x} \sim p(\boldsymbol{x}|\xi_d), \tag{2.11}$$

where ξ_e and ξ_d represent the parameter for the probabilistic distributions. We typically model these probabilistic distributions by a simple distribution such as an independent Gaussian or an independent Bernoulli. The VAE minimizes the negative of the ELBO (Equation 3.1) according to two neural networks (Equation 2.8 and Equation 2.10) typically by using a gradient-based method such as stochastic gradient descent.

One of the difficulties in maximizing ELBO with a gradient-based approach is evaluating the gradient with respect to an encoder parameter ϕ.

$$\begin{aligned}\nabla_\phi \mathcal{L}(\theta,\phi;\boldsymbol{x}) &= \nabla_\phi \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - q_\phi(\boldsymbol{z}\mid\boldsymbol{x})\right] \\ &\neq \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\phi\left(\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - q_\phi(\boldsymbol{z}\mid\boldsymbol{x})\right)\right].\end{aligned} \tag{2.12}$$

Because the expectation itself also depends on the parameter ϕ, the second equation is not equal to the gradient, and the gradient can not be computed from the naive Monte-Carlo estimator. In VAE, the probability distribution of the encoder $q_\phi(\boldsymbol{z}\mid\boldsymbol{x})$ is modeled by using a parameter-free random variable ϵ and a deterministic transformation g.

$$\boldsymbol{z} = g(\epsilon,\phi,\boldsymbol{x}),\ \epsilon \sim p(\epsilon). \tag{2.13}$$

A simple Monte-Carlo estimator of the encoder's gradient is given by this modeling:

$$\begin{aligned}\nabla_\phi \mathcal{L}(\theta,\phi;\boldsymbol{x}) &= \nabla_\phi \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - q_\phi(\boldsymbol{z}\mid\boldsymbol{x})\right] \\ &= \nabla_\phi \mathbb{E}_{p(\epsilon)}\left[\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - q_\phi(\boldsymbol{z}\mid\boldsymbol{x})\right] \\ &\simeq \frac{1}{L}\sum_{l=1}^{L} \nabla_\phi\left(\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - q_\phi(\boldsymbol{z}\mid\boldsymbol{x})\right)\Big|_{\epsilon=\epsilon^{(l)}}\end{aligned} \tag{2.14}$$

where z in above equations depends on the actual realization of ϵ through g. Such a technique is called the *reparameterization trick*.

2.2.2 *Generative Adversarial Network*

VAE needs to give the probability distribution of observations as described above explicitly. Such modeling leads to support for the probability distribution of the data across all dimensions of observation. On the other hand, considering datasets such as natural images, the support of probability distributions is low-dimensional, and the observation process is not given by simple distributions such as Gaussian. GAN (Goodfellow et al., 2014) solves these problems by optimizing a likelihood-free objective function.

GAN framework is a training algorithm that estimates generative models via an adversarial process. In the GAN framework, we train two models simultaneously: a generative model G that maps the latent variable to the data space, and a discriminator D that estimates whether each sample comes from the dataset or the model. Compared with the conventional algorithms, this method receives attention since high-fidelity samples can be obtained, particularly in image generation task. GAN optimizes the generator G and the discriminator D through the following two-player minimax game:

$$\min_G \max_D V(D,G) = \min_G \max_D \left\{ \mathbb{E}_{\boldsymbol{x}\sim P(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \right\}. \quad (2.15)$$

In the approach of GAN, the generated samples are computed by the deterministic generator $\boldsymbol{x} = G(\boldsymbol{z})$, so the randomness only comes from the latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$. Because of the construction, the samples generated by the generator are distributed at most in the dimensions of the latent variables.

2.2.3 *Normalizing Flow*

While VAE and GAN can model probability distributions of high-dimensional observational data, they do not have explicit log-likelihood. One needs to be careful to apply these methods to situations where log density values are useful, such as anomaly detection. On the other hand, NF (Dinh, Sohl-Dickstein, and Bengio, 2017; Rezende, Mohamed, and Wierstra, 2014) provides a model in which the log-likelihood can be rigorously evaluated using a deep neural network. In NF, $\boldsymbol{x}$ is generated from the random variable $\boldsymbol{z}$ via a deterministic map, similar to GAN. In this case, NF limits the transformations of each layer to those that are invertible. Let the transformation of ℓ-th layer be $f_\ell\colon \mathbb{R}^d \to \mathbb{R}^d$, the overall network is

$$\boldsymbol{x} = z_L = f_L \circ \cdots \circ f_2 \circ f_1(z_0). \quad (2.16)$$

Now, between the invertible random variables a, b in which $b = f(a)$ holds, $p(b) = p(a)|\det \partial f / \partial a|^{-1}$ holds. Using this change-of-variables law, the logarithmic density of the data in the observation space can be expressed as follows.

$$\log p(\boldsymbol{x}) = \log p(z_0) - \sum_{\ell=1}^{L} \log \left| \det \frac{\partial f_\ell}{\partial z_{\ell-1}} \right|. \tag{2.17}$$

NF is a method to optimize the objective function given in the above equation by calculating the gradient for each parameter of each layer. The log-determinant of the Jacobian for each layer is required to evaluate the log density, but the log-determinant of the high-dimensional Jacobian is generally computationally expensive. A number of layers have been proposed that have expressive power and its log-determinant can be easily evaluated (Behrmann et al., 2019; Dinh, Sohl-Dickstein, and Bengio, 2017; Hoogeboom, Van Den Berg, and Welling, 2019; Kingma and Dhariwal, 2018; Kingma et al., 2016; Tomczak and Welling, 2016, 2017).

As mentioned above, we reviewed a representative latent variable-based deep generative models. The characteristics of each model are as follows. VAE can encode data into

Table 2.1: Comparison of typical deep generative models

	VAEs	GANs	NFs
Tractable density	lower bound	✗	✓
Inference availability	✓	✗	✓
Dimensionality reduction	✓	✓	✗

latent variables and reduce the dimension of latent variables with respect to the observed dimension. This is a useful property for representation learning. On the other hand, the objective function of VAE is not an exact log-likelihood, but a variational lower bound of the log-likelihood. GAN generates high-dimensional observations from low-dimensional latent variables, but there is no encoder network, so it cannot extract the latent representation from observation. Also, since it is a likelihood-free learning method, we need to use some kind of metrics such as inception score (Salimans et al., 2016) or Fréchet inception distance (Heusel et al., 2017) to assess the model. NF takes a slightly different approach from these models. Since NF uses the variable transformation of the integral as described above, the dimension of the latent variable must be the same as the observed dimension. Due to this constraint, NF cannot perform dimensionality reduction for high-dimensional datasets such as images, which leads to an increase in the model size. Since we aim at representational learning, a model that has an encoder network and is capable of dimensionality reduction is suitable. For the above reasons, we conducted this study using VAE as a basic tool.

3

INFERENCE DYNAMICS OF VAES AGAINST CLUSTER STRUCTURE

Deep neural networks are good at extracting low-dimensional subspaces (latent spaces) that represent the essential features inside a high-dimensional dataset. Deep generative models represented by variational autoencoders (VAEs) can generate and infer high-quality datasets, such as images. In particular, VAEs can eliminate the noise contained in an image by repeating the mapping between latent and data space. To clarify the mechanism of such denoising, we numerically analyzed how the activity pattern of trained networks changes in the latent space during inference. We considered the time development of the activity pattern for specific data as one trajectory in the latent space and investigated the collective behavior of these inference trajectories for many data. Our study revealed that when a cluster structure exists in the dataset, the trajectory rapidly approaches the center of the cluster. This behavior was qualitatively consistent with the concept retrieval reported in associative memory models. Additionally, the larger the noise contained in the data, the closer the trajectory was to a more global cluster. It was demonstrated that by increasing the number of the latent variables, the trend of the approach a cluster center can be enhanced, and the generalization ability of the VAE can be improved.

3.1 INTRODUCTION

Research on deep generative models, which extract essential features from an unlabeled dataset, is currently an active research area. Deep generative models have been reported to be useful in a broad range of applications, including generating images (Goodfellow et al., 2014; Kingma and Welling, 2013; Radford, Metz, and Chintala, 2015; Rezende, Mohamed, and Wierstra, 2014), movies (Saito, Matsumoto, and Saito, 2017; Vondrick, Pirsiavash, and Torralba, 2016; Walker et al., 2016), and text (Li et al., 2017; Serban et al., 2017; Yu et al., 2017). In particular, the conventional bidirectional network structure for the recognition and generation of images has made it possible to eliminate noise from cluttered images and smoothly interpolate between different images. Recognition is the process of mapping a data point to a latent variable, and generation is the inverse of this process.

Several studies have qualitatively highlighted the importance of repeating inferences between data space and latent space multiple times (Arulkumaran, Creswell, and Bharath, 2016; Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014). In the present study, repeated inferences are defined as a process by which a deep generative model repeats the recognition and generation of images. It was shown that by using noise-containing images as initial values, deep generative models can eliminate noise by repeating recognition and generation several times (Rezende, Mohamed, and Wierstra, 2014). Moreover, compared to generating an output image from latent space to smoothly morph one image into another, repeating inferences several times improves the quality of the output image (Arulkumaran, Creswell, and Bharath, 2016). However, most of these studies only qualitatively evaluate output data through a one-shot inference from the latent space to output data. To fill this gap in the literature, we quantified the dynamics of repeated inferences to investigate why repeating inferences are effective for a wide range of applications.

In many cases covered by deep generative models, the data distribution is concentrated in the low-dimensional sparse subspace of the high-dimensional observation space. For example, in the case of natural image datasets, most of the space formed by the entire image corresponds to an image in which each pixel value is randomly chosen, but it is not plausible for natural images. The deep generative models extract a low-dimensional subspace in such a high-dimensional space by nonlinear mapping using neural networks. Because various factors, such as noise in real environments, cause original data points to deviate from this low-dimensional subspace, we are interested in *how the dynamics of the activity pattern during the inference phase is drawn into the subspace formed by the original training data*. In this study, we clarify how the activity pattern during inference in deep generative models approach the subspace where data are concentrated. In particular, we focused on the dataset that has a cluster structure, which is typically seen in image generation tasks and exists widely in nature.

We numerically analyzed the collective behavior of repeated inferences in a variational autoencoder (VAE) (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014), which is a typical type of deep generative model. We used the Modified National Institute of Standards and Technology (MNIST) dataset and Fashion-MNIST dataset, which are considered to have cluster structures consisting of 10 types of labels. We input noise-containing images to the trained VAE as initial values, and we numerically analyzed the transition of the activity patterns in the data space and the latent space. In particular, we calculated the time evolution of the distance to the vector in the latent space corresponding to quantify how the activity patterns approach the subspace of training data points, and we calculated the time development of the distance between the activity patterns in the latent space and the center of the clusters.

There are three major findings in our study. First, we numerically demonstrated that the dynamics of repeated inferences rapidly approach a center of the cluster in the latent space. Such transient behavior cooccurred with the perceptual refinement of the generated images in the data space. Second, by averaging all of these centers in the latent space, we considered the center of the cluster centers; by definition, training patterns, cluster centers, and the center of the cluster centers are hierarchically related in ascending order. We found that the inference dynamics approach the center of the cluster centers to the extent that the uncertainty of the input data increases due to noise. This result suggests that the model selects appropriate inference strategies in accordance with the fraction of the noise added to the input data. Third, we examined the effect of the latent variable dimension on inference behavior. As the number of latent variables increases, the internal representations of the clusters tend to become orthogonal, and the dynamics of repeated inferences approach each corresponding center. The generalization performance of the model was improved to the extent that the center of the cluster attracts the dynamics of repeated inferences. We also discuss the practical insight into the optimal number of inference steps from our experimental findings.

3.2 METHOD

Here, we first describe the summary of the VAE. Then, we describe the network architecture which we used in numerical experiments and how to train the VAE. Last, we describe about an inference procedure.

3.2.1 *Variational Autoencoder*

A VAE is a generative model consisting of two neural networks: an encoder and a decoder (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014). An encoder gives a mapping of data, such as natural images to a latent variable space, and the decoder gives an inverse mapping. The objective function of the VAE is obtained by finding the variational lower bound of log-likelihood $\sum_i \log p(\boldsymbol{x}^{(i)})$ for N training data $X = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$. In the following, we consider a parameter $\boldsymbol{\theta}$ that maximizes the log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$ at each data point. Using the latent variable $\boldsymbol{z}$ and its conditional probability distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}^{(i)})$ and taking the variational lower bound of the log-likelihood gives the following objective function:

$$\begin{aligned} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) &\geq -D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}^{(i)}) \middle\| p(\boldsymbol{z})\right) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \mid \boldsymbol{z})\right] \\ &= \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}). \end{aligned} \tag{3.1}$$

In the above equation, $p(\boldsymbol{z})$ is the prior distribution of latent variable $\boldsymbol{z}$, and $D_{\mathrm{KL}}(q\|p)$ is the Kullback–Leibler divergence of probability distributions q and p. The first term of the objective function corresponds to the regularization, and the second term corresponds to the reconstruction error. The VAE models conditional distributions $p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \mid \boldsymbol{z})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}^{(i)})$ using neural networks. To optimize parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ by backpropagation, samples were generated using a method called *reparameterization trick* with encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}^{(i)})$. The latent variable is modeled as follows:

$$\begin{aligned} \boldsymbol{z} &= g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \boldsymbol{x}) \\ &= \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \end{aligned} \tag{3.2}$$

to decompose $\boldsymbol{z}$ into random variable $\boldsymbol{\epsilon}$ and deterministic variables $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, where $\odot$ indicates Hadamard–Schur product. Giving $\boldsymbol{\epsilon}$ as a sample from the standard Gaussian distribution eliminates the need for a complicated integral during training. If the above conditions are assumed and the expected reconstruction error $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \mid \boldsymbol{z})\right]$ is approximated by a sample average, Equation 3.1 can be rewritten as follows:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}) \simeq \frac{1}{2}\sum_{j=1}^{N_z}\left(1 + \log\left((\sigma_j^{(i)})^2\right) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \mid \boldsymbol{z}^{(i,l)}). \tag{3.3}$$

$\boldsymbol{\phi}$ parameterizes the outputs of the encoder $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Both parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ were trained by the gradient ascent method to maximize Equation 3.3. The output of the decoder was set as the probability of the Bernoulli distribution, and the expectation of the conditional probability, namely, the second term of the objective function, was approximated by averaging L samples.

3.2.2 *Network Architecture and Optimization Procedure*

Since our research focuses on the behavior of VAE inference, we need to reduce dependence on network structure as much as possible. Based on this motivation, we used a separate three-layer, fully connected neural network for the encoder $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}^{(i)})$ and decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \mid \boldsymbol{z})$ mentioned above. The fully connected neural network is consists of a fully connected layer:

$$h_i^l = \text{activation}\left(\sum_{j=1}^{\#\,\text{units}} W_{ij}h_j^{l-1} + b_j\right). \tag{3.4}$$

h_i^l expresses the i-th unit (neuron) of the l-th layer, and $\left\{W_{ij}\right\}$ and $\left\{b_j\right\}$ are the parameters of each layer. The input to the first layer of the encoder $\boldsymbol{h}^0 = \left[\cdots, h_j^0, \cdots\right]^{\mathrm{T}}$ corresponds to a data point $\boldsymbol{x}$, and the output of the encoder $\boldsymbol{h}^2$ corresponds to $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma}^2$ in Equation 3.2 and 3.3. Likewise, the input and the output of the decoder correspond to $\boldsymbol{z}$ and $\boldsymbol{x}$, respectively. The number of units in the middle layer was set to 1,024, and the activation function was set as tanh:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{3.5}$$

We used the sigmoid function

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{3.6}$$

for the activation function of the decoder's last layer to normalize the output of the model to $[0, 1]$. The number of units of latent variable N_z was set to 100 unless otherwise noted.

We used Adam (Kingma and Ba, 2014) as the parameter optimization algorithm. Adam updates the model parameter θ according to the objective function f with the following equations:

$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}), \tag{3.7}$$

$$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t, \tag{3.8}$$

$$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2, \tag{3.9}$$

$$\hat{m}_t \leftarrow m_t/(1-\beta_1^t), \tag{3.10}$$

$$\hat{v}_t \leftarrow v_t/(1-\beta_2^t), \tag{3.11}$$

$$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon). \tag{3.12}$$

The lower script t in the above equations expresses the time step of the optimization. We note that this time step is for the training phase and is not for the inference phase. $g, m, v, \hat{m}$, and $\hat{v}$ are the intermediate variables to compute the parameter θ of the next time step. α, β_1, β_2, and ϵ in the above equations are the hyperparameters for the optimization. α is the learning rate or step size of the optimization. β_1 and $\beta_2 \in [0, 1)$ control the exponential decay rates of moving averages. We set β_1, β_2, and ϵ to the default values 0.9, 0.999, and 10^{-8}. The learning rate α was reduced in descending order as follows: 0.0005, 0.0001, and 0.00005. In our setup, we used the variational lower bound of the log-likelihood (Equation 3.3) as the objective function, and the parameters of the encoder $\boldsymbol{\phi}$ and the decoder $\boldsymbol{\theta}$ are the target to optimize. We set the number of samples for calculating the reconstruction error to $L = 2$. We trained the VAE against the MNIST dataset for 1,500 epochs. The MNIST dataset consists of 28×28-pixels '0'–'9' handwritten images with 60,000 training data and 10,000 test data. These data are considered to have cluster structures consisting of 10 types of labels, namely, '0'–'9'.

3.2.3 *Inference Procedure and Noise Injection*

Here, we describe how to perform repeated inference. In this study, noisy MNIST data were inferred using the trained network according to the following procedure, and the time evolution of latent variable $\boldsymbol{z}(t)$ was obtained. First, noise was added to an image of the training dataset. Pixels with probability p were selected from 784 pixels, the image intensities of the selected pixels were swapped, and the image was set as $\boldsymbol{x}_0$. Next, the data variable in step

$t = 0$ was taken as $x(0) = x_0$. Finally, generation and recognition were repeated T times according to the following two equations:

$$x(t+1) = \mathbb{E}_{p_\theta(x|z(t))}[x], \tag{3.13}$$
$$z(t) = \mathbb{E}_{q_\phi(z|x(t))}[z], \tag{3.14}$$

to obtain the time evolutions of data variable $x(t)$ and latent variable $z(t)$. We modeled the observation process by an independent Bernoulli distribution, so the output of the decoder corresponds to the expected value of x:

$$x(t+1) = \mathbb{E}_{p_\theta(x|z(t))}[x] = \text{decoder}_\theta(z(t)). \tag{3.15}$$

Also, since we used the Gaussian encoder as we mentioned above, the first half of the output of the encoder (μ) is the expected value of z. Therefore,

$$z(t) = \mathbb{E}_{q_\phi(z|x(t))}[z] = \mu_\phi(x(t)). \tag{3.16}$$

We expressed the first half of the encoder's output as $\mu_\phi(\cdot)$. We note that we do not need any approximation to compute the expected values thanks to the model definition. The dynamics of $x(t)$ and $z(t)$ were numerically analyzed. The randomness of these dynamics $x(t)$ and $z(t)$ only comes from the randomness of the input $x(0)$.

3.3 RESULTS

3.3.1 *Dynamics of inference trajectory: An approach to cluster centers*

First, we show that the dynamics of the latent space activities in VAEs are rapidly drawn into a low-dimensional subspace. In this study, we define the following operations as repeated inferences:

$$x(t+1) = \mathbb{E}_{p_\theta(x|z(t))}[x], \tag{3.17}$$
$$z(t) = \mathbb{E}_{q_\phi(z|x(t))}[z], \tag{3.18}$$

where $q_\phi(z \mid x)$ is the encoder network that maps x to the latent space activity z, and $p_\theta(x \mid z)$ is the decoder network that gives the inverse mapping. First, we add the noise with noise fraction p to an image of the training dataset, and we set the image as $x(0) = x_0$. By repeating the above two equations T times, we obtain the trajectory of the repeated inference on data space $x(t)$ and latent space $z(t)$. In this study, we call the processes that gradually infer the plausible image by the above update rule as repeated inferences. We numerically analyze the dynamics of the activity patterns in data space $x(t)$ and latent space $z(t)$ from the qualitative/quantitative point of view.

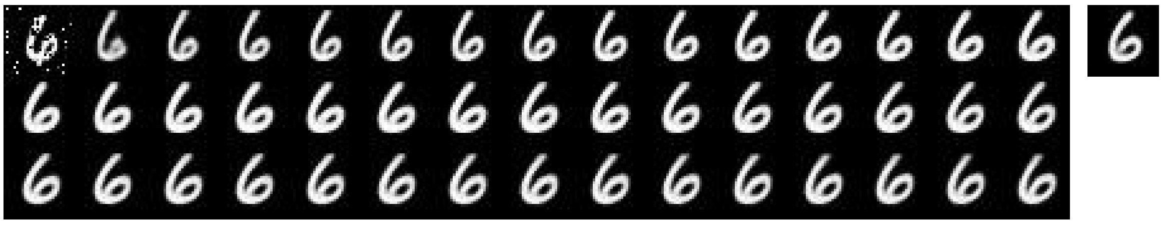


Figure 3.1: Consecutive samples in the data space (from left to right, one row after the other). The image of '6' with $p = 0.2$ noise applied was used as the initial value. The image $\mathbb{E}_{p_\theta(x|\bar{\xi}_6)}[\boldsymbol{x}]$ generated by the concept vector $\bar{\xi}_6$ is shown on the right.

Figure 3.1 expresses the consecutive samples of the repeated inference in the data space. We used the MNIST database as the training dataset. The time development of the activity pattern in data space $\boldsymbol{x}(t)$ is aligned from left to right, one row after the other. The upper-left image corresponds to the initial value, $\boldsymbol{x}_0$. The image of '6' with $p = 0.2$ noise applied was used as the initial value. From the figure, the VAE removes the noise contained in the image in the first few steps and then gradually shifts to the specific image of '6'. We also show the consecutive samples for another image in Section 3.A.

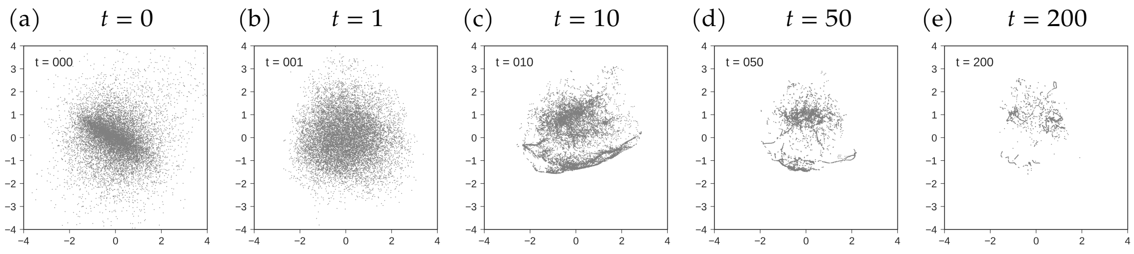


Figure 3.2: (a–e): PCA visualization of the VAE's latent activity patterns. The x- and the y-axes represent the first and second principal components. Each figure corresponds to the snapshot of the activity patterns at time t. We used a different image of '1' with a different noise realization as the initial value for each trial. The noise faction was set to $p = 0.2$.

Then, we visualize the time development of the latent space activities during the repeated inferences. The temporal evolution $\boldsymbol{z}(t)$ of the latent activity pattern for one initial image can be regarded as a trajectory in the latent space. Figure 3.2 shows the collective behavior of these trajectories in the latent space for multiple images. We used a different image of '1' with a different noise realization as the initial value for each trial. We embedded the activity patterns in latent space into two dimensions using the principal component analysis (PCA). Because PCA has the degree of freedom of rotating eigenvectors, we performed PCA for the latent activity patterns included within $[t, t+\Delta t]$ for every t to stabilize the eigenvectors. Let $Z(t)$ be the matrix that collects the activity patterns of the latent space within $[t, t+\Delta t]$:

$$Z(t) = \begin{bmatrix} | & & | & & | & & | \\ z^{(1)}(t) & \cdots & z^{(N)}(t) & \cdots & z^{(1)}(t+\Delta t) & \cdots & z^{(N)}(t+\Delta t) \\ | & & | & & | & & | \end{bmatrix}^{\top} \in \mathbb{R}^{N\Delta t \times N_z}. \tag{3.19}$$

We derived the matrix of eigenvectors $Q(t) = \left[\boldsymbol{q}_1(t), \cdots, \boldsymbol{q}_{N_z}(t)\right]$ for each time step:

$$Z(t)^{\top} Z(t) = Q(t)\Lambda(t)Q(t)^{\top}, \tag{3.20}$$

and embedded the activity patterns in the two largest eigenspace as $\left[\boldsymbol{q}_1(t), \boldsymbol{q}_2(t)\right]^{\top} z^{(i)}(t)$. Each point in the figure corresponds to the latent activity pattern for the specific initial image. The x- and the y-axes represent the first and second principal components. At the beginning of the inference $t = 0$ (Figure 3.2a), the latent activities were widely distributed as one large cluster. These activity patterns branched into two clusters at $t = 10$ (Figure 3.2c). The first cluster was widely distributed in the upper part of the figure, and the second cluster formed a string-like distribution concentrated in the lower part of the figure. After that, the activity patterns converged to individual points or string-like regions.

We show the visualization of the perceptually generated images during the repeated inferences in Figure 3.3. Each figure (from Figure 3.3a to Figure 3.3e) corresponds to the two-dimensional PCA embeddings. The x- and the y-axes represent the first and the second principal components. At every time step t, the activity pattern $z^{(i)}(t)$ for i-th initial image specifies the coordinate in the principal component space. We plotted the inferred images $\mathbb{E}_{p_\theta(x|z^{(i)}(t))}[x]$ for the latent activity patterns in these coordinates. We cropped the x- and the y-axes in the range of $[-2, 2]$ for clarity.

At the initial phase (Figure 3.3a and Figure 3.3b) of repeated inferences, the inferred images moderately included some noise. At the time step $t = 10$ (Figure 3.3c), the crowd of the activity patterns branched into two clusters. We found that the upper half of these two clusters corresponded to the trials where the inferred images deviated from '1'. The lower half of the cluster shaped like a string expressed various angles of '1' smoothly. The angle of the inferred image of '1' gradually shifted from left to right. These continuously distributed angles converged to several specific values at the end of the inference (Figure 3.3e). The crowd of these activity patterns branched to multiple string-shaped clusters and slowly converged to points. These results suggest that there are multiple saddle points, which have stable and unstable directions, in the latent space of trained VAEs. It is indicated that the data points which typically appear in the training dataset became stable fixed points and the lines between these points appeared as string-shaped subspaces.

From the aforementioned results, we numerically clarified that the latent activity patterns gradually branch into several clusters during repeated inference. Specifically, the collective

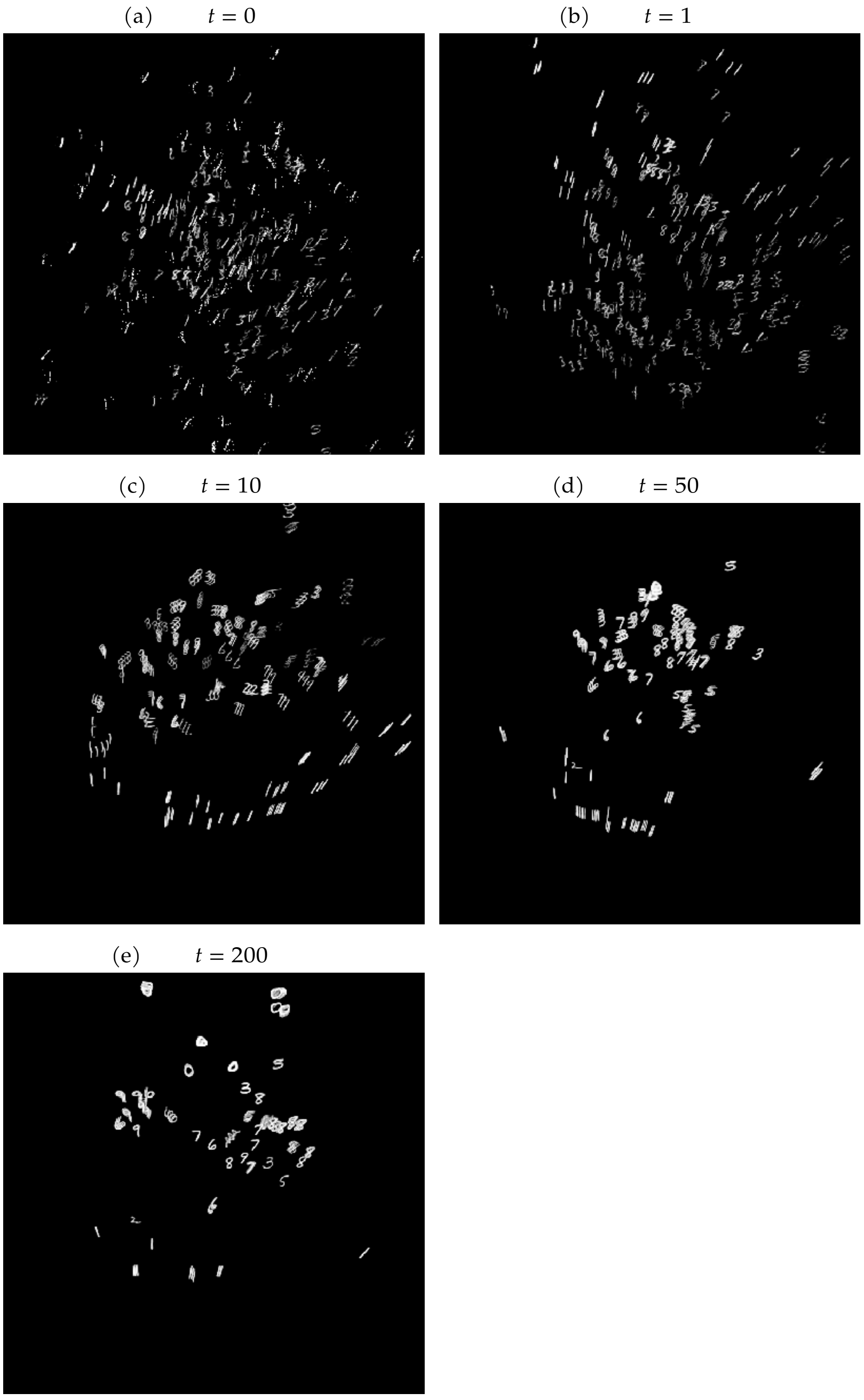


Figure 3.3: (a–e): The inferred images that correspond to the dynamics of activity patterns in latent space. The x- and y-axes indicate the first and the second principal components. Each figure shows the snapshot of the activity patterns at the specific time step from $t = 0$ to $t = 200$. The x- and the y-axes are cropped in the range of $[-2, 2]$ for clarity.

behavior rapidly approached the low-dimensional subspace near $t = 10$. We numerically quantify this type of approaching behavior. In the following, we assume that the dataset is composed of one cluster for each label for the clarity of numerical analysis. We now evaluate the distance between latent activity patterns and the center of the clusters.

$$\bar{\xi}_{\mathrm{num}} = \frac{1}{N_{\mathrm{num}}} \sum_{i}^{N_{\mathrm{num}}} \xi^{(i)}_{\mathrm{num}}, \tag{3.21}$$

where $\xi^{(i)}_{\mathrm{num}}$ indicates the activity pattern of the latent variable for the i-th training data with label num:

$$\xi^{(i)}_{\mathrm{num}} = \mathbb{E}_{q_\phi(z|x^{(i)}_{\mathrm{num}})}[z]. \tag{3.22}$$

This definition is known as the mathematical quantity called "concept", in studies on associative memory models (Amari, 1977; Matsumoto et al., 2005). The attraction of the activity patterns of neural networks into subspaces has been mainly studied with associative memory models (Amari, 1977; Matsumoto et al., 2005). In the problem setting of Matsumoto et al. (2005), they first randomly generated a small number of concept patterns, and then they made memory patterns with a precise correlation with these concept patterns. In other words, the concept pattern vector corresponds to the center of each cluster. We call $\bar{\xi}_{\mathrm{num}}$ a concept vector and $\xi^{(i)}_{\mathrm{num}}$ a memory vector for the i-th training data in the following sections. The relationship between the time development of inference and the concept vector of each label ('0'–'9') represented in the MNIST data was numerically analyzed.

We show the image $\mathbb{E}_{p_\theta(x|\bar{\xi}_6)}[x]$ generated by the concept vector of '6', $\bar{\xi}_6$ on the right side of Figure 3.1. By qualitatively comparing this image and the consecutive samples, it is suggested that the result of the VAE inference approaches the image generated by $\bar{\xi}_6$ once. Here, we define a trajectory "approaching to a concept vector" as follows: the trajectory whose distance to the concept vector takes a minimum at a unique halfway point and is closer than its synthetic linear interpolation. We quantitatively evaluated the gradual changes of the Euclidean distance; namely,

$$\left\| z(t) - \bar{\xi}_{\mathrm{num}} \right\|_2, \tag{3.23}$$

between the neural activity patterns and the cluster center for every label of MNIST data in the latent space (Figure 3.4a). The distance between the cluster center and 300 different initial images was calculated. Each figure corresponds to each label, which was used as initial input for the VAE. The x-axis expresses the time step t of repeated inference, and the y-axis expresses the Euclidean distance (Equation 3.23). It was clarified that the trajectory of the VAE's inference rapidly find the cluster structure. This result is qualitatively consistent with all labels of the MNIST data. A previous study using associative memory models (Matsumoto et al., 2005) reported that activity patterns approached the concept vector once in the middle of inference when the inference was started from data with noise applied to each memory pattern. Figure 3.4b shows the same figure for the distance between the activity patterns and the memory vector. As same for the concept vector, the

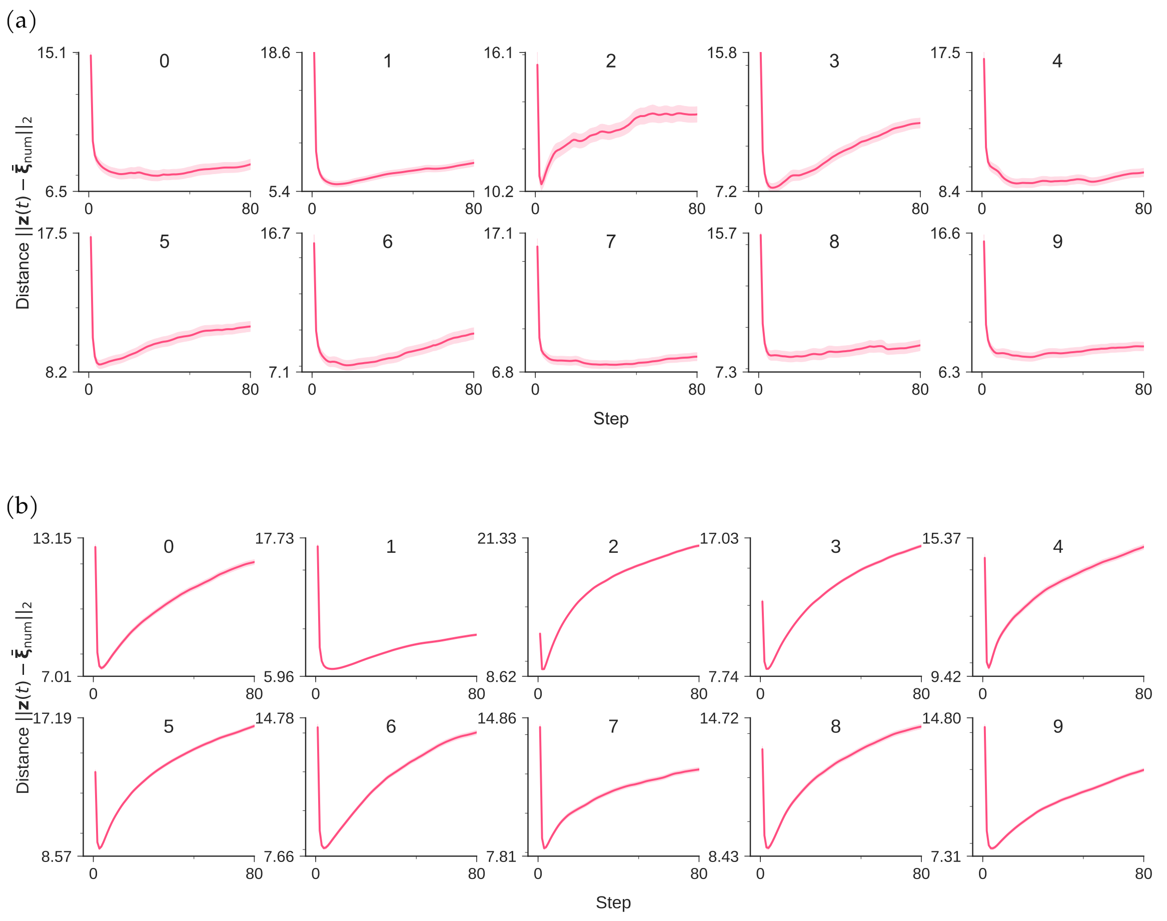


Figure 3.4: Time development of the Euclid distance for all labels of the MNIST data. The distances from $\bar{\xi}_{\text{num}}$ are shown in (a) and the distances from $\xi^{(i)}_{\text{num}}$ are shown in (b). The shades represent the ± 1 standard error of the mean (300 trials). We used a different image of each label with a different noise realization as the initial value for each trial. All figures were generated with the noise fraction $p = 0.2$.

activity patterns were closest to the memory vector early in the inference. If the concept vector is a stable fixed point, the activity patterns should monotonically approach the concept vector. In other words, these results suggest that the latent space of the trained VAE has a saddle point structure that attracts in the direction to which noise is applied and diverges in the orthogonal direction. The results obtained in this study were qualitatively consistent with these previous findings. We also performed the same numerical experiment on the Fashion-MNIST dataset (Xiao, Rasul, and Vollgraf, 2017), which is a dataset of Zalando's article images consisting of various fashion images. The numerical results for the Fashion-MNIST dataset were also qualitatively consistent with those for the MNIST dataset. Please see Section 3.B for more details.

3.3.2 *Relationship between data hierarchy and inference*

We arbitrarily determined the amount of noise added to the initial input images in the previous section. To examine the effect of noise on the dynamics of repeated interferences, we then modulated the amount of noise. Because noise in input images causes the data to deviate from the original distribution, we created another class, the "abstract concept vector" for convenience, which averages all the labels' concept vectors, as well as the concept and memory vectors. By measuring the distance between the trajectory of each neural activity pattern and its corresponding classes in the latent space, we identified the class that most attracts the neural activity patterns.

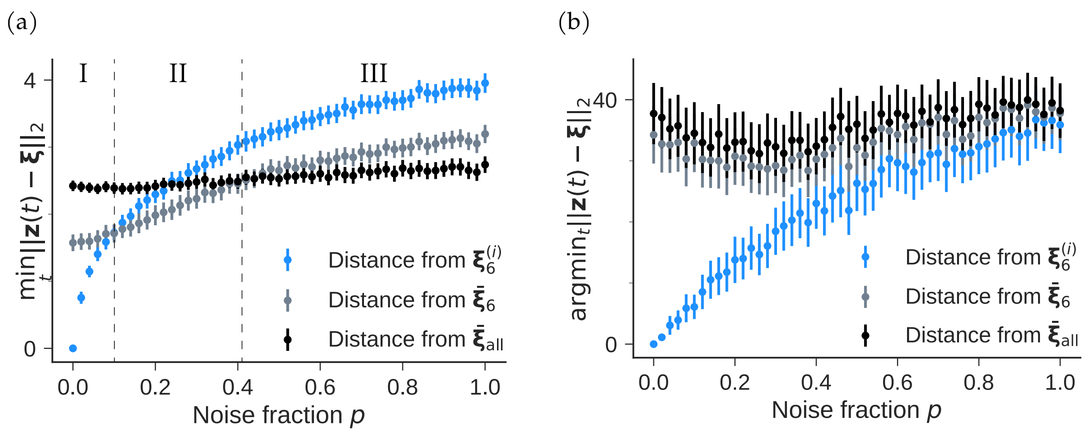


Figure 3.5: (a): Minimum distances from concepts according to noise fraction p. The bars represent the ± 2 standard error of the mean (500 trials). We used a different image of '6' with a different noise realization as the initial value for each trial. (b): Time step of which the distance takes a minimum.

We define the "abstract concept vector" as

$$\bar{\xi}_{\text{all}} = \frac{1}{10} \sum_{\text{num}=0}^{9} \bar{\xi}_{\text{num}}. \tag{3.24}$$

The three classes (memory vectors, concept vectors, and the abstract concept vector) are in a hierarchical relationship from detailed to coarse information in the order $\xi^{(i)}_{\text{num}}$, $\bar{\xi}_{\text{num}}$, and $\bar{\xi}_{\text{all}}$. Note that the abstract concept vector is not a useful representation from the practical viewpoint. We chose this metric as an anchor to unveil the dynamics of repeated inferences. We calculated the minimum distances between neural activity patterns $z(t)$ and corresponding classes,

$$\min_t \|z(t) - \xi\|_2. \tag{3.25}$$

Figure 3.5a shows the minimum distances according to the noise fraction. In Figure 3.5a, the x-axis represents noise fraction p, which is the probability that the image intensities of the pixels are swapped. For every noise fraction, the minimum distances between the firing pattern $z(t)$ and hierarchical concept vectors were calculated by changing the initial image 500 times. The dots in the figure express the mean of the minimum distance, and the bars are the ± 2 standard error of the mean (500 trials). We divided the parameter regions into three stages, I, II, and III, corresponding to the minimum distance between the firing pattern $z(t)$ and hierarchical concept vectors, $\xi^{(i)}_6$, $\bar{\xi}_6$, and $\bar{\xi}_{\text{all}}$, respectively. In stage I, the firing activity was closest to the original pattern $\xi^{(i)}_6$ with a small amount of noise. Interestingly, the closest class was $\bar{\xi}_6$ with moderate noise in stage II. The activity was close to the abstract concept vector $\bar{\xi}_{\text{all}}$ in stage III. In stages I and II, the memory was successfully retrieved because the inference path was close to the cluster in which the input data belonged; however, in stage III, the model could not determine the original cluster, so recall failed. Accordingly, the model achieves the inference dynamics depending on the input uncertainty. Figure 3.5b shows the time step of which the distance takes its minimum according to the noise fraction. The activity pattern approached the memory vector earliest in all stages. In addition, the time step of which the activity patterns are closest to the memory vector was dependent on the amount of noise. The dependence on the noise fraction of the minimum time step for the concept or the abstract concept vector was less significant than the one of the memory vector. These results suggest that the number of inference steps should be increased according to the amount of noise included in the input when performing noiseless reconstruction. On the other hand, the inference step should be around 20-30 independent from the noise fraction when performing label detection.

To confirm the abovementioned suggestion about the label prediction, we estimated the class label of the generated images at each step of inference. Figure 3.6 represents the label with the highest number of predictions in 200 trials. We used another convolutional neural network as a classification network in each trial. The classification network has the following structure: Input-Convolution-Convolution-Pooling-Dropout1-FullyConnected-Dropout2-SoftMax.

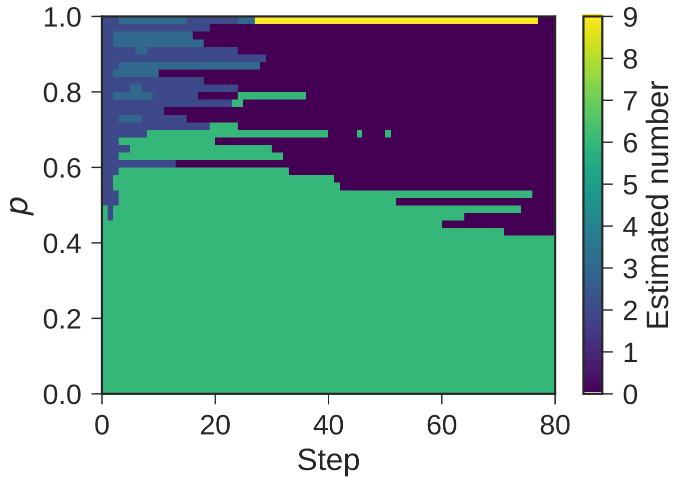


Figure 3.6: Estimated label of generated images. The class labels of the generated images at each step of inference were predicted using the classification network. The heatmap represents the label with the highest number of predictions in 200 trials. We used a different image of '6' with a different noise realization as the initial value for each trial.

The kernel size of the convolution is three, the pooling size is two, and the dropout probability is 0.25 and 0.5 in order from the input side. We used a rectified linear unit (ReLU) as the activation function. The classification network was trained on the original MNIST dataset before classifying the generated images of VAE. Based on the figure, the generated images started from '6' were classified correctly in every time step in stages I and II ($p < 0.4$). In stage III, the generated images were classified as '3' or '4' at the beginning of inference and were classified as '0' at the end of inference. There was a particular region of inference steps that the generated images were '6' for $p < 0.7$, and the region was close to 20-30. This result was consistent with the above mentioned suggestion.

As shown previously, the VAE extracts the cluster structures inherent in the MNIST data and infers images through the center of each cluster. These experimental results indicate that the dynamics of this inference are as shown in Figure 3.7. As shown in Figure 3.8a, when the dimensionality of the latent variable is large, only a few neurons contribute to representing the MNIST dataset. They would span a space that expresses each number (depicted as the numeric space in Figure 3.7). According to the manifold hypothesis (Rifai et al., 2011), adding noises to images will cause the initial value to deviate from the numeric space, which is believed to be low-dimensional. The results of our first analysis suggest that when the inference begins with a position far from the space expressing the MNIST data, the activity patterns first approach the memory vector and then quickly go to the corresponding concept vectors.

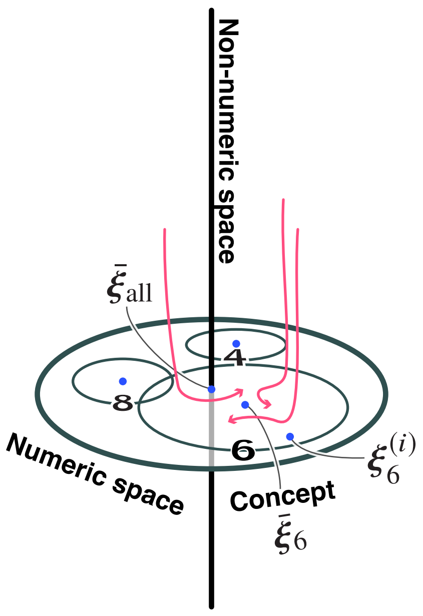


Figure 3.7: Schematic diagram of firing patterns in the latent state space.

3.3.3 *Effect of latent variable dimensionality*

Because the VAEs are the generative models that learn the mapping between the high-dimensional data space and the low-dimensional latent space, the dimensionality of the latent space is the essential hyperparameter for acquiring internal representation. The setting of the latent space dimensionality is predicted to drastically affect not only the quality of generated images and generalization ability but also the dynamics of repeated inferences. In this section, we numerically analyze the effect of the latent space dimensionality on the dynamics of repeated inferences.

We first show the relationships between the cluster centers of each label for the aforementioned setting (Figure 3.8). We set the dimension of the latent space to 100 in these experiments. The activity patterns in the latent variable space of each numerical concept are shown in Figure 3.8a. The heat map expresses the activity pattern of each neuron, which corresponds to the latent variable, where the x-axis represents the hidden neuron's index, and the y-axis represents the label. Only a few neurons out of 100 contribute to information representation, and many neurons are pruned and inactive. According to our observations, 14 out of 100 neurons were active. The dimensions of the latent space were examined using the cumulative contribution ratio determined by principal component analysis. The cumulative contribution ratios of each principal component when the training images were given

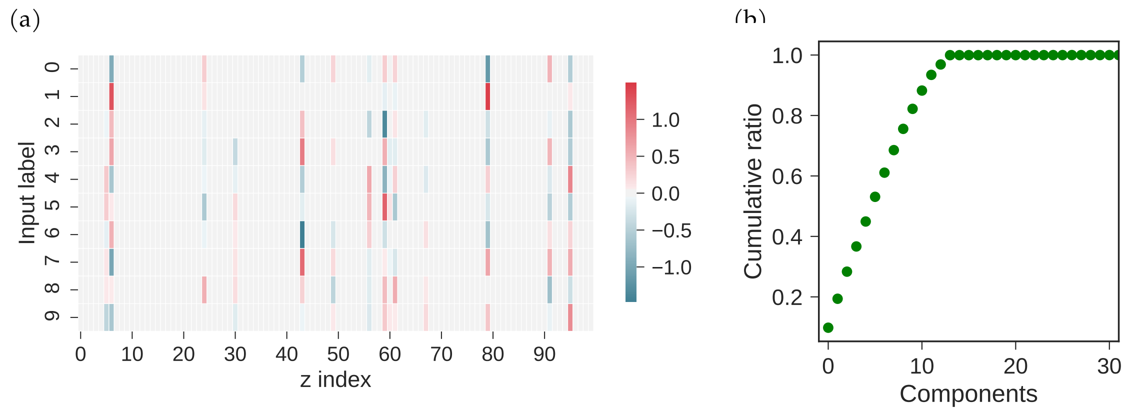


Figure 3.8: (a) Activity pattern in the latent variable space of each cluster center. The x-axis represents the neuron index of the latent variable, the y-axis represents the label, and the heat map shows the activity pattern of each neuron. (b) Cumulative contribution ratio of principal components.

to the VAE were shown in Figure 3.8b. The variance in the latent space was explained entirely by 14 dimensions, and 70% of this was explained by nine dimensions. Then, we quantified how much the concept vectors on latent space, which correspond to the row vector of Figure 3.8a, correlate with each other. We define the cosine similarity matrix C, where the element of the i-th row and j-th column is the cosine similarity between the concept vectors of labels i and j:

$$C_{ij} = \frac{\bar{\xi}_i \cdot \bar{\xi}_j}{\|\bar{\xi}_i\|_2 \|\bar{\xi}_j\|_2}. \tag{3.26}$$

We quantified the orthogonality between the concept vectors as $\|C - I\|_F^2$, where $\|A\|_F^2$ is the Frobenius norm of A: $\|A\|_F^2 = \sqrt{\sum_{ij} A_{ij}^2}$. By definition, the cosine similarity between concepts of the same label is one. However, the cosine similarity between concepts of different labels in nondiagonal terms is minimal, namely, near zero. In other words, as the vectors between the labels are orthogonal, the above quantity approaches zero. The left side of Figure 3.9 shows the aforementioned orthogonality according to the dimension of the latent space N_z. We trained VAEs from scratch for each N_z and calculated the orthogonality for learned representations. From the figure, the orthogonality of the internal representation increased with N_z, and the orthogonality converged to a value close to zero when N_z was approximately 10–20. We also visualized the typical dynamics of repeated inferences for $N_z = 2, 20, 100$ on the right side of the figure as (A), (B), and (C). We used the same setting as that of Figure 3.4 except for N_z, and the trained VAEs started repeated inferences from the images of '6' with input noise. The approach to the cluster center mentioned above appeared remarkably with the increase in the orthogonality of the internal representation. Because previous studies on the associative memory models (Amari, 1977; Matsumoto et al., 2005; Okada, 1996) also identified an approach to the concept vector during inferences under the assumption that the concept vectors were orthogonal, our findings were qualitatively consistent with these studies.

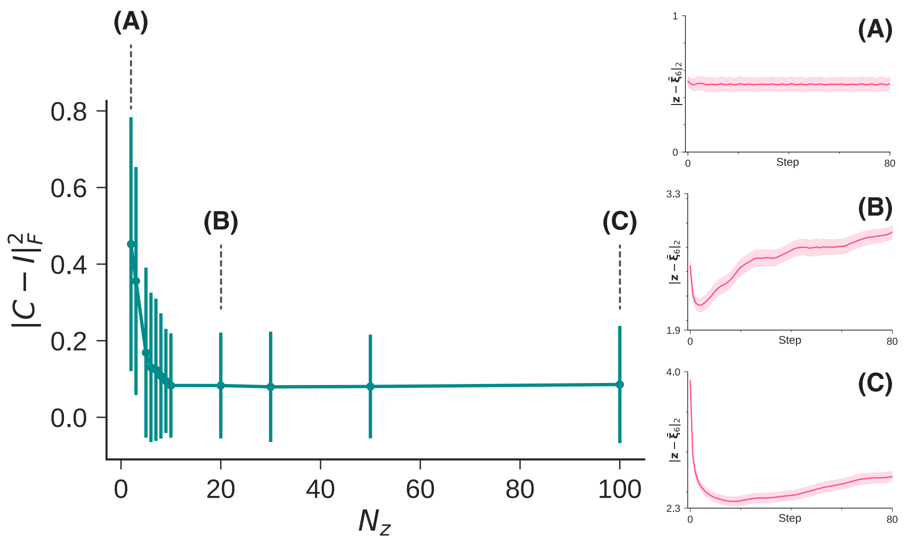


Figure 3.9: The orthogonality between the concept vectors in latent space. The error bars represent the ± 1 standard deviation.

The detailed change in the cosine similarity matrix C and its corresponding latent dynamics is in Figure 3.10. Figure 3.10a to Figure 3.10e visualize the cosine similarity matrices for corresponding latent space dimensionality N_z. The x- and the y-axes indicate the row and column of the matrix, and the figures visualize the value of each element as a heatmap. Because the cosine similarity matrix is symmetric, we omitted the upper half of the figures. The number of neurons in the latent variable N_z was controlled in the following order: 2, 5, 10, 20, and 100, and we trained the VAEs from scratch for each hyperparameter. By definition, the value of the diagonal elements is one. According to Figure 3.10a to Figure 3.10e, the values between structurally similar labels such as '0' and '6' or '7' and '9' were large. The cosine similarity between these values remained high even for the large N_z. We also compared the time evolution of the distance from a cluster center of '6' with different model hyperparameters in Figure 3.10f to Figure 3.10j. Under condition $N_z = 5$, the trajectory escaped the cluster center, and the trajectory also did not approach the cluster center.

These results and the previous findings imply that orthogonality is necessary between cluster centers for the trajectory of inference to be drawn into the cluster center. Because the

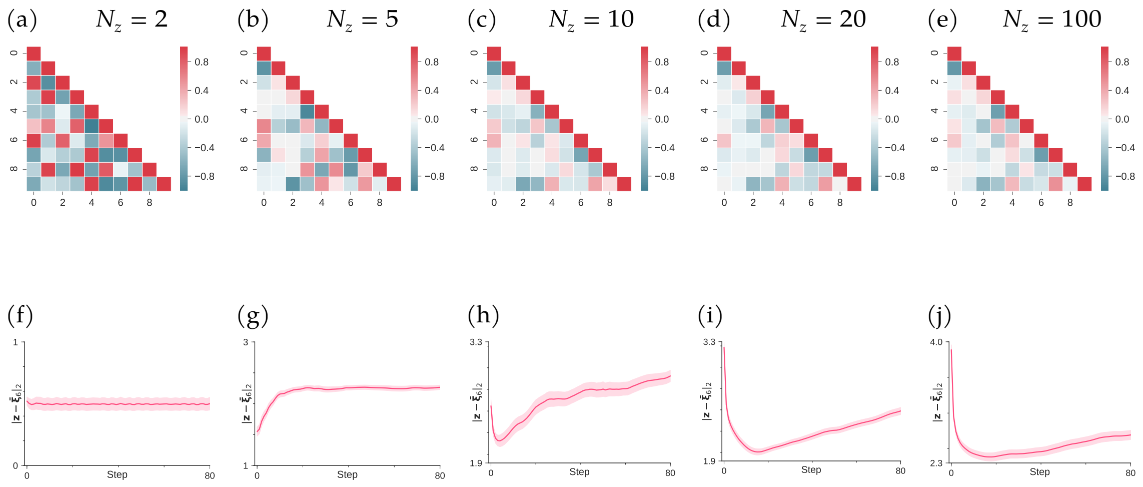


Figure 3.10: (a–e): Cosine similarity between the memory patterns of each concept. The number of elements in the latent variable is written as N_z. (f–j): Time development of the distance with a concept of '6' in (a–e).

number of latent variables decreases, it is necessary to express data in fewer dimensions, and the orthogonality is lost. The reduction in the number of latent variables is considered to cause unstable memory patterns corresponding to the training data, and only the center of clusters is stabilized. As a result, the trajectory of inference goes straight to a stable point. We also numerically assessed whether other labels confuse repeated inferences in the VAE (e.g., although an inference starts from label '6', it is incorrectly attracted to the concept associated with label '0'). The result of this assessment is shown in Section 3.C.

We also numerically analyzed the generalization performance according to N_z (Figure 3.11). The performance of the model was evaluated using the variational lower bound (Equation 3.1) of the log-likelihood for the test MNIST data. In each N_z, parameters that minimize the generalization error at epoch 100 with a total of nine conditions were selected from learning rates 0.01, 0.001, and 0.0001 and minibatch sizes of 50, 100, and 200. The generalization error was the minimum value in the vicinity of $N_z = 14$, and it did not change significantly afterward. In total, 14 of 100 latent variable neurons express training data under condition $N_z = 100$ (Figure 3.8a), and the number of neurons that minimize the generalization error is consistent with this result. These results suggest that approximately 14 latent neurons are required to express the MNIST data in the network structure used in this study. Moreover, in the vicinity of $N_z = 14$, the cluster structure appears in the representation of the latent variable space, and the trajectory of inference is drawn into the concept. These results suggest that it is possible to judge the generalization performance of the model without computing the generalization error or orthogonality of internal representations by simply observing the dynamics of repeated inference.

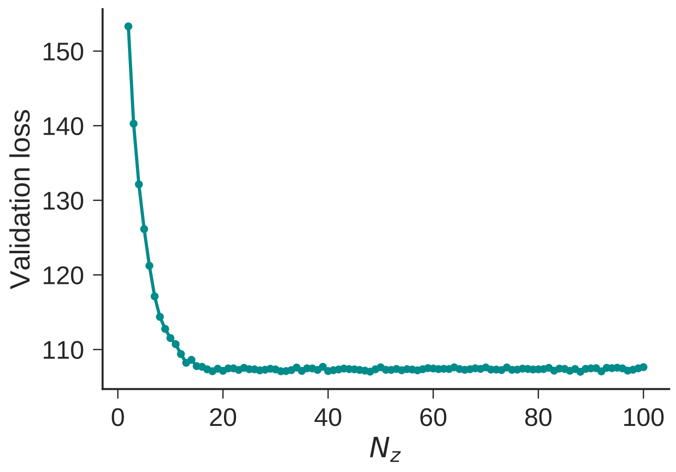


Figure 3.11: Generalization error for the number of elements of latent variables N_z. The y-axis represents the variational lower bound of the log-likelihood of the test data.

3.4 DISCUSSION

In this study, we numerically analyzed the dynamics of repeated inferences in VAEs for the datasets with a cluster structure. Based on the numerical analysis of the collective behaviors, the activity patterns in latent space rapidly approached a specific subspace. We also found that VAEs extract the cluster structures inherent in the MNIST and infer images via the center of each cluster. The results of the first analysis suggest that when the inference starts from a point far away from the original data distribution, the repeated inferences approach the concept vector at high speed. The approach of activity patterns to the area where the training dataset is concentrated is considered to be the cause of the improvement in the quality of the generated image by repeated inference, which was perceptually noted in the previous research.

The learning and inference of multiple memory patterns have been widely studied using associative memory models (Amit, Gutfreund, and Sompolinsky, 1985; Hopfield, 1982; Okada, 1996). In an associative memory model with multiple embedded, correlated patterns, the centroid of the correlated patterns spontaneously evolves to a fixed point (Amari, 1977), and the time evolution of the activity patterns approaches the concept (Matsumoto et al., 2005). The results of our first and second analyses are qualitatively consistent with these findings, suggesting that the mechanism underlying the dynamics of repeated inferences in the VAE is related to the traditional associative memory model.

Previously, several studies demonstrated that repeated inferences successfully denoise (Rezende, Mohamed, and Wierstra, 2014) and improve the quality of inferred images (Arulkumaran, Creswell, and Bharath, 2016). Our study suggests that the dynamics of repeated inferences approaching the center of the cluster inherent in the data lead to denoising and improving the quality of output images, which were quantitatively observed in the data space. It is critical to use a sufficient number of latent variables to precisely represent the concept inherent in the data; if the number of the latent variables is insufficient, the cluster structures will not be realized in the latent space, so the concept will be hardly identified. Our results suggest that stage II in Figure 3.5a appears only when the number of latent variables is sufficiently large, and the number of latent variables qualitatively changes the dynamics of repeated inferences.

We also studied the time profile of repeated inference. Our numerical experiments revealed that the latent activity pattern, which started from noisy input, approached the noiseless embedding (memory vector) earliest. In addition, the time step of this approaching was dependent on the amount of noise. These results gave us the practical implication about the optimal number of steps of VAE's repeated inference. The VAE can be used for several purposes, including noiseless reconstruction and embedding unknown data points for label detection. Our numerical experiments suggest that the number of inference steps should be increased according to the amount of noise when performing noiseless reconstruction. In addition, when performing label detection, the inference step should be larger than noiseless reconstruction.

In this study, we numerically analyzed the repeated inferences of VAEs for specific datasets. We mainly focused on the MNIST and the Fashion-MNIST, which have clear cluster structures. Hierarchical structures are one of the primary concerns of previous studies on the relationship between the structure of datasets and the behavior of deep neural networks. For example, deep neural networks are claimed to express abstract information in deep layers (Bengio et al., 2013; Lee et al., 2009). In particular, Bengio et al. stated that deep layers speed up the mixing of Markov chains using their ability to manifest abstract information. Moreover, Saxe et al. analytically showed that deep neural networks learn data in order from large to small modes, and the internal representations branch accordingly (Saxe, McClelland, and Ganguli, 2014). To clarify the universal behavior regarding the inference dynamics of deep generative models, we need to address the structure of various datasets that are not limited to the cluster structure, including the hierarchical structure.

Recently, researchers have been actively working on models that can capture features inherent in data as forms of internal representations (Grattarola, Livi, and Alippi, 2018; Higgins et al., 2017; Mathieu et al., 2019; Nagano et al., 2019; Nickel and Kiela, 2017; Ovinnikov, 2019; Tomczak and Welling, 2017). The VAE used in this study embeds the data points in a simple isomorphic Gaussian distribution. As a next step to expand on these works using other deep generative models, we aim to further investigate what factors influence the behavior of repeated inferences approaching the concept. In addition, we will analyze the dynamics of repeated inferences in another model using training datasets with more and varied hierarchies.

CHAPTER APPENDIX

3.A CONSECUTIVE SAMPLES FOR MNIST DATASET

We showed the consecutive samples for the initial image of '6' in Section 3.3.1. In this section, we additionally show the consecutive samples for other initial images in Figure 3.A.1. The visualizations and the experimental conditions all follow the one of Section 3.3.1. From the figures, the noise of the generated images gradually decreased with the inference step. In Figure 3.A.1, we note that we visualized the trials where the generated image at the last inference step clearly remained at the same label. The result of repeated inferences at the final step varies stochastically due to the effect of noise applied to the initial value. To clarify this effect, we also show the trials where the trained model failed to infer the appropriate images in Figure 3.A.2. The generated images tended to transition to perceptually similar images to the initial images when the inference fails: '2' to '8' or '5' to '3'. We also verified the effect of such a type of failure on the distance from the concept vector in a later section. Please see these sections for more detail

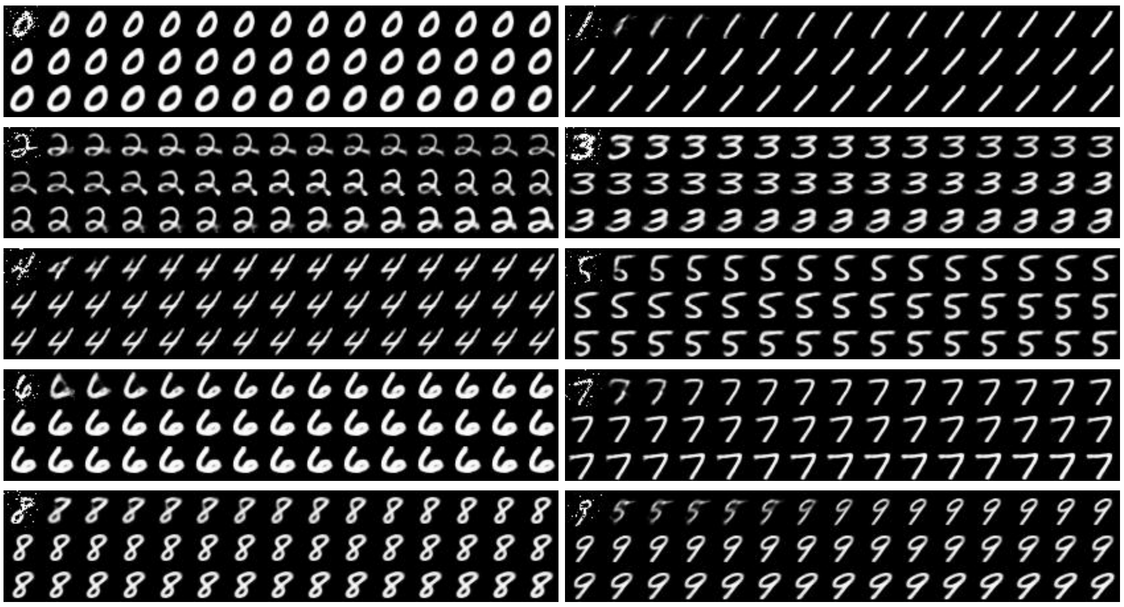


Figure 3.A.1: Consecutive samples for the MNIST dataset. All figures only show the trials where the generated image at the last inference step clearly remained at the same label.

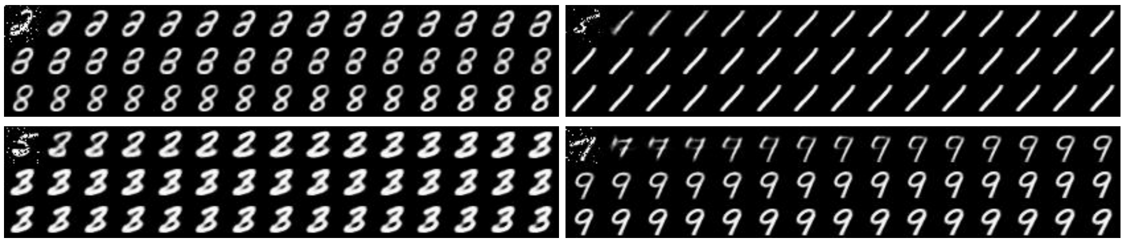


Figure 3.A.2: Consecutive samples for the MNIST dataset. All figures show the trials where the trained model failed to infer the appropriate images.

3.B NUMERICAL EVALUATIONS ON FASHION-MNIST DATASET

We numerically analyzed the collective behavior of latent activity patterns using the MNIST dataset in Section 3.3.1. This section shows the same numerical experiment on the Fashion-MNIST dataset, which is a dataset of Zalando's article images consisting of various fashion images. The Fashion-MNIST dataset was created to replace the original MNIST dataset for benchmarking machine learning algorithms. The size of images and the number of labels are exactly the same as the MNIST dataset. Based on this construction, the Fashion-MNIST dataset is also considered to have a cluster structure. We used the same network architecture and the hyperparameters as the experiments on the MNIST in the following.

Both Figure 3.B.1 and Figure 3.B.2 are the consecutive samples for the Fashion-MNIST dataset. Figure 3.B.1 shows the trials where the trained VAEs succeeded to infer the original label at the final step of repeated inferences, and Figure 3.B.2 shows the trials where the inference failed. The VAEs succeeded in removing the noise in the initial images as well as the case of the MNIST dataset. Notably, they reduced the noise drastically during the first several steps of repeated inferences. Fine structures such as the design of T-shirts were lost. Such behavior is considered to occur due to the limitation of multilayer perceptrons' ability to express and the noise applied to the initial image. In trials where inference has failed, the trained models inferred the images that only preserve the rough structure in the initial images. For example, the detailed structure of the handle of the bag was lost and became a jacket-like image. The lower half of the pants was integrated and changed to a dress-like image.

Then, we analyzed the distance between the latent activity patterns and the cluster centers. Figure 3.B.3 shows the time development for all labels. The meaning of the figure is as same as in Section 3.3.1. The distance between the cluster center and 300 different initial images was calculated. Each figure corresponds to each label that was used as initial input for the VAE. The x-axis expresses the time step t of repeated inference, and the y-axis expresses the Euclidean distance. All results were qualitatively consistent with the results of the MNIST dataset. The activity patterns in the latent space quickly approached the cluster centers and slowly left.

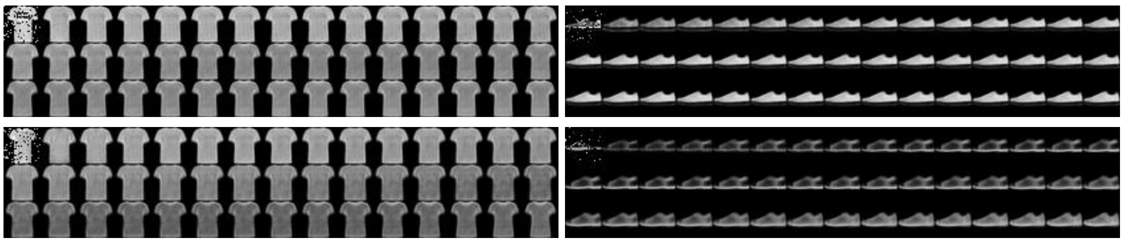


Figure 3.B.1: Consecutive samples for the Fashion-MNIST dataset. All figures only show the trials where the generated image at the last inference step clearly remained at the same label.

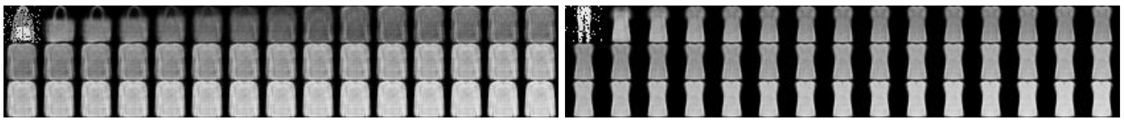


Figure 3.B.2: Consecutive samples for the Fashion-MNIST dataset. All figures show the trials where the trained model failed to infer the appropriate images.

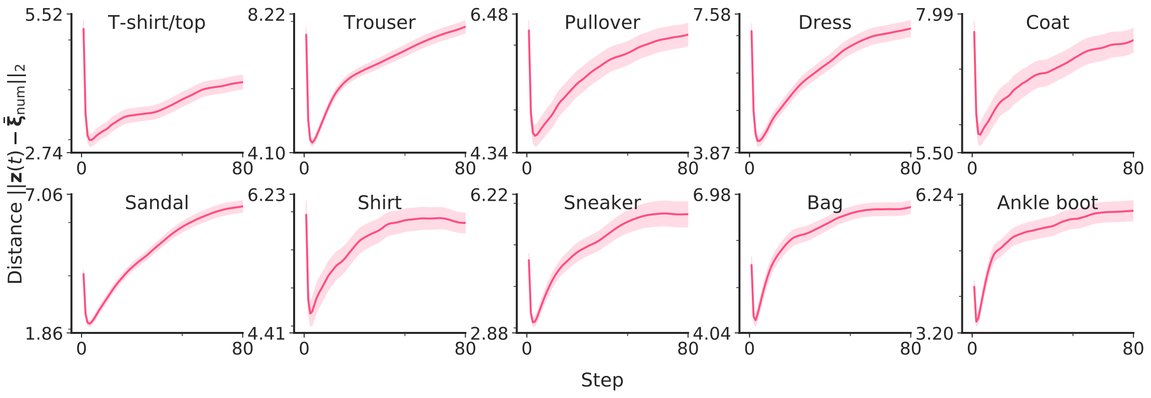


Figure 3.B.3: Time development of the distance from $\bar{\xi}_{\text{num}}$ for all labels of the Fashion-MNIST data. The shades represent the ± 1 standard error of the mean (300 trials). All figures were generated with the noise fraction $p = 0.2$.

3.C VERIFYING THE EFFECT OF MOVING OTHER NUMBERS

The possibility that other labels confused the repeated inferences in the VAE was numerically tested. It is conceivable that the escape from the cluster center is caused by attraction to another cluster. To eliminate this possibility, a discriminative neural network was constructed separately from the VAE, and the final state of inference of the VAE was classified.

In the following analysis, a model with an Input-Convolution-Convolution-Pooling-Dropout1-FullyConnected-Dropout2-SoftMax structure was constructed as the discriminative neural network. The kernel size of the convolution was set to three, the pooling size was set to two, and the dropout probability was set to 0.25, 0.5 in order from the input side. A rectified linear unit (ReLU) was used as the activation function. This model recorded a discrimination ability of 99.25% against the test data included in the MNIST dataset.

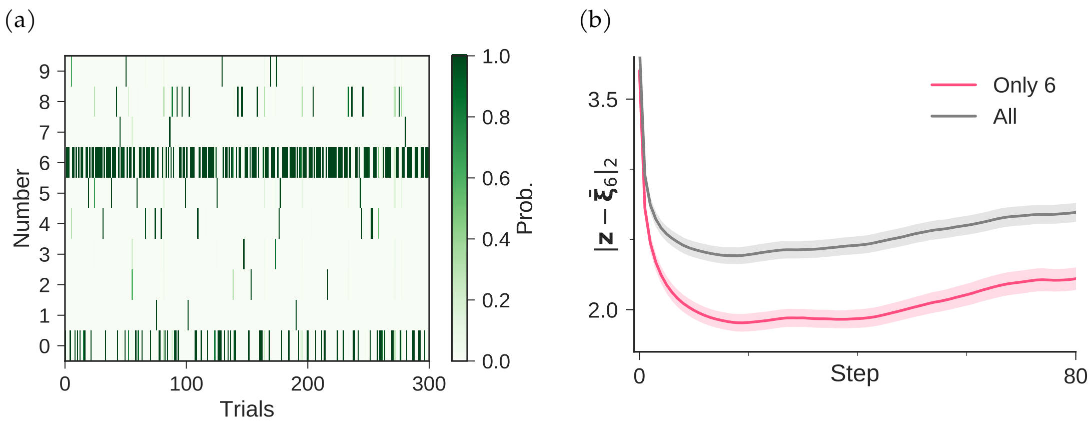


Figure 3.C.1: (a) The result of classifying the final state $T = 80$ of the inference for image '6'. (b) The time evolution of the distance from a concept of '6'. The condition excluding trials in which the activity pattern switched to different numbers is expressed in red, and the condition containing all the trials is expressed in gray.

The result of classifying the final state of inference using the aforementioned discriminative neural network is shown in Figure 3.C.1a. The x-axis represents a trial of each inference with various initial images, and the y-axis represents the number label. The heat map indicates the classification probability for each number label. An image of '6' was used as the initial value of the inference. The discriminator classified the final state of 193 of 300 trials as '6'.

We considered the effect of other labels as the cause of the neural activity patterns approaching mismatched clusters. Using the label '6', we first measured the distances between each neural activity pattern and the concept of '6'. We divided all the neural activity patterns into two conditions. We classified the trial in which the final state of the trajectory was inside the cluster of '6', as a condition "only 6", and all trials as a condition "all". Then, we averaged the distances in each condition and compared their means. The average trajectories are compared in Figure 3.C.1b. The red shows the average of the trial with the final state identified as '6', and gray shows the average of all trials.

As shown in Figure 3.C.1b, the neural activity patterns in both conditions approached the cluster center before moving to the corresponding patterns. This result suggests that the presence of other labels does not cause the neural activity patterns to move away from the cluster center.

4

PROBABILISTIC DISTRIBUTION ON HYPERBOLIC SPACE FOR HIERARCHICAL STRUCTURE

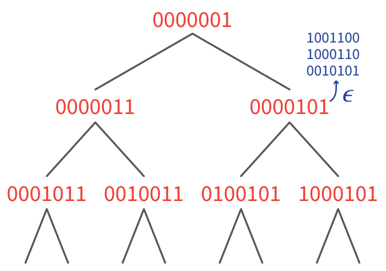
Hyperbolic space is a geometry that is known to be well-suited for representation learning of data with an underlying hierarchical structure. In this paper, we present a novel hyperbolic distribution called *hyperbolic wrapped distribution*, a wrapped normal distribution on hyperbolic space whose density can be evaluated analytically and differentiated with respect to the parameters. Our distribution enables the gradient-based learning of the probabilistic models on hyperbolic space that could never have been considered before. Also, we can sample from this hyperbolic probability distribution without resorting to auxiliary means like rejection sampling. As applications of our distribution, we develop a hyperbolic-analog of variational autoencoder and a method of probabilistic word embedding on hyperbolic space. We demonstrate the efficacy of our distribution on various datasets including MNIST, Atari 2600 Breakout, and WordNet.

4.1 INTRODUCTION

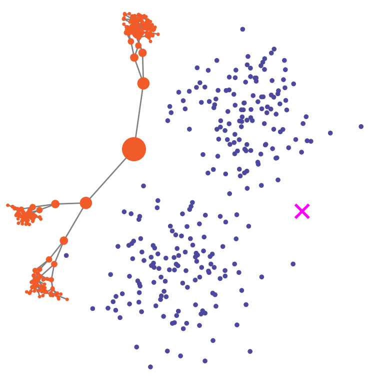
Recently, hyperbolic geometry is drawing attention as a powerful geometry to assist deep networks in capturing fundamental structural properties of data such as a hierarchy. Hyperbolic attention network (Gülçehre et al., 2019) improved the generalization performance of neural networks on various tasks including machine translation by imposing the hyperbolic geometry on several parts of neural networks. Poincaré embeddings (Nickel and Kiela, 2017) succeeded in learning a parsimonious representation of symbolic data by embedding the dataset into Poincaré balls.

In the task of data embedding, the choice of the target space determines the properties of the dataset that can be learned from the embedding. For the dataset with a hierarchical structure, in particular, the number of relevant features can grow exponentially with the depth of the hierarchy. Euclidean space is often inadequate for capturing the structural information (Figure 4.1.1). If the choice of the target space of the embedding is limited to Euclidean space, one might have to prepare extremely high dimensional space as the target

(a) A tree representation of the training dataset



(b) Vanilla VAE ($\beta = 1.0$)



(c) Hyperbolic VAE

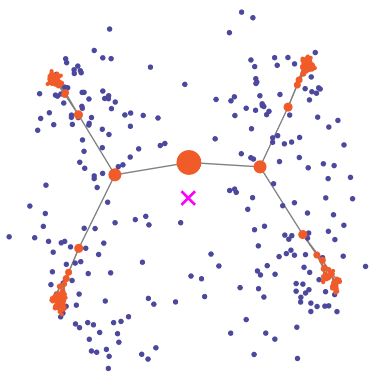


Figure 4.1.1: The visual results of Hyperbolic VAE applied to an artificial dataset generated by applying random perturbations to a binary tree. The visualization is being done on the Poincaré ball. The red points are the embeddings of the original tree, and the blue points are the embeddings of noisy observations generated from the tree. The pink × represents the origin of the hyperbolic space. The VAE was trained without the prior knowledge of the tree structure. Please see Section 4.6.1 for experimental details

space to guarantee small distortion. However, the same embedding can be done remarkably well if the destination is hyperbolic space (Sala et al., 2018; Sarkar, 2012).

Now, the next natural question is; "how can we extend these works to *probabilistic* inference problems on hyperbolic space?" When we know in advance that there is a hierarchical structure in the dataset, a prior distribution on hyperbolic space might serve as a good *informative* prior. We might also want to make Bayesian inference on a dataset with hierarchical structure by training a variational autoencoder (VAE) (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014) with latent variables defined on hyperbolic space. We might also want to conduct probabilistic word embedding into hyperbolic space while taking into account the uncertainty that arises from the underlying hierarchical relationship among words. Finally, it would be best if we can compare different probabilistic models on hyperbolic space based on popular statistical measures like divergence that requires the explicit form of the probability density function.

The endeavors we mentioned in the previous paragraph all require *probability distributions on hyperbolic space* that admit a parametrization of the density function that can be **computed analytically** and **differentiated** with respect to the parameter. Also, we want to be able to **sample from the distribution efficiently**; that is, we do not want to resort to auxiliary methods like rejection sampling.

In this study, we present a novel hyperbolic distribution called *hyperbolic wrapped distribution*, a wrapped normal distribution on hyperbolic space that resolves all these problems. We construct this distribution by defining Gaussian distribution on the tangent space at the origin of the hyperbolic space and projecting the distribution onto hyperbolic space after transporting the tangent space to a desired location in the space. This operation can be formalized by a combination of the parallel transport and the exponential map for the Lorentz model of hyperbolic space.

We can use our hyperbolic wrapped distribution to construct a probabilistic model on hyperbolic space that can be trained with gradient-based learning. For example, our distribution can be used as a prior of a VAE (Figure 4.1.1 and Figure 4.6.3). It is also possible to extend the existing probabilistic embedding method to hyperbolic space using our distribution, such as probabilistic word embedding. We will demonstrate the utility of our method through the experiments of probabilistic hyperbolic models on benchmark datasets including MNIST, Atari 2600 Breakout, and WordNet.

4.2 BACKGROUND

4.2.1 *Hyperbolic Geometry*

Hyperbolic geometry is a non-Euclidean geometry with a constant negative Gaussian curvature, and it can be visualized as the forward sheet of the two-sheeted hyperboloid. Many applications of hyperbolic space to machine learning to date have adopted the Poincaré

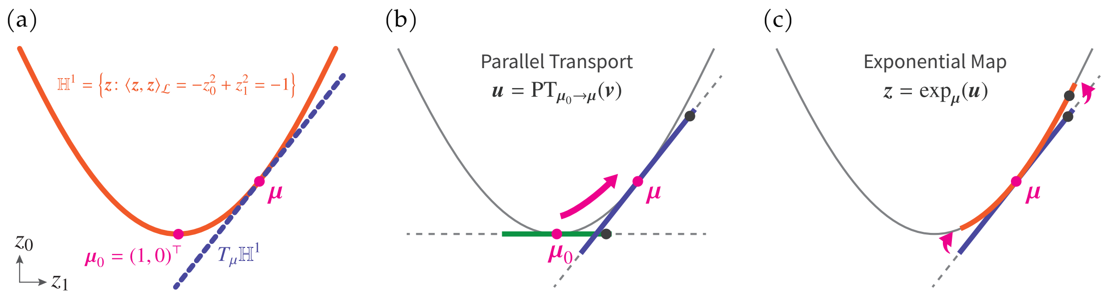


Figure 4.2.1: (a) One-dimensional Lorentz model $\mathbb{H}^1$ (red) and its tangent space $T_\mu\mathbb{H}^1$ (blue). (b) Parallel transport carries $v \in T_{\mu_0}$ (green) to $u \in T_\mu$ (blue) while preserving $\|\cdot\|_{\mathcal{L}}$. (c) Exponential map projects the $u \in T_\mu$ (blue) to $z \in \mathbb{H}^n$ (red). The distance between μ and $\exp_\mu(u)$ which is measured on the surface of $\mathbb{H}^n$ coincides with $\|u\|_{\mathcal{L}}$.

disk model as the subject of study (Ganea, Bécigneul, and Hofmann, 2018a,b; Nickel and Kiela, 2017; Sala et al., 2018). In this study, however, we will use the Lorentz model that, as claimed in Nickel and Kiela (2018), comes with a simpler closed form of the geodesics and does not suffer from the numerical instabilities in approximating the distance. We will also exploit the fact that both exponential map and parallel transportation have a clean closed form in the Lorentz model.

Lorentz model $\mathbb{H}^n$ (Figure 4.2.1a) can be represented as a set of points $z \in \mathbb{R}^{n+1}$ with $z_0 > 0$ such that its Lorentzian product (negative Minkowski bilinear form)

$$\langle z, z'\rangle_{\mathcal{L}} = -z_0 z_0' + \sum_{i=1}^{n} z_i z_i', \tag{4.1}$$

with itself is -1. That is,

$$\mathbb{H}^n = \left\{ z \in \mathbb{R}^{n+1} : \langle z, z\rangle_{\mathcal{L}} = -1,\ z_0 > 0 \right\}. \tag{4.2}$$

Lorentzian inner product also functions as the metric tensor on hyperbolic space. We will refer to the one-hot vector $\mu_0 = [1, 0, 0, ...0] \in \mathbb{H}^n \subset \mathbb{R}^{n+1}$ as the *origin* of the hyperbolic space. Also, the distance between two points z, z' on $\mathbb{H}^n$ is given by

$$d_\ell(z, z') = \operatorname{arccosh}\left(-\langle z, z'\rangle_{\mathcal{L}}\right), \tag{4.3}$$

which is also the length of the geodesic that connects z and z'.

4.2.2 *Parallel Transport and Exponential Map*

The rough explanation of our strategy for the construction of hyperbolic wrapped distribution $\mathcal{G}(\mu, \Sigma)$ with $\mu \in \mathbb{H}^n$ and a positive positive definite matrix Σ is as follows. We (1) sample a vector from $\mathcal{N}(0, \Sigma)$, (2) transport the vector from μ_0 to μ along the geodesic, and (3) project the vector onto the surface. To formalize this sequence of operations, we need to define the tangent space on hyperbolic space as well as the way to transport the tangent space and the way to project a vector in the tangent space to the surface. The transportation of the tangent vector requires *parallel transport*, and the projection of the tangent vector to the surface requires the definition of *exponential map.*

TANGENT SPACE OF HYPERBOLIC SPACE Let us use $T_\mu\mathbb{H}^n$ to denote the tangent space of $\mathbb{H}^n$ at μ (Figure 4.2.1a). Representing $T_\mu\mathbb{H}^n$ as a set of vectors in the same ambient space $\mathbb{R}^{n+1}$ into which $\mathbb{H}^n$ is embedded, $T_\mu\mathbb{H}^n$ can be characterized as the set of points satisfying the orthogonality relation with respect to the Lorentzian product:

$$T_\mu\mathbb{H}^n := \{u \colon \langle u, \mu\rangle_{\mathcal{L}} = 0\}. \tag{4.4}$$

$T_\mu\mathbb{H}^n$ set can be literally thought of as the tangent space of the forward hyperboloid sheet at μ. Note that $T_{\mu_0}\mathbb{H}^n$ consists of $v \in \mathbb{R}^{n+1}$ with $v_0 = 0$, and $\|v\|_{\mathcal{L}} := \sqrt{\langle v, v\rangle_{\mathcal{L}}} = \|v\|_2$.

PARALLEL TRANSPORT AND INVERSE PARALLEL TRANSPORT Next, for an arbitrary pair of point $\mu, \nu \in \mathbb{H}^n$, the parallel transport from ν to μ is defined as a map $\mathrm{PT}_{\nu\to\mu}$ from $T_\nu\mathbb{H}^n$ to $T_\mu\mathbb{H}^n$ that carries a vector in $T_\nu\mathbb{H}^n$ along the geodesic from ν to μ in a parallel manner without changing its metric tensor. In other words, if PT is the parallel transport on hyperbolic space, then $\langle \mathrm{PT}_{\nu\to\mu}(v), \mathrm{PT}_{\nu\to\mu}(v')\rangle_{\mathcal{L}} = \langle v, v'\rangle_{\mathcal{L}}$.

The parallel transportation on the Lorentz model along the geodesic from ν to μ is given by

$$\begin{aligned}\mathrm{PT}_{\nu\to\mu}(v) &= v - \frac{\langle \exp_\nu^{-1}(\mu), v\rangle_{\mathcal{L}}}{d_\ell(\nu,\mu)^2}\left(\exp_\nu^{-1}(\mu) + \exp_\mu^{-1}(\nu)\right)\\ &= v + \frac{\langle \mu - \alpha\nu, v\rangle_{\mathcal{L}}}{\alpha+1}(\nu+\mu),\end{aligned} \tag{4.5}$$

where $\alpha = -\langle \nu, \mu\rangle_{\mathcal{L}}$. We derived the above equations by using the inverse of the exponential map $\exp^{-1}(\cdot)$ described below. Next, likewise, for the exponential map, we need to be able to compute the inverse of the parallel transform. Solving Equation 4.5 for v, we get

$$v = u - \frac{\langle \mu - \alpha\nu, v\rangle_{\mathcal{L}}}{\alpha+1}(\nu+\mu). \tag{4.6}$$

Now, observing that

$$\begin{aligned}\langle \nu - \alpha\mu, u\rangle_{\mathcal{L}} =& \langle \nu, v\rangle_{\mathcal{L}} + \frac{\langle \mu - \alpha\nu, v\rangle_{\mathcal{L}}}{\alpha+1}(\langle \nu,\nu\rangle_{\mathcal{L}} + \langle \mu,\nu\rangle_{\mathcal{L}})\\ =& -\langle \mu, v\rangle_{\mathcal{L}} = -\langle \mu - \alpha\nu, v\rangle_{\mathcal{L}},\end{aligned} \tag{4.7}$$

we can write the inverse parallel transport as

$$v = \mathrm{PT}_{\nu \to \mu}^{-1}(\boldsymbol{u}) = \boldsymbol{u} + \frac{\langle \nu - \alpha\mu, \boldsymbol{u} \rangle_{\mathcal{L}}}{\alpha + 1}(\nu + \mu). \tag{4.8}$$

The inverse of parallel transport from ν to μ coincides with the parallel transport from μ to ν. The inverse parallel transport $\mathrm{PT}_{\nu \to \mu}^{-1}$ simply carries the vector in $T_\mu \mathbb{H}^n$ back to $T_\nu \mathbb{H}^n$ along the geodesic. That is,

$$v = \mathrm{PT}_{\nu \to \mu}^{-1}(\boldsymbol{u}) = \mathrm{PT}_{\mu \to \nu}(\boldsymbol{u}). \tag{4.9}$$

EXPONENTIAL MAP AND INVERSE EXPONENTIAL MAP Finally, we will describe a function that maps a vector in a tangent space to its surface.

According to the basic theory of differential geometry, every $\boldsymbol{u} \in T_\mu \mathbb{H}^n$ determines a unique maximal geodesic $\gamma_\mu : [0,1] \to \mathbb{H}^n$ with $\gamma_\mu(0) = \mu$ and $\dot{\gamma}_\mu(0) = \boldsymbol{u}$. Exponential map $\exp_\mu : T_\mu \mathbb{H}^n \to \mathbb{H}^n$ is a map defined by $\exp_\mu(\boldsymbol{u}) = \gamma_\mu(1)$, and we can use this map to project a vector v in $T_\mu \mathbb{H}^n$ onto $\mathbb{H}^n$ in a way that the distance from μ to destination of the map coincides with $\|v\|_{\mathcal{L}}$, the metric norm of v. For hyperbolic space, this map (Figure 4.2.1c) is given by

$$z = \exp_\mu(\boldsymbol{u}) = \cosh(\|\boldsymbol{u}\|_{\mathcal{L}})\mu + \sinh(\|\boldsymbol{u}\|_{\mathcal{L}})\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|_{\mathcal{L}}}. \tag{4.10}$$

As we can confirm with straightforward computation, this exponential map is norm preserving in the sense that $d_\ell(\mu, \exp_\mu(\boldsymbol{u})) = \operatorname{arccosh}\left(-\langle \mu, \exp_\mu(\boldsymbol{u}) \rangle_{\mathcal{L}}\right) = \|\boldsymbol{u}\|_{\mathcal{L}}$. Now, in order to evaluate the density of a point on hyperbolic space, we need to be able to map the point back to the tangent space, on which the distribution is initially defined. We, therefore, need to be able to compute the inverse of the exponential map, which is also called logarithm map, as well.

Solving Equation 4.10 for $\boldsymbol{u}$, we obtain

$$\boldsymbol{u} = \frac{\|\boldsymbol{u}\|_{\mathcal{L}}}{\sinh(\|\boldsymbol{u}\|_{\mathcal{L}})}(z - \cosh(\|\boldsymbol{u}\|_{\mathcal{L}})\mu). \tag{4.11}$$

We still need to obtain the evaluatable expression for $\|\boldsymbol{u}\|_{\mathcal{L}}$. Using the characterization of the tangent space (Equation 4.4), we see that

$$\begin{aligned} \langle \mu, \boldsymbol{u} \rangle_{\mathcal{L}} &= \frac{\|\boldsymbol{u}\|_{\mathcal{L}}}{\sinh(\|\boldsymbol{u}\|_{\mathcal{L}})}\left(\langle \mu, z \rangle_{\mathcal{L}} - \cosh(\|\boldsymbol{u}\|_{\mathcal{L}})\langle \mu, \mu \rangle_{\mathcal{L}}\right) = 0, \\ \iff \cosh(\|\boldsymbol{u}\|_{\mathcal{L}}) &= -\langle \mu, z \rangle_{\mathcal{L}}, \\ \iff \quad \|\boldsymbol{u}\|_{\mathcal{L}} &= \operatorname{arccosh}(-\langle \mu, z \rangle_{\mathcal{L}}). \end{aligned} \tag{4.12}$$

Now, defining $\alpha = -\langle \mu, z \rangle_{\mathcal{L}}$, we can obtain the inverse exponential function as

$$\boldsymbol{u} = \exp_\mu^{-1}(z) = \frac{\operatorname{arccosh}(\alpha)}{\sqrt{\alpha^2 - 1}}(z - \alpha\mu). \tag{4.13}$$

.

Algorithm 4.3.1 Sampling on hyperbolic space

```
1: **Input:** parameter $\boldsymbol{\mu} \in \mathbb{H}^n, \Sigma$
2: **Output:** $z \in \mathbb{H}^n$
3: **Require:** $\boldsymbol{\mu}_0 = (1, 0, \cdots, 0)^\top \in \mathbb{H}^n$
4: Sample $\tilde{v} \sim \mathcal{N}(0, \Sigma) \in \mathbb{R}^n$
5: $v = [0, \tilde{v}] \in T_{\boldsymbol{\mu}_0}\mathbb{H}^n$
6: Move $v$ to $\boldsymbol{u} = \mathrm{PT}_{\boldsymbol{\mu}_0 \to \boldsymbol{\mu}}(v) \in T_{\boldsymbol{\mu}}\mathbb{H}^n$ by Equation 4.5
7: Map $\boldsymbol{u}$ to $z = \exp_{\boldsymbol{\mu}}(\boldsymbol{u}) \in \mathbb{H}^n$ by Equation 4.10
```

4.3 HYPERBOLIC WRAPPED DISTRIBUTION

4.3.1 *Construction*

Finally, we are ready to provide the construction of our hyperbolic wrapped distribution $\mathcal{G}(\boldsymbol{\mu}, \Sigma)$ on hyperbolic space with $\boldsymbol{\mu} \in \mathbb{H}^n$ and positive definite Σ (Figure 4.3.1).

In the language of the differential geometry, our strategy can be re-described as follows:

1. Sample a vector $\tilde{v}$ from the Gaussian distribution $\mathcal{N}(0, \Sigma)$ defined over $\mathbb{R}^n$.
2. Interpret $\tilde{v}$ as an element of $T_{\boldsymbol{\mu}_0}\mathbb{H}^n \subset \mathbb{R}^{n+1}$ by rewriting $\tilde{v}$ as $v = [0, \tilde{v}]$.
3. Parallel transport the vector v to $\boldsymbol{u} \in T_{\boldsymbol{\mu}}\mathbb{H}^n \subset \mathbb{R}^{n+1}$ along the geodesic from $\boldsymbol{\mu}_0$ to $\boldsymbol{\mu}$.
4. Map $\boldsymbol{u}$ to $\mathbb{H}^n$ by $\exp_{\boldsymbol{\mu}}$.

Algorithm 4.3.1 is an algorithmic description of the sampling procedure based on our construction.

4.3.2 *Probability Density Function*

Note that both $\mathrm{PT}_{\boldsymbol{\mu}_0 \to \boldsymbol{\mu}}$ and $\exp_{\boldsymbol{\mu}}$ are differentiable functions that can be evaluated analytically. Thus, by the construction of $\mathcal{G}(\boldsymbol{\mu}, \Sigma)$, we can compute the probability density of $\mathcal{G}(\boldsymbol{\mu}, \Sigma)$ at $z \in \mathbb{H}^n$ using a composition of differentiable functions, $\mathrm{PT}_{\boldsymbol{\mu}_0 \to \boldsymbol{\mu}}$ and $\exp_{\boldsymbol{\mu}}$. Let $\mathrm{proj}_{\boldsymbol{\mu}} := \exp_{\boldsymbol{\mu}} \circ \mathrm{PT}_{\boldsymbol{\mu}_0 \to \boldsymbol{\mu}}$.

In general, if $\boldsymbol{X}$ is a random variable endowed with the probability density function $p(\boldsymbol{x})$, the log likelihood of $Y = f(\boldsymbol{X})$ at $\boldsymbol{y}$ can be expressed as

$$\log p(\boldsymbol{y}) = \log p(\boldsymbol{x}) - \log \det \left(\frac{\partial f}{\partial \boldsymbol{x}} \right) \tag{4.14}$$

Algorithm 4.3.2 Calculate log-pdf

```
1: Input: sample $z \in \mathbb{H}^n$, parameter $\mu \in \mathbb{H}^n$, $\Sigma$
2: Output: $\log p(z)$
3: Require: $\mu_0 = (1, 0, \cdots, 0)^\top \in \mathbb{H}^n$
4: Map $z$ to $u = \exp_\mu^{-1}(z) \in T_\mu \mathbb{H}^n$ by Equation 4.13
5: Move $u$ to $v = \mathrm{PT}_{\mu_0 \to \mu}^{-1}(u) \in T_{\mu_0} \mathbb{H}^n$ by Equation 4.8
6: Calculate $\log p(z)$ by Equation 4.15
```

where f is a invertible and continuous map. Thus, all we need in order to evaluate the probability density of $\mathcal{G}(\mu, \Sigma)$ at $z = \mathrm{proj}_\mu(v)$ is the way to evaluate $\det\left(\partial \, \mathrm{proj}_\mu(v) / \partial v\right)$:

$$\log p(z) = \log p(v) - \log \det \left(\frac{\partial \, \mathrm{proj}_\mu(v)}{\partial v} \right). \tag{4.15}$$

Algorithm 4.3.2 is an algorithmic description for the computation of the pdf.

For the implementation of Algorithm 4.3.1 and Algorithm 4.3.2, we would need to be able to evaluate not only $\exp_\mu(u)$, $\mathrm{PT}_{\mu_0 \to \mu}(v)$ and their inverses, but also need to evaluate the determinant. We provide an analytic solution to each one of them below.

LOG-DETERMINANT For the evaluation of Equation 4.15, we need to compute the log determinant of the projection function that maps a vector in the tangent space $T_{\mu_0}(\mathbb{H}^n)$ at origin to the tangent space $T_\mu(\mathbb{H}^n)$ at an arbitrary point μ in the hyperbolic space.

Appealing to the chain-rule and the property of determinant, we can decompose the expression into two components:

$$\det\left(\frac{\partial \, \mathrm{proj}_\mu(v)}{\partial v} \right) = \det \left(\frac{\partial \exp_\mu(u)}{\partial u} \right) \cdot \det \left(\frac{\partial \, \mathrm{PT}_{\mu_0 \to \mu}(v)}{\partial v} \right). \tag{4.16}$$

We evaluate each piece one by one. First, let us recall that $\partial \exp_\mu(u) / \partial u$ is a map that sends an element in $T_u(T_\mu(\mathbb{H}^n)) = T_\mu(\mathbb{H}^n)$ to an element in $T_v(\mathbb{H}^n)$, where $v = \exp_\mu(u)$. We have a freedom in choosing a basis to evaluate the determinant of this expression. For convenience, let us choose an orthonormal basis of $T_\mu(\mathbb{H}^n)$ that contains $\bar{u} = u / \|u\|_\mathcal{L}$:

$$\{\bar{u}, u'_1, u'_2, \ldots . u'_{n-1}\}. \tag{4.17}$$

The desired determinant can be computed by tracking how much each element of this basis grows in magnitude under the transformation.

The derivative in the direction of each basis element can be computed as follows:

$$\begin{aligned}
\mathrm{d}\exp_\mu(\bar{\boldsymbol{u}}) &= \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \exp_\mu(\boldsymbol{u}+\epsilon\bar{\boldsymbol{u}}) \\
&= \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \left[\cosh(r+\epsilon)\boldsymbol{\mu} + \sinh(r+\epsilon)\frac{\boldsymbol{u}+\epsilon\bar{\boldsymbol{u}}}{\|\boldsymbol{u}+\epsilon\bar{\boldsymbol{u}}\|_{\mathcal{L}}}\right] \\
&= \sinh(r)\boldsymbol{\mu} + \cosh(r)\bar{\boldsymbol{u}}, \qquad (4.18)
\end{aligned}$$

$$\begin{aligned}
\mathrm{d}\exp_\mu(\boldsymbol{u}'_k) &= \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \exp_\mu(\boldsymbol{u}+\epsilon\boldsymbol{u}'_k) \\
&= \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \left[\cosh(r)\boldsymbol{\mu} + \sinh(r)\frac{\boldsymbol{u}+\epsilon\boldsymbol{u}'}{r}\right] \\
&= \frac{\sinh r}{r}\boldsymbol{u}'. \qquad (4.19)
\end{aligned}$$

In the second line of the computation of the directional derivative with respect to $\boldsymbol{u}'_k$, we used the fact that $\|\boldsymbol{u}+\epsilon\boldsymbol{u}'_k\|_{\mathcal{L}} = \sqrt{\langle\boldsymbol{u},\boldsymbol{u}\rangle_{\mathcal{L}} + \epsilon\langle\boldsymbol{u},\boldsymbol{u}'_k\rangle_{\mathcal{L}} + \epsilon^2\langle\boldsymbol{u}'_k,\boldsymbol{u}'_k\rangle_{\mathcal{L}}} = \|\boldsymbol{u}\|_{\mathcal{L}} + O(\epsilon^2)$ and that $O(\epsilon^2)$ in the above expression will disappear in the $\epsilon \to 0$ limit of the finite difference. All together, the derivatives computed with respect to our choice of the basis elements are given by

$$\left(\sinh(r)\boldsymbol{\mu} + \cosh(r)\bar{\boldsymbol{u}}, \frac{\sinh r}{r}\boldsymbol{u}'_1, \frac{\sinh r}{r}\boldsymbol{u}'_2, \cdots, \frac{\sinh r}{r}\boldsymbol{u}'_{n-1}\right). \qquad (4.20)$$

The desired determinant is the product of the Lorentzian norms of the vectors of the set above. Because all elements of $T_\mu(\mathcal{H}^n) \subset \mathbb{R}^n$ are orthogonal with respect to the Lorentzian inner product and because $\|\sinh(r)\boldsymbol{\mu} + \cosh(r)\bar{\boldsymbol{u}}\|_{\mathcal{L}} = 1$ and $\|\sinh(r)/r \cdot \boldsymbol{u}'\|_{\mathcal{L}} = \sinh(r)/r$, we get

$$\det\left(\frac{\partial\exp_\mu(\boldsymbol{u})}{\partial\boldsymbol{u}}\right) = \left(\frac{\sinh r}{r}\right)^{n-1}. \qquad (4.21)$$

Next, let us compute the determinant of the parallel transport. Let $\boldsymbol{v} \in T_{\mu_0}\mathbb{H}^n$, and let $\boldsymbol{u} = \mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{v}) \in T_\mu\mathbb{H}^n$. The derivative of this map is a map from $T_v(T_{\mu_0}(\mathbb{H}^n))$ to $T_u(\mathbb{H}^n)$. Let us choose an orthonormal basis ξ_k (In Lorentzian sense). Likewise above, we can compute the desired determinant by tracking how much each element of this basis grows in magnitude under the transformation. Denoting $\alpha = -\langle\boldsymbol{\mu}_0,\boldsymbol{\mu}\rangle_{\mathcal{L}}$, we get

$$\begin{aligned}
\mathrm{d}\,\mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{\xi}) &= \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{v}+\epsilon\boldsymbol{\xi}) \\
&= \frac{\partial}{\partial\epsilon}\bigg|_{\epsilon=0} \left[(\boldsymbol{v}+\epsilon\boldsymbol{\xi}) + \frac{\langle\boldsymbol{\mu}-\alpha\boldsymbol{\mu}_0, \boldsymbol{v}+\epsilon\boldsymbol{\xi}\rangle_{\mathcal{L}}}{\alpha+1}(\boldsymbol{\mu}_0+\boldsymbol{\mu})\right] \\
&= \boldsymbol{\xi} + \frac{\langle\boldsymbol{\mu}-\alpha\boldsymbol{\mu}_0, \boldsymbol{\xi}\rangle_{\mathcal{L}}}{\alpha+1}(\boldsymbol{\mu}_0+\boldsymbol{\mu}) = \mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{\xi}), \qquad (4.22)
\end{aligned}$$

and see that each basis element $\boldsymbol{\xi}_k$ is mapped by $\mathrm{d}\,\mathrm{PT}_{\mu_0\to\mu}$ to

$$\left(\mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{\xi}_1), \mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{\xi}_2)\ \cdots,\ \mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{\xi}_n)\right). \tag{4.23}$$

Because parallel transport is a norm preserving map, $\left\|\mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{\xi})\right\|_{\mathcal{L}} = 1$. That is,

$$\det\left(\frac{\partial\,\mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{v})}{\partial \boldsymbol{v}}\right) = 1. \tag{4.24}$$

All together, we get

$$\det\left(\frac{\partial\,\mathrm{proj}_{\mu}(\boldsymbol{v})}{\partial \boldsymbol{v}}\right) = \left(\frac{\sinh r}{r}\right)^{n-1}. \tag{4.25}$$

The whole evaluation of the log determinant can be computed in $\mathcal{O}(n)$. Figure 4.3.1 shows example densities on $\mathbb{H}^2$.

Since the metric at the tangent space coincides with the Euclidean metric, we can produce various types of distributions on hyperbolic space by applying our construction strategy to other distributions defined on Euclidean space, such as Laplace and Cauchy distribution.

4.4 APPLICATIONS OF HYPERBOLIC WRAPPED DISTRIBUTION

4.4.1 *Hyperbolic Variational Autoencoder*

As an application of hyperbolic wrapped distribution $\mathcal{G}(\boldsymbol{\mu}, \Sigma)$, we will introduce *hyperbolic variational autoencoder* (Hyperbolic VAE), a variant of the variational autoencoder (VAE) (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014) in which the latent variables are defined on hyperbolic space. Given dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$, the method of variational autoencoder aims to train a decoder model $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ that can create from $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ a dataset that resembles $\mathcal{D}$. The decoder model is trained together with the encoder model $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ by maximizing the sum of evidence lower bound (ELBO) that is defined for each $\boldsymbol{x}^{(i)}$;

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z})\right] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})\|p_{\boldsymbol{\theta}}(\boldsymbol{z})), \tag{4.26}$$

where $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})$ is the variational posterior distribution. In classic VAE, the choice of the prior $p_{\boldsymbol{\theta}}$ is the standard normal, and the posterior distribution is also variationally approximated by a Gaussian. Hyperbolic VAE is a simple modification of the classic VAE in which $p_{\boldsymbol{\theta}} = \mathcal{G}(\boldsymbol{\mu}_0, I)$ and $q_{\boldsymbol{\phi}} = \mathcal{G}(\boldsymbol{\mu}, \Sigma)$. The model of $\boldsymbol{\mu}$ and Σ is often referred to as encoder. This parametric formulation of $q_{\boldsymbol{\phi}}$ is called reparametrization trick, and it enables the evaluation of the gradient of the objective function with respect to the network parameters. To compare our method against, we used β-VAE (Higgins et al., 2017), a variant of VAE that applies a scalar weight β to the KL term in the objective function.

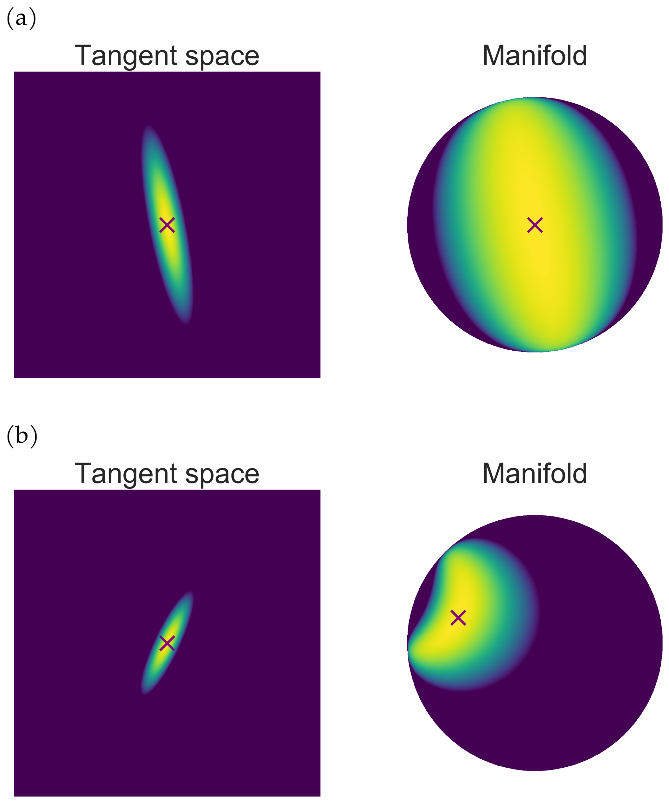


Figure 4.3.1: The heatmaps of log-likelihood of the hyperbolic wrapped distribution with various $\boldsymbol{\mu}$ and Σ. We designate the origin of hyperbolic space by the $\times$ mark. See Section 4.A for further details.

In Hyperbolic VAE, we assure that output $\boldsymbol{\mu}$ of the encoder is in $\mathbb{H}^n$ by applying $\exp_{\boldsymbol{\mu}_0}$ to the final layer of the encoder. That is, if $\boldsymbol{h}$ is the output, we can simply use

$$\boldsymbol{\mu} = \exp_{\boldsymbol{\mu}_0}(\boldsymbol{h}) = \left(\cosh(\|\boldsymbol{h}\|_2), \quad \sinh(\|\boldsymbol{h}\|_2)\frac{\boldsymbol{h}}{\|\boldsymbol{h}\|_2}\right)^\top. \tag{4.27}$$

As stated in the previous sections, our distribution $\mathcal{G}(\boldsymbol{\mu}, \Sigma)$ allows us to evaluate the ELBO exactly and to take the gradient of the objective function. In a way, our distribution of the variational posterior is an hyperbolic-analog of the reparametrization trick.

4.4.2 *Word Embedding*

We can use our hyperbolic wrapped distribution $\mathcal{G}$ for probabilistic word embedding. The work of Vilnis and McCallum (2015) attempted to extract the linguistic and contextual properties of words in a dictionary by embedding every word and every context to a Gaussian distribution defined on Euclidean space. We may extend their work by changing the destination of the map to the family of $\mathcal{G}$. Let us write $a \sim b$ to convey that there is a link between words a and b, and let us use q_s to designate the distribution to be assigned to the word s. The objective function used in Vilnis and McCallum (2015) is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(s \sim t, s \nsim t')}[\max(0, m + E(s,t) - E(s,t'))], \tag{4.28}$$

where $E(s,t)$ represents the measure of similarity between s and t evaluated with $D_{\mathrm{KL}}(q_s \| q_t)$. In the original work, q_s and q_t were chosen to be a Gaussian distribution. We can incorporate hyperbolic geometry into this idea by choosing $q_s = \mathcal{G}(\boldsymbol{\mu}(s), \Sigma(s))$.

4.5 RELATED WORKS

The construction of hyperbolic distribution based on projection is not entirely new on its own. For example, CCM-AAE (Grattarola, Livi, and Alippi, 2018) uses a prior distribution on hyperbolic space centered at origin by projecting a distribution constructed on the tangent space. Wrapped normal distribution (on sphere) also is a creation of similar philosophy. Still yet, as mentioned in the introduction, most studies to date that use hyperbolic space consider only deterministic mappings (Ganea, Bécigneul, and Hofmann, 2018a,b; Gülçehre et al., 2019; Nickel and Kiela, 2017, 2018).

Ovinnikov (2019) and Mathieu et al. (2019) proposed an extension of Gaussian distribution on Poincaré ball model and its application to VAEs. Ovinnikov (2019) used Wasserstein Maximum Mean Discrepancy (Gretton et al., 2012), and Mathieu et al. (2019) used Monte Carlo approximation of ELBO for training their models. By construction, however, their method can only create isotropic distribution on Riemannian manifold and they also have to use rejection sampling in their method. Meanwhile, our method can wrap $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ onto $\mathbb{H}^n$ for arbitrary choice of Σ, and we do not have to use rejection sampling. Our distribution $\mathcal{G}(\boldsymbol{\mu}, \Sigma)$ can be defined for any $\boldsymbol{\mu}$ in $\mathbb{H}^n$ and any positive definite matrix $\Sigma \in \mathbb{R}_{+}^{n \times n}$.

For word embedding, several deterministic methods have been proposed to date, including the celebrated Word2Vec (Mikolov et al., 2013). The aforementioned Nickel and Kiela (2017) uses deterministic hyperbolic embedding to exploit the hierarchical relationships among words. The probabilistic word embedding was first proposed by Vilnis and McCallum (2015). As stated in the method section, their method maps each word to a Gaussian distribution on Euclidean space. Their work suggests the importance of investigating the uncertainty of word embedding. In the field of representation learning of word vectors, our work is the first in using hyperbolic probability distribution for word embedding.

On the other hand, the idea to use a noninformative, non-Gaussian prior in VAE is not new. For example, Davidson et al. (2018) proposes the use of von Mises-Fisher prior, and Jang, Gu, and Poole (2017) and Rolfe (2017) use discrete distributions as their prior. With the method of Normalizing flow (Rezende and Mohamed, 2015), one can construct even more complex priors as well (Kingma et al., 2016). The appropriate choice of the prior shall

Model		Correlation	Correlation w/ noise
Vanilla	$\beta = 0.1$	$0.665_{\pm.006}$	$0.470_{\pm.018}$
	$\beta = 1.0$	$0.644_{\pm.007}$	$0.550_{\pm.012}$
	$\beta = 2.0$	$0.644_{\pm.011}$	$0.537_{\pm.012}$
	$\beta = 3.0$	$0.638_{\pm.004}$	$0.501_{\pm.044}$
	$\beta = 4.0$	$0.217_{\pm.143}$	$0.002_{\pm.042}$
Hyperbolic		**0.768** $\pm.003$	**0.590** $\pm.018$

Table 4.6.1: Results of tree embedding experiments for the Hyperbolic VAE and Vanilla VAEs trained with different weight constants for the KL term. We calculated the mean and the ± 1 SD with five different experiments.

depend on the type of dataset. As we will show in the experiment section, our distribution is well suited to the dataset with underlying tree structures. Another choice of the VAE prior that specializes in such dataset has been proposed by Vikram, Hoffman, and Johnson (2018). For the sampling, they use time-marginalized coalescent, a model that samples a random tree structure by a stochastic process. Theoretically, their method can be used in combination with our approach by replacing their Gaussian random walk with a hyperbolic random walk.

4.6 experiments

4.6.1 *Synthetic Binary Tree*

We trained Hyperbolic VAE for an artificial dataset constructed from a binary tree of depth $d = 8$. To construct the dataset, we first obtained a binary representation for each node in the tree so that the Hamming distance between any pair of nodes is the same as the distance on the graph representation of the tree (Figure 4.1.1a). Let us call the set of binaries obtained this way by A_0. We then generated a set of binaries, A, by randomly flipping each coordinate value of A_0 with probability $\epsilon = 0.1$. The binary set A was then embedded into $\mathbb{R}^d$ by mapping $a_1 a_2 ... a_d$ to $[a_1, a_2, ..., a_d]$. We used an Multi Layer Parceptron (MLP) of depth 3 and 100 hidden variables at each layer for both encoder and decoder. For activation function we used *tanh*.

Table 4.6.1 summarizes the quantitative comparison of Vanilla VAE against our Hyperbolic VAE. For each pair of points in the tree, we computed their Hamming distance as well as their distance in the latent space of VAE. That is, we used hyperbolic distance for Hyperbolic VAE, and used Euclidean distance for Vanilla VAE. We used the strength of correlation between the Hamming distances and the distances in the latent space as a measure of performance. Hyperbolic VAE was performing better both on the original tree and on the artificial dataset generated from the tree. Vanilla VAE performed the best with $\beta = 2.0$, and collapsed with $\beta = 3.0$. The difference between Vanilla VAE and Hyperbolic VAE can be observed with much more clarity using the 2-dimensional visualization of the generated dataset on Poincaré Ball (See Figure 4.1.1 and Section 4.B.1). The red points are the embeddings of A_0, and the blue points are the embeddings of all other points in A. The pink $\times$ mark designates the origin of hyperbolic space. For the visualization, we used the canonical diffeomorphism between the Lorenz model and the Poincaré ball model.

4.6.2 *MNIST*

	Vannila VAE		Hyperbolic VAE	
n	ELBO	LL	ELBO	LL
2	-145.53 ±.65	-140.45 ±.47	-143.23 ±0.63	**-138.61** ±0.45
5	-111.32 ±.38	-105.78 ±.51	-111.09 ±0.39	**-105.38** ±0.61
10	-92.49 ±.52	**-86.25** ±.52	-93.10 ±0.26	-86.40 ±0.28
20	-85.17 ±.40	**-77.89** ±.36	-88.28 ±0.34	-79.23 ±0.20

Table 4.6.2: Quantitative comparison of Hyperbolic VAE against Vanilla VAE on the MNIST dataset in terms of ELBO and log-likelihood (LL) for several values of latent space dimension n. LL was computed using 500 samples of latent variables. We calculated the mean and the ± 1 SD with five different experiments.

We applied Hyperbolic VAE to a binarized version of MNIST. We used an MLP of depth 3 and 500 hidden units at each layer for both the encoder and the decoder. Table 4.6.2 shows the quantitative results of the experiments. Log-likelihood was approximated with an empirical integration of the Bayesian predictor with respect to the latent variables (Burda, Grosse, and Salakhutdinov, 2016). Our method outperformed Vanilla VAE with small latent dimension. Figure 4.6.1a are the samples of the Hyperbolic VAE that was trained with 5-dimensional latent variables, and Figure 4.6.1b are the Poincaré Ball representations of the interpolations produced on $\mathbb{H}^2$ by the Hyperbolic VAE that was trained with 2-dimensional latent variables.

4.6.3 *Atari 2600 Breakout*

In reinforcement learning, the number of possible state-action trajectories grows exponentially with the time horizon. We may say that these trajectories often have a tree-like hierar-

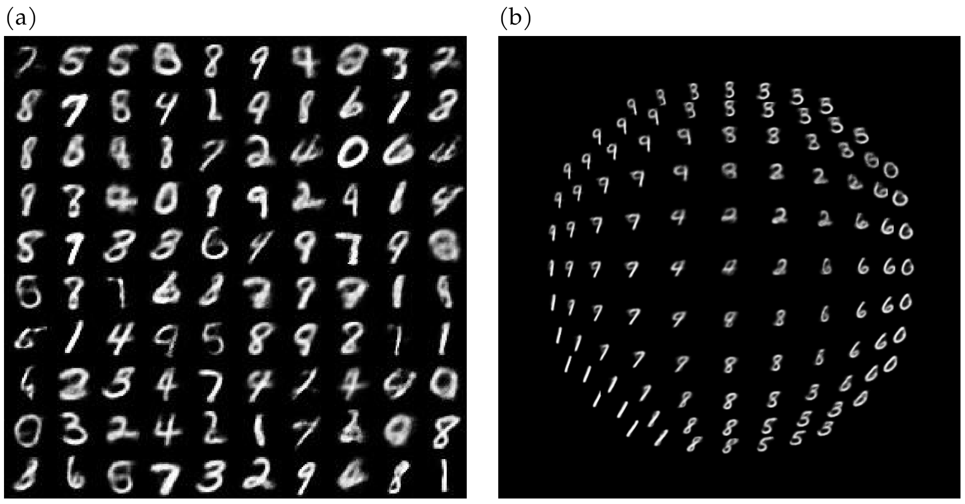


Figure 4.6.1: (a) Samples generated from the Hyperbolic VAE trained on MNIST with latent dimension $n = 5$. (b): Interpolation of the MNIST dataset produced by the Hyperbolic VAE with latent dimension $n = 2$, represented on the Poincaré ball.

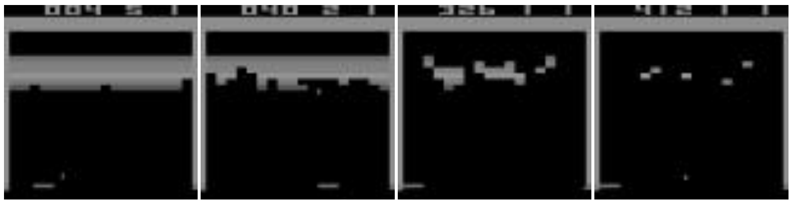


Figure 4.6.2: Examples of the observed screens in Atari 2600 Breakout.

chical structure that starts from the initial states. We applied our Hyperbolic VAE to a set of trajectories that were explored by a trained policy during multiple episodes of Breakout in Atari 2600. To collect the trajectories, we used a pretrained Deep Q-Network (Mnih et al., 2015), and used epsilon-greegy with $\epsilon = 0.1$. We amassed a set of trajectories whose total length is 100,000, of which we used 80,000 as the training set, 10,000 as the validation set, and 10,000 as the test set. Each frame in the dataset was gray-scaled and resized to 80 × 80. The images in the Figure 4.6.2 are samples from the dataset. We used a DCGAN-based architecture (Radford, Metz, and Chintala, 2015) with latent space dimension $n = 20$. Please see Section 4.C for more details.

The Figure 4.6.3 is a visualization of our results. The top three rows are the samples from Vanilla VAE, and the bottom three rows are the samples from Hyperbolic VAE. Each row consists of samples generated from latent variables of the form $a\tilde{v}/\|\tilde{v}\|_2$ with positive scalar

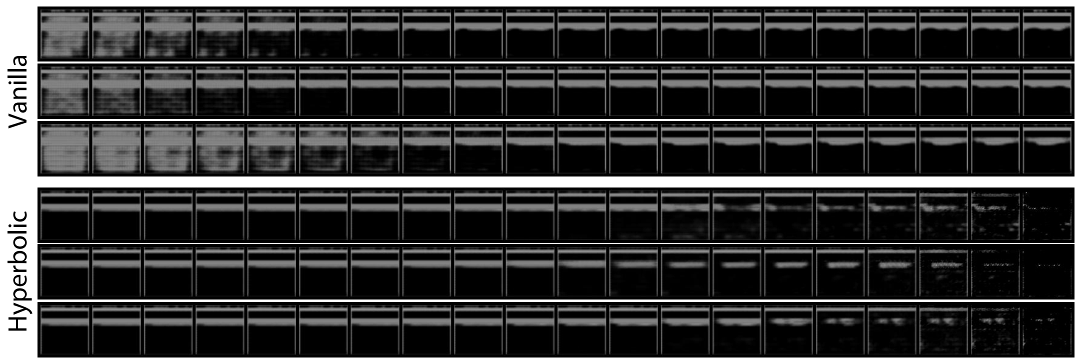


Figure 4.6.3: Samples from Vanilla and Hyperbolic VAEs trained on Atari 2600 Breakout screens. Each row was generated by sweeping the norm of $\tilde{v}$ from 1.0 to 10.0 in a log-scale.

a in range $[1, 10]$. Samples in each row are listed in increasing order of a. For Vanilla VAE, we used $\mathcal{N}(0, I)$ as the prior. For Hyperbolic VAE, we used $\mathcal{G}(\boldsymbol{\mu}_0, I)$ as the prior. We can see that the number of blocks decreases gradually and consistently in each row for Hyperbolic VAE. Please see Section 4.B.2 for more details and more visualizations.

In Breakout, the number of blocks is always finite, and blocks are located only in a specific region. Let's refer to this specific region as R. In order to evaluate each model-output based on the number of blocks, we binarized each pixel in each output based on a prescribed luminance threshold and measured the proportion of the pixels with pixel value 1 in the region R. For each generated image, we used this proportion as the measure of the number blocks contained in the image.

Figure 4.6.4 shows the estimated proportions of remaining blocks for Vanilla and Hyperbolic VAEs with different norm of $\tilde{v}$. For Vanilla VAE, samples generated from $\tilde{v}$ with its norm as large as $\|\tilde{v}\|_2 = 200$ contained considerable amount of blocks. On the other hand, the number of blocks contained in a sample generated by Hyperbolic VAE decreased more consistently with the norm of $\|\tilde{v}\|_2$. This fact suggests that the cumulative reward up to a given state can be approximated well by the norm of Hyperbolic VAE's latent representation. To validate this, we computed latent representation for each state in the test set and measured its correlation with the cumulative reward. The correlation was 0.846 for the Hyperbolic VAE. For the Vanilla VAE, the correlation was 0.712. We emphasize that no information regarding the reward was used during the training of both Vanilla and Hyperbolic VAEs.

4.6.4 *Word Embeddings*

Lastly, we applied hyperbolic wrapped distribution to word embedding problem. We trained probabilistic word embedding models with WordNet nouns dataset (Miller, 1998) and eval-

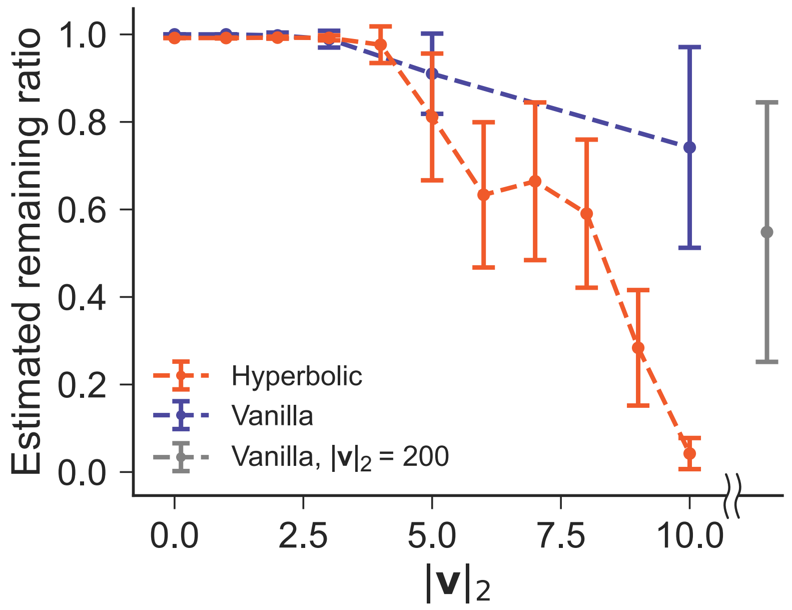


Figure 4.6.4: Estimated proportions of remaining blocks for Vanilla and Hyperbolic VAEs trained on Atari 2600 Breakout screens as they vary with the norm of latent variables sampled from a prior.

uated the reconstruction performance of them (Table 4.6.3). We followed the procedure of Poincaré embedding (Nickel and Kiela, 2017) and initialized all embeddings in the neighborhood of the origin. In particular, we initialized each weight in the first linear part of the embedding by $\mathcal{N}(0, 0.01)$. We treated the first 50 epochs as a burn-in phase and reduced the learning rate by a factor of 40 after the burn-in phase.

In Table 4.6.3, 'Euclid' refers to the word embedding with Gaussian distribution on Euclidean space (Vilnis and McCallum, 2015), and 'Hyperbolic' refers to our proposed method based on hyperbolic wrapped distribution. Our hyperbolic model performed better than Vilnis' Euclidean counterpart when the latent space is low dimensional. We used diagonal variance for both models above.

We also performed the same experiment with unit variance(Table 4.6.4). When the dimensions of the latent variable are small, the performance of the model on hyperbolic space did not deteriorate much by changing the variance from diagonal to unit. However, the same change dramatically worsened the performance of the model on Euclidean space.

	Euclid		Hyperbolic		Nickel and Kiela (2017)	
n	MAP	Rank	MAP	Rank	MAP	Rank
5	0.296 ±.006	25.09 ±.80	**0.506** ±.017	**20.55** ±1.34	0.823	4.9
10	0.778 ±.007	**4.70** ±.05	**0.795** ±.007	5.07 ±.12	0.851	4.02
20	0.894 ±.002	**2.23** ±.03	**0.897** ±.005	2.54 ±.20	0.855	3.84
50	0.942 ±.003	1.51 ±.04	**0.975** ±.001	**1.19** ±.01	0.86	3.98
100	0.953 ±.002	1.34 ±.02	**0.978** ±.002	**1.15** ±.01	0.857	3.9

Table 4.6.3: Experimental results of the reconstruction performance on the transitive closure of the WordNet noun hierarchy for several latent space dimension n. We calculated the mean and the ± 1 SD with three different experiments.

	Euclid		Hyperbolic	
n	MAP	Rank	MAP	Rank
5	0.217 ±.008	55.28 ±3.54	**0.529** ±.010	**22.38** ±.70
10	0.698 ±.030	6.54 ±.65	**0.771** ±.006	**5.89** ±.29
20	0.832 ±.016	3.08 ±.16	**0.862** ±.002	**2.80** ±.13
50	**0.910** ±.006	**1.78** ±.071	0.903 ±.003	1.94 ±.03
100	0.882 ±.008	4.75 ±2.01	**0.884** ±.003	**2.57** ±.09

Table 4.6.4: Experimental results of the word embedding models with unit variance on the WordNet noun dataset. We calculated the mean and the ± 1 SD with three different experiments.

4.7 CONCLUSION

In this paper, we proposed a novel parametrizaiton for the density of hyperbolic wrapped distribution that can both be differentiated and evaluated analytically. Our experimental results on hyperbolic word embedding and hyperbolic VAE suggest that there is much more room left for the application of hyperbolic space. Our parametrization enables gradient-based training of probabilistic models defined on hyperbolic space and opens the door to the investigation of complex models on hyperbolic space that could not have been explored before.

CHAPTER APPENDIX

4.A VISUAL EXAMPLES OF HYPERBOLIC WRAPPED DISTRIBUTION

Figure 4.A.1 shows examples of hyperbolic wrapped distribution $\mathcal{G}(\boldsymbol{\mu}, \Sigma)$ with various μ and Σ. We plotted the log-density of these distributions by heatmaps. We designate the $\boldsymbol{\mu}$ by the $\times$ mark. The right side of these figures expresses their log-density on the Poincaré ball model, and the left side expresses the same one on the corresponding tangent space.

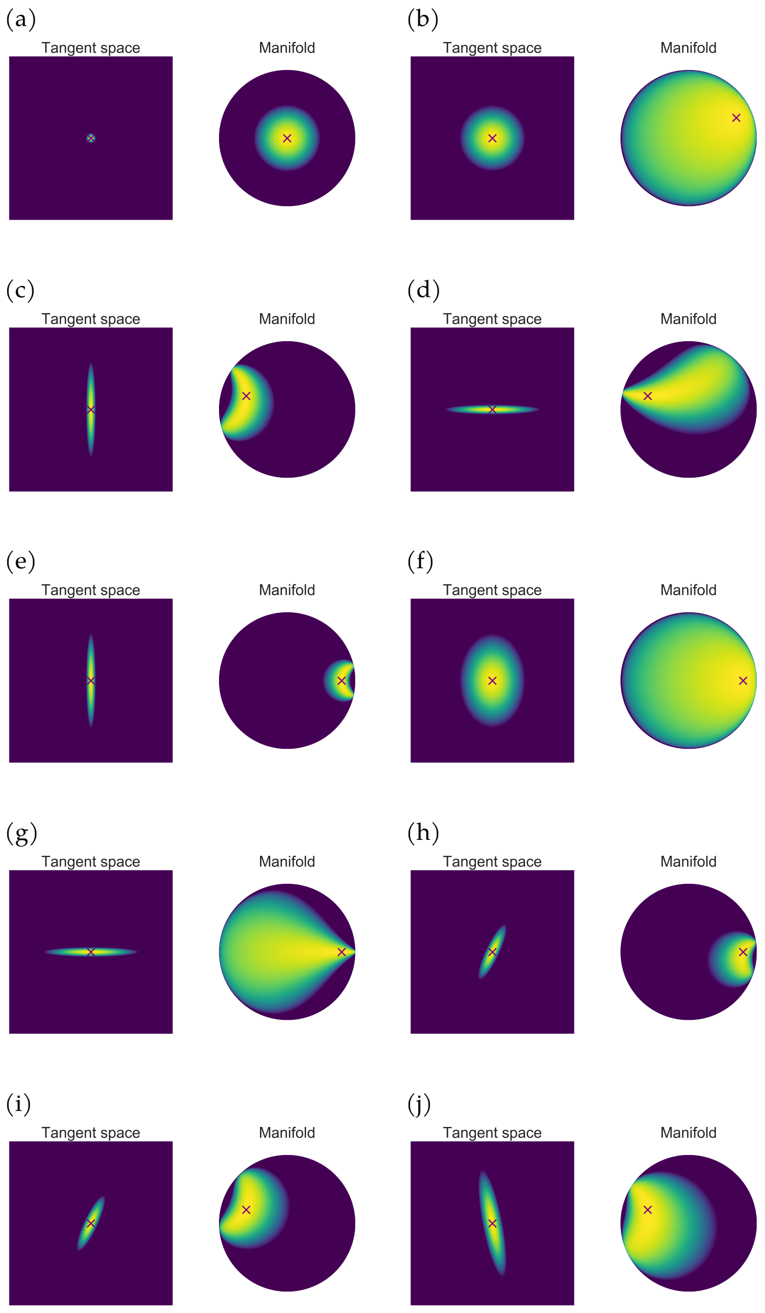


Figure 4.A.1: Visual examples of hyperbolic wrapped distribution on $\mathbb{H}^2$. Log-density is illustrated on $\mathcal{B}^2$ by translating each point from $\mathbb{H}^2$ for clarity. We designate the origin of hyperbolic space by the $\times$ mark.

4.B ADDITIONAL NUMERICAL EVALUATIONS

4.B.1 *Synthetic Binary Tree*

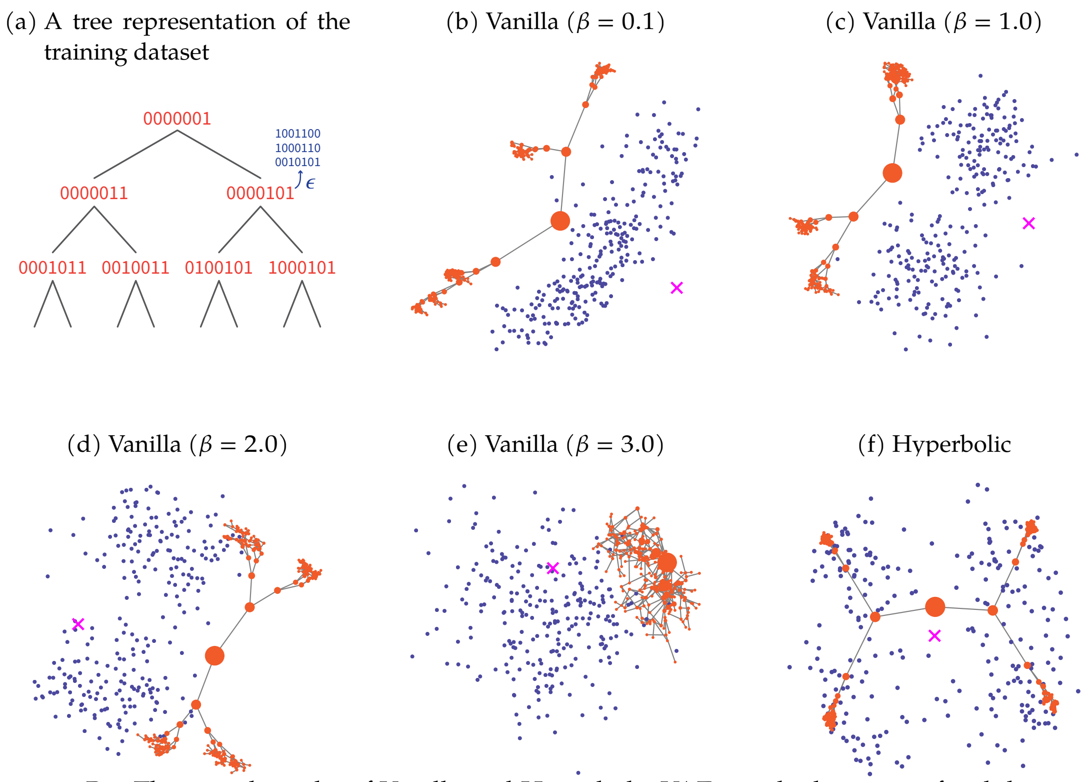


Figure 4.B.1: The visual results of Vanilla and Hyperbolic VAEs applied to an artificial dataset generated by applying a random perturbation to a binary tree. The visualization is being done in the Poincaré ball. Red points are the embeddings of the original tree, and the blue points are the embeddings of all other points in the dataset. Pink × represents the origin of hyperbolic space. Note that the hierarchical relations in the original tree was **not** used during the training phase.

We qualitatively compared the learned latent space of Vanilla and Hyperbolic VAEs. Figure 4.B.1 shows the embedding vectors of the synthetic binary tree dataset on the two-dimensional latent space. We evaluated the latent space of Vanilla VAE with $\beta = 0.1, 1.0, 2.0,$ and 3.0, and Hyperbolic VAE. Note that the hierarchical relations in the original tree were **not** used during the training phase. Red points are the embeddings of the noiseless observations. As we mentioned in Section 4.6.1, we evaluated the correlation coefficient between the Hamming distance on the data space and the hyperbolic (Euclidean for Vanilla VAEs) distance on the latent space. Consistently with this metric, the latent space of the Hyperbolic

VAE captured the hierarchical structure inherent in the dataset well. In the comparison between Vanilla VAEs, the latent space captured the hierarchical structure according to increase the β. However, the posterior distribution of the Vanilla VAE with $\beta = 3.0$ collapsed and lost the structure. Also, the blue points are the embeddings of noisy observation, and pink × represents the origin of the latent space. In latent space of Vanilla VAEs, there was bias in which embeddings of noisy observations were biased to the center side.

4.B.2 *Atari 2600 Breakout*

To evaluate the performance of Hyperbolic VAE for hierarchically organized dataset according to time development, we applied our Hyperbolic VAE to a set of trajectories that were explored by an agent with a trained policy during multiple episodes of Breakout in Atari 2600. We used a pretrained Deep Q-Network to collect trajectories, and Figure 4.B.2 shows examples of observed screens.

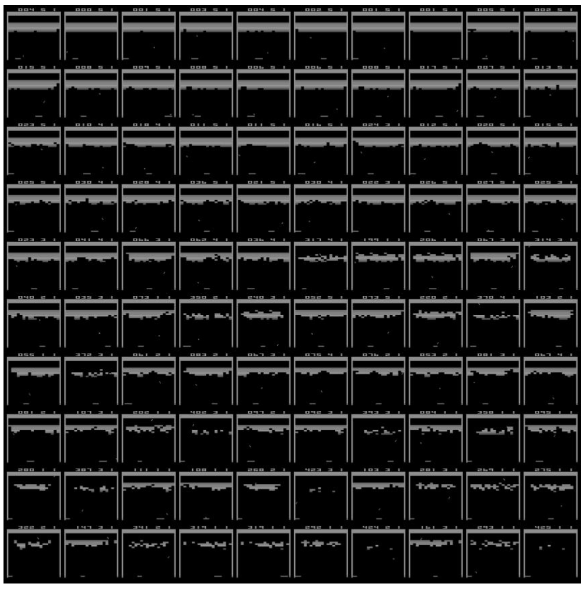
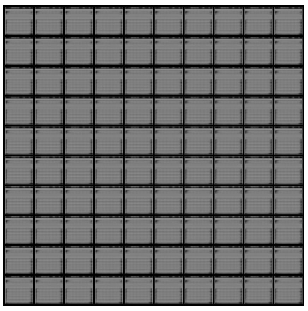


Figure 4.B.2: Examples of observed screens in Atari 2600 Breakout.

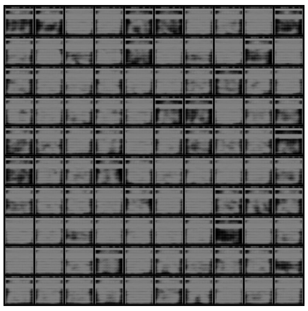
We showed three trajectories of samples from the prior distribution with the scaled norm for both models in Section 4.6.3. We also visualize more samples in Figure 4.B.3 and Figure 4.B.4. For both models, we generated samples with $\|\tilde{v}\|_2 = 0, 1, 2, 3, 5,$ and 10.

Vanilla VAE tended to generate oversaturated images when the norm $\|\tilde{v}\|$ was small. Although the model generated several images which include a small number of blocks as the norm increases, it also generated images with a constant amount of blocks even $\|\tilde{v}\| = 10$. On the other hand, the number of blocks contained in the generated image of Hyperbolic VAE gradually decreased according to the norm.

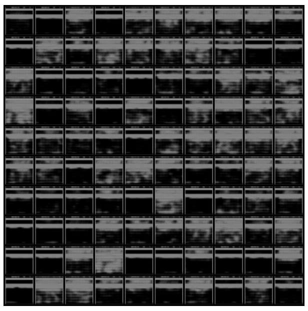
(a) $\|\tilde{v}\|_2 = 0$



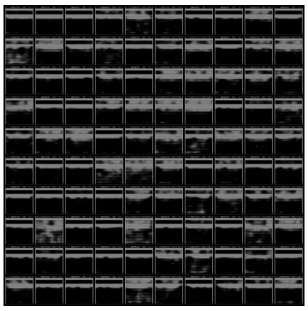
(b) $\|\tilde{v}\|_2 = 1$



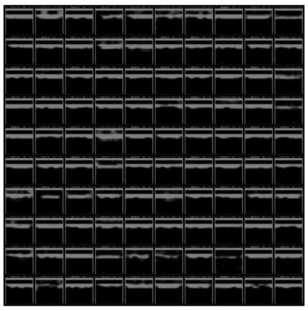
(c) $\|\tilde{v}\|_2 = 2$



(d) $\|\tilde{v}\|_2 = 3$



(e) $\|\tilde{v}\|_2 = 5$



(f) $\|\tilde{v}\|_2 = 10$

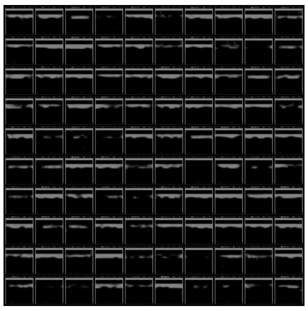
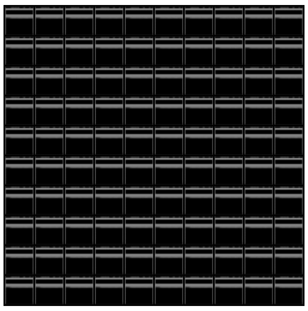
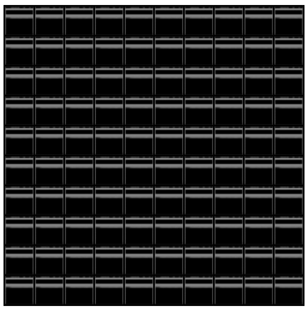


Figure 4.B.3: Images generated by Vanilla VAE with constant norm $\|\tilde{v}\|_2 = a$.

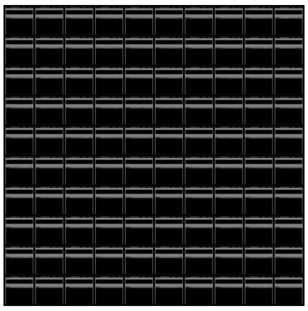
(a) $\|\tilde{v}\|_2 = 0$



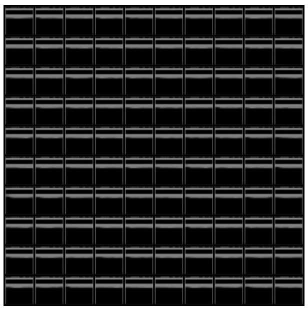
(b) $\|\tilde{v}\|_2 = 1$



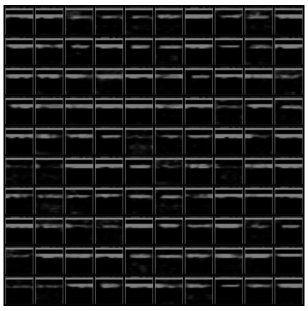
(c) $\|\tilde{v}\|_2 = 2$



(d) $\|\tilde{v}\|_2 = 3$



(e) $\|\tilde{v}\|_2 = 5$



(f) $\|\tilde{v}\|_2 = 10$

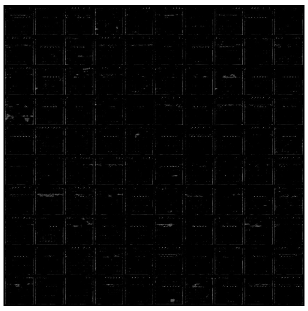


Figure 4.B.4: Images generated by Hyperbolic VAE with constant norm $\|\tilde{v}\|_2 = a$.

4.C NETWORK ARCHITECTURE

Table 4.C.1 shows the network architecture that we used in Breakout experiments. We evaluated Vanilla and Hyperbolic VAEs with a DCGAN-based architecture (Radford, Metz, and Chintala, 2015) with the kernel size of the convolution and deconvolution layers as 3. We used leaky ReLU nonlinearities for the encoder and ReLU nonlinearities for the decoder. We set the latent space dimension as 20. We gradually increased β from 0.1 to 4.0 linearly during the first 30 epochs. To ensure the initial embedding vector close to the origin, we initialized γ for the batch normalization layer (Ioffe and Szegedy, 2015) of the encoder as 0.1. We modeled the probability distribution of the data space $p(\boldsymbol{x}|\boldsymbol{z})$ as Gaussian, so the decoder output a vector twice as large as the original image.

Encoder	
Layer	Size
Input	$80 \times 80 \times 1$
Convolution	$80 \times 80 \times 16$
BatchNormalization	
Convolution	$40 \times 40 \times 32$
BatchNormalization	
Convolution	$40 \times 40 \times 32$
BatchNormalization	
Convolution	$20 \times 20 \times 64$
BatchNormalization	
Convolution	$20 \times 20 \times 64$
BatchNormalization	
Convolution	$10 \times 10 \times 64$
Linear	$2n$

Decoder	
Layer	Size
Linear	$10 \times 10 \times 64$
BatchNormalization	
Deconvolution	$20 \times 20 \times 32$
BatchNormalization	
Convolution	$20 \times 20 \times 32$
BatchNormalization	
Deconvolution	$40 \times 40 \times 16$
BatchNormalization	
Convolution	$40 \times 40 \times 16$
Deconvolution	$80 \times 80 \times 2$
Convolution	$80 \times 80 \times 2$

Table 4.C.1: Network architecture for Atari 2600 Breakout dataset.

5 SELF-SUPERVISED META-LEARNING FOR LOCAL STRUCTURE

Extracting the hidden structure of the external environment is an essential component of intelligent agents and human learning. Especially, disentanglement, the decomposition of the small number of factors that control high-dimensional observations, has attracted notable attention. The disentangled representations are known to be unachievable in general without any inductive biases on the dataset, although an explicit form of the bias was less considered. In this study, we introduce the additional assumption on the dataset that the magnitude of fluctuation can categorize the disentanglement factors into at least two groups: global and local factors. We propose the local variational autoencoder (VAE), which generalizes the VAE to have the different model parameters for each local subset and train these local parameters by the gradient-based meta-learning. Our empirical results showed that the local VAE succeeded in learning the locally disentangled representations against the dataset with local structure, including the 3D Shapes dataset, and generated high-quality images.

5.1 INTRODUCTION

Extracting the hidden structure of the external environment is essential for achieving intelligent agents and for modeling human learning (Achille et al., 2018; Higgins et al., 2017; Kemp and Tenenbaum, 2008; Lake, Salakhutdinov, and Tenenbaum, 2015; Saxe, McClelland, and Ganguli, 2019). Human beings and animals can effectively learn internal representations from a few experiences. Owing to the development of deep generative models (Goodfellow et al., 2014; Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014; Rezende and Mohamed, 2015), we can now handle high-dimensional datasets for many individual problems.

Since most datasets tend to be governed by the consistent rules of the physical world (Achille et al., 2018), it is reasonable to expect that a dataset can be modeled by a small number of control parameters even when high-dimensional observations are involved. For instance, human-made objects of the same shape or size often have multiple color options, and human faces can be decomposed into a variety of factors such as age, gender, and facial

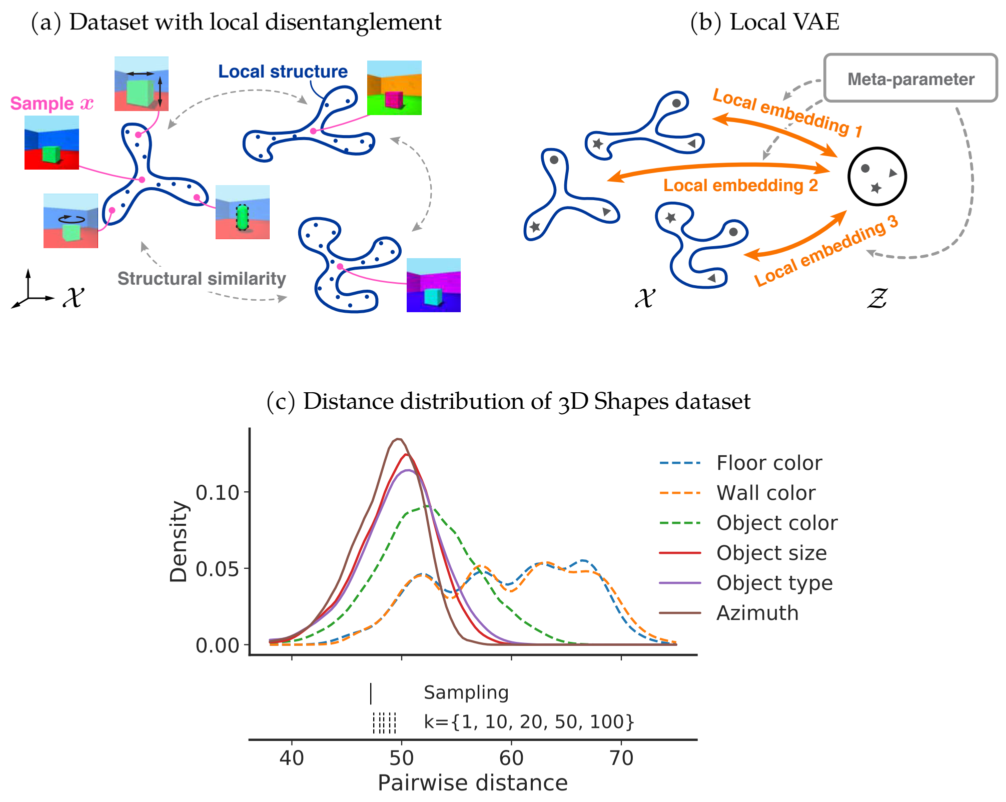


Figure 5.1.1: (a-b): Schematic diagrams of a dataset and the proposed model with the assumption of local disentanglement. (c): Empirical distribution of the pairwise ℓ^2 distance evaluated on the 3D Shapes dataset. Please see Section 5.5 for further details.

expressions. A representation that independently extracts such control factors is referred to as a disentangled representation (Higgins et al., 2017, 2018); accordingly, such representations have attracted notable attention in the extraction of structures using variational autoencoders (VAEs) (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014).

β-VAE is a typical model that aims to achieve disentangled representation (Higgins et al., 2017). This model forces the latent space of VAE to become dimensionally independent by strongly regularizing the posterior distribution of the latent variable to be isotropic Gaussian. To date, many similar approaches that extend the objective function assuming only the independence of disentanglement factors have been studied (Burgess et al., 2017; Chen et al., 2018; Kim and Mnih, 2018; Kumar, Sattigeri, and Balakrishnan, 2018). However, Locatello et al. (2019) showed that disentanglement is fundamentally impossible without inductive biases both on models and datasets. Therefore, we consider explicitly introducing inductive biases to a dataset.

In this study, we introduce the locality of disentanglement factors as an inductive bias to a dataset. The disentanglement factors that we are interested in often have a typical scale. Some factors cause datasets to fluctuate significantly, while others have small fluctuations in the data space. For example, for a human-made object, a change in color is larger than a change in other appearances including object size, type, and azimuth (Figure 5.1.1c). Furthermore, in the case of human facial images, although people of every different age, gender, etc., have common facial expressions (Ekman and Keltner, 1997), the actual facial expressions of emotions (for example, laughing or crying) can drastically vary from person to person. In other words, the local factor named facial expression can depend on the global factor named individual differences. Figure 5.1.1a shows a schematic representation of a dataset where the local factor depends on the global factor. Since such assumptions are considered to be universal regardless of the domain of the dataset, the locality is a promising candidate for solving disentanglement.

When we need to model such a context-based modulation, meta-learning is an effective approach. Meta-learning algorithms quickly learn the rules for each task for a dataset consisting of multiple tasks (Andrychowicz et al., 2016; Bengio et al., 1992; Finn, Abbeel, and Levine, 2017; Ravi and Larochelle, 2017; Schmidhuber, 1987). In this study, by considering each local structure as a task for meta-learning, we extract the transferable knowledge between each local structure. We propose a meta-embedding model, the parameters of which capture the common local structure and quickly adapt to each subspace by utilizing the structural similarity.

Here, we generalize the typical deep generative model known as the VAE thereby making it applicable to a dataset with local structure. We extend the graphical model of the VAE to contain different model parameters for each local subset of the dataset (Figure 5.1.1b), and perform the knowledge transfer by using the gradient-based meta-learning (Finn, Abbeel, and Levine, 2017; Grant et al., 2018). By treating the neighborhood of each data point as a task to adopt meta-learning, our proposed local VAE is able to quickly learn similar structures between neighbors. We show that the local VAE can naturally express the locally disentangled representation, which is defined as the context-dependent decomposition with respect to a group action. This definition is a natural extension of the definition proposed by Higgins et al. (2018). We can interpret the original disentanglement as the special case where the structure of each local subset is exactly identical throughout the entire dataset, so our proposed model has broader applicability. We also evaluate the performance of our proposed model with the 3D Shapes dataset (Burgess and Kim, 2018) and a concatenated dataset consisting of Cars3D (Reed et al., 2014) and SmallNORB (LeCun, Huang, and Bottou, 2004). Numerical experiments show that the locality enables the proposed model to achieve a disentangled representation for each subspace without any label information.

5.2 BACKGROUND

5.2.1 *Variational Autoencoder*

First, we introduce the VAE (Kingma and Welling, 2013; Rezende, Mohamed, and Wierstra, 2014), which is a deep generative model that has been studied extensively in recent years. The objective function of the VAE is defined as the variational lower bound of the log-likelihood (referred to as the evidence lower bound, ELBO). Given the dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$, the ELBO is defined as follows for each $\boldsymbol{x}^{(i)}$:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \geq & \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z})\right] - D_{\mathrm{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})\Big\|p_{\boldsymbol{\theta}}(\boldsymbol{z})\Big) \\ = & -\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}),\end{aligned} \tag{5.1}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z})$ is the conditional likelihood referred to as the decoder, and $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})$ is the variational posterior distribution referred to as the encoder. The distribution of the prior $p_{\boldsymbol{\theta}}$ is typically the standard normal, and the posterior distribution is also variationally approximated by a Gaussian. This parametric formulation of $q_{\boldsymbol{\phi}}$ is called the reparameterization trick, which enables the evaluation of the gradient of the objective function with respect to the network parameters. Overall, we can train the decoder and encoder networks by minimizing the negative ELBO using the gradient descent method.

5.2.2 *Model-Agnostic Meta-Learning*

Next, to incorporate context-dependent modulation into the VAE, we utilize model-agnostic meta-learning (MAML) (Finn, Abbeel, and Levine, 2017), which is a gradient-based meta-learning algorithm. The goal of MAML is to find task-independent knowledge from a number of previous related tasks. Once the meta-learner learns this task-independent knowledge, the model can quickly adapt to a new task using only a few data points and training iterations. To connect this algorithm to the probabilistic inference, we introduce a the maximum likelihood formulation (Grant et al., 2018) instead of the original formulation. In the maximum likelihood setting, each data point is assumed to be sampled from a task-specific distribution $\boldsymbol{x}_t^{(1)}, \ldots, \boldsymbol{x}_t^{(N_t)} \sim p_{\mathcal{T}_t}(\boldsymbol{x}_t)$. The objective function of MAML in this setting is

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T}\left[\frac{1}{N_t}\sum_{i=1}^{N_t} -\log p\Big(\boldsymbol{x}_t^{(i)} \mid \underbrace{\boldsymbol{\theta} - \alpha\nabla_{\boldsymbol{\theta}}\frac{1}{N_t}\sum_{j=1}^{N_t} -\log p(\boldsymbol{x}_t^{(j)}|\boldsymbol{\theta})}_{\boldsymbol{\theta}_t'}\Big)\right], \tag{5.2}$$

where $\boldsymbol{\theta}_t'$ represents the task-specific parameter after a single batch update by gradient descent from $\boldsymbol{\theta}$. The meta-learner can acquire the parameter $\boldsymbol{\theta}$ and can quickly adapt to new tasks with a small amount of data by optimizing Equation 5.2 using the gradient descent method. We note that $\boldsymbol{\theta}$ can be interpreted as the parameter of the prior distribution for the task-specific parameter $\boldsymbol{\theta}_t$. By replacing the expectation with repsect to the original posterior distribution by the maximum likelihood estimate $\int f(\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t|\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}_t \simeq f(\boldsymbol{\theta}'_t)$, the abovementioned objective function (Equation 5.2) recovers.

5.3 LOCAL VARIATIONAL AUTOENCODER

In this section, we will present the local VAE, a variant of the VAE suitable for representation learning of a dataset with the assumption of local disentanglement.

Here, we extend the objective function of the VAE (Equation 5.1) to have different parameters for each local subset. First, we assume T local subsets and the data points for each t-th local subset is sampled in an iid manner: $\boldsymbol{x}_t^{(i)} \overset{\text{iid}}{\sim} p_{\mathcal{T}_t}(\boldsymbol{x}_t)$. Suppose that we have a dataset $\mathcal{D} = \{\boldsymbol{x}_t^{(i)}\}_{i=1...N_t, t=1...T}$. Here, we model the data-generating distribution for t-th local subset by the parameter $\boldsymbol{\theta}_t$ and the latent variable z_t. Based on the assumption that each local subset shares transferable knowledge, we introduce the meta-parameter $\boldsymbol{\theta}$ as a prior distribution of this local parameter. The overall graphical model of the local VAE is as shown in Figure 5.3.1. The model performs the probabilistic inference through the conditional distribution from the meta-parameters.

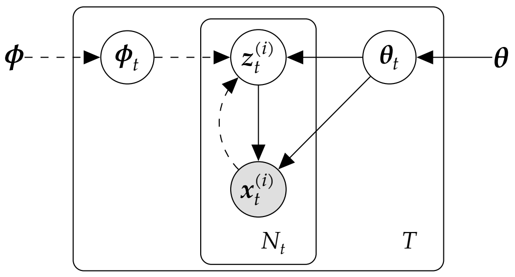


Figure 5.3.1: The graphical model of a Local VAE.

We consider the following lower bound of the log-likelihood by using Jensen's inequality:

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(i)}) &= \log \int p(\boldsymbol{x}_t^{(i)}|z_t, \boldsymbol{\theta}_t)p(z_t|\boldsymbol{\theta}_t)p_{\boldsymbol{\theta}}(\boldsymbol{\theta}_t)\mathrm{d}z_t\mathrm{d}\boldsymbol{\theta}_t \\ &= \log \int \left\{ \int \frac{q(z_t|\boldsymbol{x}_t^{(i)}, \boldsymbol{\phi}_t)}{q(z_t|\boldsymbol{x}_t^{(i)}, \boldsymbol{\phi}_t)} q_{\boldsymbol{\phi}}(\boldsymbol{\phi}_t)\mathrm{d}\boldsymbol{\phi}_t \right\} p(\boldsymbol{x}_t^{(i)}|z_t, \boldsymbol{\theta}_t)p(z_t|\boldsymbol{\theta}_t)p_{\boldsymbol{\theta}}(\boldsymbol{\theta}_t)\mathrm{d}z_t\mathrm{d}\boldsymbol{\theta}_t \\ &\geq \int p_{\boldsymbol{\theta}}(\boldsymbol{\theta}_t)q_{\boldsymbol{\phi}}(\boldsymbol{\phi}_t)q(z_t|\boldsymbol{x}_t^{(i)}, \boldsymbol{\phi}_t) \log \frac{p(\boldsymbol{x}_t^{(i)}|z_t, \boldsymbol{\theta}_t)p(z_t|\boldsymbol{\theta}_t)}{q(z_t|\boldsymbol{x}_t^{(i)}, \boldsymbol{\phi}_t)} \mathrm{d}z_t\mathrm{d}\boldsymbol{\theta}_t\mathrm{d}\boldsymbol{\phi}_t \\ &= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\theta}_t)q_{\boldsymbol{\phi}}(\boldsymbol{\phi}_t)}\Big[\mathbb{E}_{q(z_t|\boldsymbol{x}_t^{(i)}, \boldsymbol{\phi}_t)}\big[\log p(\boldsymbol{x}_t^{(i)}|z_t, \boldsymbol{\theta}_t)\big] - D_{\mathrm{KL}}\big(q(z_t|\boldsymbol{x}_t^{(i)}, \boldsymbol{\phi}_t)\|p(z_t|\boldsymbol{\theta}_t)\big)\Big],\end{aligned} \tag{5.3}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{\theta}_t)$ and $q_{\boldsymbol{\phi}}(\boldsymbol{\phi}_t)$ are the conditional distributions of the local parameters. We note that the integral variables of the expectation and the Kullback-Leibler divergence in Equation 5.3 are z_t, $\boldsymbol{\theta}_t$ and $\boldsymbol{\phi}_t$.

As mentioned above, the integral variables of Equation 5.3 include $\boldsymbol{\theta}_t$ and $\boldsymbol{\phi}_t$. This means that Equation 5.3 needs to take an integral of the model parameters to evaluate the objective function, while that of the vanilla VAE requires only the Monte Carlo expectation of z. Such an integral is unreasonable in deep generative models, whose model parameters are often high-dimensional. To overcome this problem, we assume the probability distribution of $\boldsymbol{\theta}_t$ ($\boldsymbol{\phi}_t$) given $\boldsymbol{\theta}$ ($\boldsymbol{\phi}$) as Gaussian of constant variance and replace the integral with the maximum likelihood estimator. We use the one-step gradient descent method to compute this estimator, as we described in Section 5.2.2. Let $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_t^{(i)})$ be the negative of the expression obtained by Equation 5.3. By replacing the integral of $\boldsymbol{\theta}_t$ and $\boldsymbol{\phi}_t$ with the maximum likelihood estimator $\boldsymbol{\theta}'_t$ and $\boldsymbol{\phi}'_t$, we obtain

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_t^{(i)}) \simeq & - \mathbb{E}_{q_{\boldsymbol{\phi}'_t}(z_t|\boldsymbol{x}_t^{(i)})}\left[\log p_{\boldsymbol{\theta}'_t}(\boldsymbol{x}_t^{(i)}|z_t)\right] + D_{\mathrm{KL}}(q_{\boldsymbol{\phi}'_t}(z_t|\boldsymbol{x}_t^{(i)}) \| p_{\boldsymbol{\theta}'_t}(z_t)) \\ =& \mathcal{L}_g(\boldsymbol{\theta}'_t, \boldsymbol{\phi}'_t; \boldsymbol{x}_t^{(i)}). \end{aligned} \tag{5.4}$$

We note that the integral variable of Equation 5.4 is now only z_t. The maximum likelihood estimator of the local parameters can be obtained by the following update rule:

$$\boldsymbol{\theta}'_t = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \frac{1}{K} \sum_{j=1}^{K} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_t^{(j)}), \tag{5.5}$$

$$\boldsymbol{\phi}'_t = \boldsymbol{\phi} - \alpha \nabla_{\boldsymbol{\phi}} \frac{1}{K} \sum_{j=1}^{K} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_t^{(j)}), \tag{5.6}$$

where K is the batch-size for the samples from the t-th local subset. $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ is the ELBO of the vanilla VAE defined in Equation 5.1. Algorithm 5.3.1 shows the overall algorithm.

From the perspective of the graphical model, our proposed algorithm corresponds to the assumption that the dataset approximately lies on multiple subsets and that each subset is generated from different parameters. Alternatively, from a meta-learning viewpoint, our objective function is consistent with the case of training VAEs by MAML when task information is given as a neighbor graph. We can also give the relationship of our model with the local learning approach that is represented by the locally linear embedding (LLE) (Roweis and Saul, 2000) (please see Section 5.6.1).

5.4 LOCALLY DISENTANGLED REPRESENTATION

In this section, we show that a computational graph of local VAEs can naturally express locally disentangled representations. First, we define a locally disentangled representation

Algorithm 5.3.1 Optimization of a local VAE

```
1: while until converge do
2:   Sample $T$ reference points as a mini-batch: $\mathcal{B} = \boldsymbol{x}_1, \dots, \boldsymbol{x}_T$
3:   for $\boldsymbol{x}_t$ in the mini-batch $\mathcal{B}$ do
4:     Sample $K$ points from the same local subset of $\boldsymbol{x}_t$: $\boldsymbol{x}_t^{(1)}, \dots, \boldsymbol{x}_t^{(K)} \sim N(\boldsymbol{x}_t)$.
5:     Evaluate the local objective $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ for the $K$-th neighborhood w.r.t. the meta-
       parameters based on Equation 5.1.
6:     Update the local parameters:
       $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta} - \alpha\nabla_{\boldsymbol{\theta}}\frac{1}{K}\sum_i \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_t^{(i)})$,
       $\boldsymbol{\phi}_t \leftarrow \boldsymbol{\phi} - \alpha\nabla_{\boldsymbol{\phi}}\frac{1}{K}\sum_i \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}_t^{(i)})$.
7:     Evaluate the global objective $\mathcal{L}_g(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t; \boldsymbol{x}_t^{(i)})$ w.r.t. the local parameters based on
       Equation 5.4.
8:   end for
9:   Update the meta-parameters:
     $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\frac{1}{TK}\sum_t\sum_i \mathcal{L}_g(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t; \boldsymbol{x}_t^{(i)})$,
     $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta\nabla_{\boldsymbol{\phi}}\frac{1}{TK}\sum_t\sum_i \mathcal{L}_g(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t; \boldsymbol{x}_t^{(i)})$.
10: end while
```

as an extension of the definition proposed by Higgins et al. (2018), who defined the disentanglement as the decomposition of (vector) spaces under a group action; we extend this decomposition itself to depend partly on the context. Then, we show the relationship between this representation and the local VAE. A brief overview of the original definition by Higgins et al. (2018) is shown in Section 5.A.

Here, we consider a group action $\cdot : A \times X \to X$. Suppose that group A is decomposed into a direct product $A = A_1 \times \cdots \times A_d$. We refer to the actions of the full group A and subgroup A_i as $\cdot$ and $\cdot_i$, respectively. We define a *locally disentangled group action* as follows.

Definition 5.4.1. *An action is locally disentangled if there is a decomposition $X = X_1 \times \cdots \times X_r \times X^\ell$ and we can take an isomorphism from X^ℓ to $X_{r+1} \times \cdots \times X_d$ depending on $\boldsymbol{x}_r \in X_1 \times \cdots \times X_r$ such that*

$$(a_1, \cdots, a_d) \cdot (x_1, \cdots, x_d) = (a_1 \cdot_1 x_1, \cdots, a_d \cdot_d x_d). \tag{5.7}$$

When X is a vector space, we require the decomposition to be a direct sum, and the isomorphism to be linear transformation, which corresponds to a change of basis.

This definition states that an element of A_i acts only on X_i and not on $X_j, j \neq i$. The definition reduces to the original disentangled group action if this isomorphism is independent of $\boldsymbol{x}_r$. We assume that each decomposed subspace X_i is one-dimensional in the following.

We then define a *locally disentangled representation* by using the above locally disentangled group action. Let W be the set of world-states, and let X be the observation. We also assume a generative process $b : W \to X$ and an inference process $g : X \to Z$. We refer to the composition of these processes as $h = g \circ b$.

Definition 5.4.2. *A representation Z is locally disentangled with respect to the decomposition $A = A_1 \times \cdots \times A_d$ if*

1. *There is an action $\cdot : A \times Z \to Z$,*
2. *The map $h : W \to Z$ is equivariant between the actions on W and Z:*

$$a \cdot h(w) = h(a \cdot w) \;\; \forall a \in A, w \in W. \tag{5.8}$$

3. *The action is locally disentangled with respect to Z.*

Finally, we show that a computational graph of our proposed local VAE can naturally express a locally disentangled representation. We show only the main claim in the following; please see Section 5.B for the detailed derivation.

The local VAE is a method for training VAE against a dataset, which we can split into local subsets, by using the MAML algorithm. The MAML is a nested loop optimization algorithm that is composed of an inner loop for each task and an outer loop. Here, we show the interpretation of the MAML algorithm as a computational graph by treating the inner loop as part of the inference model. Suppose that the outputs of the encoder and decoder of the local VAE for a specific task τ are z_τ and $\hat{x}_\tau$, respectively. We can expand these outputs as linear combinations of the scalar coefficients and their corresponding bases:

$$z_\tau = \sum_{i=1}^{d} g_i(\boldsymbol{x})\boldsymbol{e}^i + \alpha \sum_{i=1}^{|\Phi|} p^i_{\mathcal{D}_\tau} \frac{\partial}{\partial \phi_i} g(\boldsymbol{x}), \tag{5.9}$$

$$\hat{x}_\tau = \sum_{i=1}^{|X|} f_i \circ g(\boldsymbol{x})\boldsymbol{e}^i + \frac{\alpha}{2}\left\{ \sum_{i=1}^{|\Phi|} p^i_{\mathcal{D}_\tau} \nabla_z f(z) \frac{\partial}{\partial \phi_i} g(\boldsymbol{x}) + \sum_{i=1}^{|\Theta|} q^i_{\mathcal{D}_\tau} \frac{\partial}{\partial \theta_i} f(z) \right\}, \tag{5.10}$$

where g and f are the encoder and decoder, respectively, $\mathcal{D}_\tau$ is the τ-th task dataset (local subset), and $\boldsymbol{e}^i$ is the i-th orthonormal basis. Both $p^i_{\mathcal{D}_\tau}$ and $q^i_{\mathcal{D}_\tau}$ are scalar coefficients that depend only on the task dataset, not on the current input $\boldsymbol{x}$. Note that we ignored the term $O(\alpha^2)$ by regarding the inner learning rate α as being sufficiently small.

The first terms in Equation 5.9 and Equation 5.10 correspond to the map of Vanilla VAE, while the rest of the terms are the modulation terms only for the local VAE. As shown in these equations, the basis vectors for the local VAE depend on the current input $\boldsymbol{x}$, whereas the bases of the vanilla VAE are fixed. Owing to this property, the local VAE can naturally find the decomposition of the bases that depend on the current context. Such a context-dependent decomposition is exactly the definition of a locally disentangled representation provided above. For the linearized encoder of the local VAE (Equation 5.9), under some particular conditions, we can at least say the following:

Theorem 5.4.1. (*informal*) *The encoder of the local VAE can represent any locally disentangled representation up to $\mathcal{O}(\alpha^2)$ error when we can take*

$$\mathcal{D}_\tau(\boldsymbol{x}) \subseteq \{\boldsymbol{x}' | \boldsymbol{x}' = b(\boldsymbol{w}'), \forall \boldsymbol{w}' \in W \text{ and } \boldsymbol{w}'^{\ell} = \boldsymbol{w}^{\ell}\}. \tag{5.11}$$

We show the formal theorem and its derivation in Section 5.B. Note that this statement concerns the condition required for the capability of the local VAE. The identifiability and trainability of the solution are beyond the scope of this statement.

5.5 NEIGHBORHOOD CONSTRUCTION

As we mentioned before, the proposed method takes the local subsets of the entire dataset as the tasks of the MAML algorithm. The most common method for constructing local subsets is the k-nearest neighbor graph, which is built on the original data space. In general, we can arbitrarily choose how to construct the neighborhood, but the methods affects the embedding quality. Additionally, according to the constraints imposed by the abovementioned condition (Equation 5.11), it is desirable to be able to sample the subset within which the local factor is fixed for a locally disentangled representation. We evaluated two types of neighborhoods in the following experiments: *the k-nearest neighborhood in the data space* and *synthetic neighborhood by sampling*. Although the former is a typical neighborhood construction method, we propose the latter construction method for a specific dataset because of the following observation.

Figure 5.1.1c shows the pairwise distances between two images in the 3D Shapes dataset (Burgess and Kim, 2018). We picked two samples between which only the specific disentanglement factor is different, and we empirically evaluated the ℓ^2 distance between the samples. The 3D Shapes dataset contains three factors related to color (dashed lines in the figure) and to three other factors (solid lines). The magnitudes of fluctuation of the three color factors are more significant than those of the other three factors, so we expected to extract these color factors as global features. We also evaluated the pairwise distances between the k-th nearest neighbors with $k = \{1, 10, 20, 50, 100\}$ (dashed vertical lines in the lower part of the figure). We noted a small number of image pairs in which the distance is comparable to the distance of the k-nearest neighbor, even for $k = 1$.

To anchor some disentanglement factors inside the local subsets as much as possible, we propose using *the synthetic neighborhood by sampling* as the "zero"-nearest neighbor (solid vertical line). In the synthetic neighborhood by sampling, we sampled K different examples for each $\boldsymbol{x}_t$ from the noise distribution, which we assumed represent the observations of the data, and we used these examples as the neighborhood of $\boldsymbol{x}_t$.

We used the synthetic neighborhood in the experiment on the 3D Shapes dataset, and we employed the k-nearest neighborhood in the experiment on the CarsNORB dataset, which we will describe later. We also compared the performance of these two methods. We used

Facebook AI Similarity Search (Faiss) (Johnson, Douze, and Jégou, 2017) for the similarity search.

5.6 RELATED WORKS

The property of disentanglement has attracted considerable attention in the extraction of structures using VAEs (Burgess et al., 2017; Chen et al., 2018; Higgins et al., 2017; Kim and Mnih, 2018; Kumar, Sattigeri, and Balakrishnan, 2018). Most proposed models try to achieve a disentanglement representation by modifying the penalty term of the objective function or by modifying the network architectures. However, Locatello et al. (2019) showed that disentanglement is fundamentally impossible without inductive biases both in models and in datasets. On the other hand, our approach focuses on how to learn parameters suitable for (locally) disentangled representations so that we can utilize the aforementioned techniques and our proposed method at the same time.

Theoretical developments have also been achieved for disentanglement. The original concept of disentanglement was mathematically vague. Higgins et al. (2018) presented a formal definition of disentanglement in the sense of a group action. Khemakhem, Kingma, and Hyvärinen (2019) subsequently clarified that the solution of the VAE is identifiable when the latent variables are conditioned on an additionally observed variable and conditionally factorized with each other. Khemakhem, Kingma, and Hyvärinen (2019) clarified that the solution of the VAE is identifiable when the latent variables are conditionally factorized given an additional observed variable. Shu et al. (2020) investigated the theoretical limitation of some types of weak supervision for disentanglement as used in Klys, Snell, and Zemel (2018) and Bouchacourt, Tomioka, and Nowozin (2018). They divided the disentanglement into two properties: consistency and restrictiveness, and showed that a situation where a learner has multiple samples with some fixed attributes is sufficient only for the consistency of those attributes. While the way of supervision of our approach corresponds to theirs (called *match-pairing*), the model and learning algorithm is different. They assumed to capture all disentanglement factors by the latent variable and revealed only a sufficient condition. The graphical model of our method differs from theirs and introduce locality not only on the encoder but also on the decoder by MAML. MAML has a strong inductive bias to keep each task-specific parameter close to the global parameter. As will be empirically validated in Section 3.3, this inductive bias naturally evokes local disentanglement. Moreover, while they need to access at least one disentanglement factor, our method does not.

From the viewpoint of generating data by a deep generative model with some supervision, conditional generation is commonly practiced (Kingma et al., 2014; Mirza and Osindero, 2014; Sohn, Lee, and Yan, 2015). Our method is similar to this approach in that the density function is conditioned on the neighborhood of a specific data point. However, while the conventional conditional generation approach generates data by using only one model parameter with the known class label as an additional latent code, our proposed model contains different network parameters for each neighborhood. Therefore, the situation is similar to the case of *match-pairing* that these approaches need to access at least class label information.

In addition, substantial research has been conducted on the extension of generative models to make such models applicable to structured datasets. Accordingly, various researchers proposed to generalize the latent space of the VAE to a non-Euclidean space, such as a spherical surface (Davidson et al., 2018), hyperbolic space (Mathieu et al., 2019; Nagano et al., 2019; Ovinnikov, 2019), or discrete space (Jang, Gu, and Poole, 2017; Rolfe, 2017).

In this study, we employed the gradient-based MAML algorithm (Finn, Abbeel, and Levine, 2017) and its probabilistic formulation (Grant et al., 2018) to find local parameters from a few data points. Several studies (Hsu, Levine, and Finn, 2019; Metz et al., 2019) proposed the integration of unsupervised learning and meta-learning from another perspective. Hsu, Levine, and Finn (2019) proposed an algorithm to generate MAML task information by utilizing embedded similarity information created with unsupervised learning. In contrast to this case of using unsupervised learning **for** meta-learning, we used meta-learning to perform unsupervised learning. In addition, Metz et al. (2019) proposed a way to seek the objective function for representation learning with meta-learning.

Furthermore, local learning approaches, including LLE (Roweis and Saul, 2000) and Isomap (Tenenbaum, Silva, and Langford, 2000), are deeply related to our work. We discuss the relationship between LLE and our work in detail in Section 5.6.1.

5.6.1 *Relationship to Conventional Local Learning*

Here, we discuss a possible interpretation of the local VAE as a variant of conventional local learning approaches. Many studies have incorporated locality for dimensionality reduction and representational learning (Kambhatla and Leen, 1997; Roweis and Saul, 2000; Tenenbaum, Silva, and Langford, 2000). These studies aim to find mappings between the data and the coordinate space under the assumption that the data space is composed of multiple low-dimensional subspaces (Brand, 2003; Vincent and Bengio, 2003). We refer to this approach as local learning.

Here, we introduce a locally linear embedding (LLE) (Roweis and Saul, 2000) as a typical local learning algorithm. LLE extracts low-dimensional neighborhood-preserving embeddings based on a precomputed neighbor graph. This method assumes that the dataset consists of a combination of locally linear spaces and applies a linear projection to each neighborhood. For the dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$, the objective function of LLE is defined as

$$\mathcal{L}(W) = \sum_i \left\| \boldsymbol{x}^{(i)} - \sum_j W_{ij} \boldsymbol{x}^{(j)} \right\|^2, \tag{5.12}$$

where parameter W is an $N \times N$ matrix. The element W_{ij} of W is nonzero only if $\boldsymbol{x}^{(j)}$ belongs to the set of neighbors of $\boldsymbol{x}^{(i)}$, and $\sum_j W_{ij} = 0$. The neighbor graph of $\boldsymbol{x}^{(i)}$ is built by using the k-nearest neighbor method. Since Equation 5.12 is known not to have local minima, we can derive the solution of Equation 5.12 by a basic matrix calculation. Once the model parameter W has been derived, we can obtain low-dimensional embeddings $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \dots, \boldsymbol{z}^{(N)}$ of each data point by minimizing the loss $\sum_i \|\boldsymbol{z}^{(i)} - \sum_j W_{ij}\boldsymbol{z}^{(j)}\|^2$ with respect to $\boldsymbol{z}$.

Here, we consider extending the embedding model of LLE from a linear projection to a general nonlinear model. The embedding model of $\boldsymbol{x}^{(t)}$ corresponds to $\sum_j W_{tj}\boldsymbol{x}^{(j)}$ in Equation 5.12. In other words, if we denote the index of the neighborhood of $\boldsymbol{x}^{(t)}$ by $j_1, \dots, j_K$, the model parameters of the neighborhood are $\left[W_{tj_1}, W_{tj_2}, \dots, W_{tj_K}\right]$. In the following, we generalize these parameters $\left[W_{tj_1}, W_{tj_2}, \dots, W_{tj_K}\right]$ as parameter $\boldsymbol{\theta}_t$ for the same local subset of $\boldsymbol{x}^{(t)}$. Then, the aforementioned objective function is given by the following:

$$\mathcal{L}(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N) = \sum_t \left\| \boldsymbol{x}^{(t)} - g_{\boldsymbol{\theta}_t}\left(N(\boldsymbol{x}^{(t)})\right) \right\|^2, \tag{5.13}$$

where we wrote the same local subset of $\boldsymbol{x}^{(t)}$ as $N(\boldsymbol{x}^{(t)})$. Unlike Equation 5.12, there are no restrictions on the number of parameters or the formulation, so the optimization of the above equation is generally challenging. Notably, in the case of $g_{\boldsymbol{\theta}_t}(\cdot)$ being a deep neural network, a massive quantity of data and extended training time are required **for each t-th neighborhood** $N(\boldsymbol{x}^{(t)})$.

Consider taking only $\boldsymbol{x}^{(t)}$ itself instead of a set of neighborhoods $N(\boldsymbol{x}^{(t)})$ of $\boldsymbol{x}^{(t)}$ as input to function $g_{\boldsymbol{\theta}_t}(\cdot)$ in Equation 5.13. If we denote the model parameters by $\boldsymbol{\theta}_t$ and $\boldsymbol{\phi}_t$ and use an autoencoder for model $g(\cdot)$, Equation 5.13 corresponds to the objective function of the local VAE with the Gaussian decoder.

From the perspective of optimization, following a naive way of incorporating the local learning approaches into the training of deep generative models could be problematic, although it will give us a new model that has both the capacity for high-dimensional inputs and flexibility for locally changing environments at the same time. The reason is that the conventional local learning approaches train a different embedding model for each neighborhood from scratch; nevertheless, deep neural networks generally require a large quantity of data, and training them takes a long time (LeCun, Bengio, and Hinton, 2015).

Here, we show that the approach of our local VAE is superior to naive local learning in terms of computation time. We numerically evaluated the computation time of our local VAE. Figure 5.6.1 shows the computation time for each iteration for various batch sizes. We performed all experiments on a single GeForce GTX 1080Ti GPU. Each bar represents the time per iteration for each model, averaged over 1,000 trials. Since the batch size of the outer loop of MAML corresponds to the number of tasks for each iteration, the total computation time was predicted to increase linearly with the batch size. The empirical result is consistent with this prediction. Because the local VAE requires extra computation involving the gradient descent, the computation time of the local VAE was slightly longer than that of the

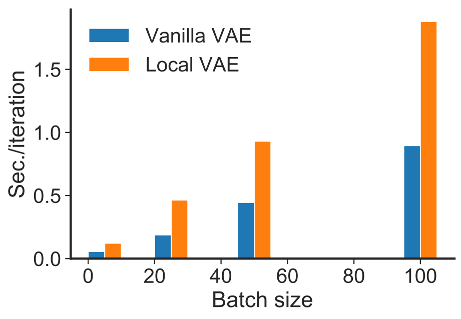


Figure 5.6.1: Comparison of computation times for various batch sizes.

vanilla VAE. However, this increase in cost remained at a constant level. On the other hand, using a naive local learning approach will require the time of a vanilla VAE's computation multiplied by the number of divisions of the entire dataset because we need the embedding model for every local subset. Since a local VAE transfers knowledge between local subsets, we can omit the calculation.

5.7 NUMERICAL EVALUATIONS

5.7.1 *3D Shapes Dataset*

Here, we numerically evaluate the performance of the local VAE on the 3D Shapes dataset. We followed all the experimental settings in Locatello et al. (2019) (except the batch size and the number of tasks) to eliminate effects not related to the proposed method as much as possible (please see Section 5.C for further details). Then we qualitatively assessed the generated images and quantitatively evaluated the model performance by using the disentanglement metric (DCI score) proposed by Eastwood and Williams (2018) and the Fréchet inception distance (FID) (Heusel et al., 2017).

Qualitative Evaluations

Figure 5.7.1 shows the conditionally generated images of the trained model. In the inference phase, the model obtains the local parameters $\boldsymbol{\theta}_t$ by applying the one-step gradient descent method using the randomly selected training data and generates images from these local parameters. We trained multiple models with different values of α, which is the hyperparameter of the local VAEs. Note that the original objective function of the vanilla VAE is recovered in the case of $\alpha = 0$ since the local parameters are strictly consistent with the meta-parameters. According to the subjective assessment, the quality of the generated images is better when α is relatively large.

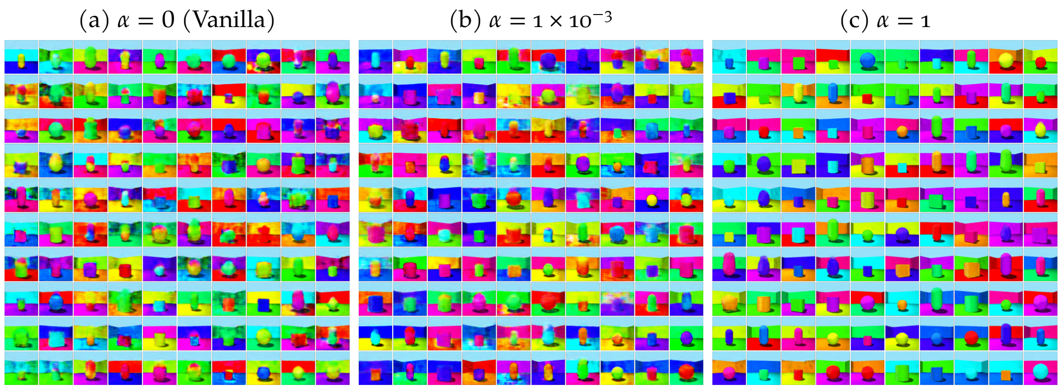


Figure 5.7.1: Qualitative comparison of the randomly-selected conditional prior samples.

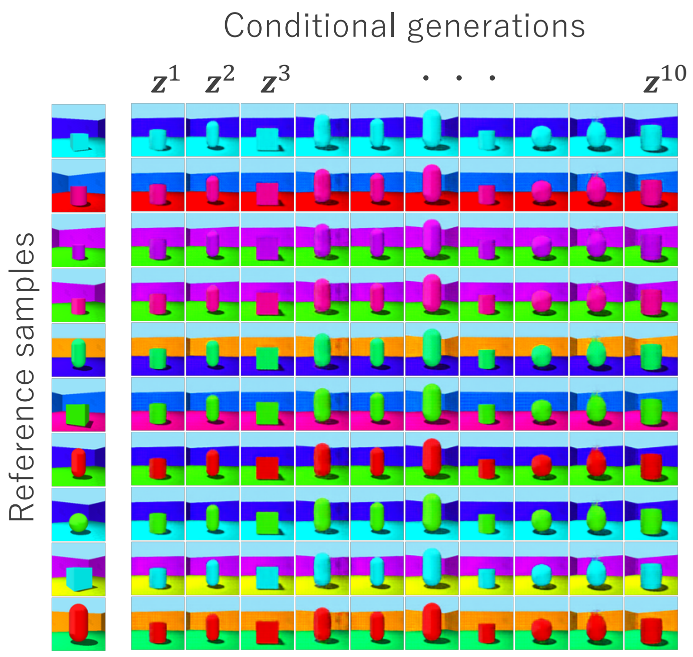


Figure 5.7.2: Reference training samples and the corresponding conditionally generated images of the local VAE with α =1. The leftmost column shows the reference training samples. Each row visualizes the generated images conditioned on the reference sample, and the images shown in the same column share their latent code. The trained model extracted color information as global features and other information as the local features.

Next, we show that the local VAE separated the global and local features spontaneously by using the meta-parameters. We illustrate the reference samples and the corresponding generated images of the local VAE with $\alpha = 1$ in Figure 5.7.2. The leftmost column shows the reference training samples used for the conditional localized generation. Each row visualizes the generated images conditioned on the corresponding reference sample in the leftmost column. We randomly picked ten latent codes $z^1, \dots, z^{10}$ from the prior distribution and then used these codes for each conditional generation. In other words, the images shown in the same column share their latent code. According to the figure, the generated images conditioned on the corresponding reference sample have the same color information as the reference. Additionally, we can see that the local VAE model did not overfit specific data because the trained model generated clearly different images in the same row conditioned on one training sample. Moreover, the shape, angle, and size of the object are the same; only the color differs in each column. These results strongly suggest that the model trained by the proposed method segregated color information as global features and other information as local features and obtained an internal representation independent of the global features. Note that the local VAE uses only neighborhood relationships and **does not use any label information**. We discussed the magnitude of fluctuation of each disentanglement factor in Section 5.5. The results shown in Figure 5.7.2 are consistent with the tendency of these fluctuations.

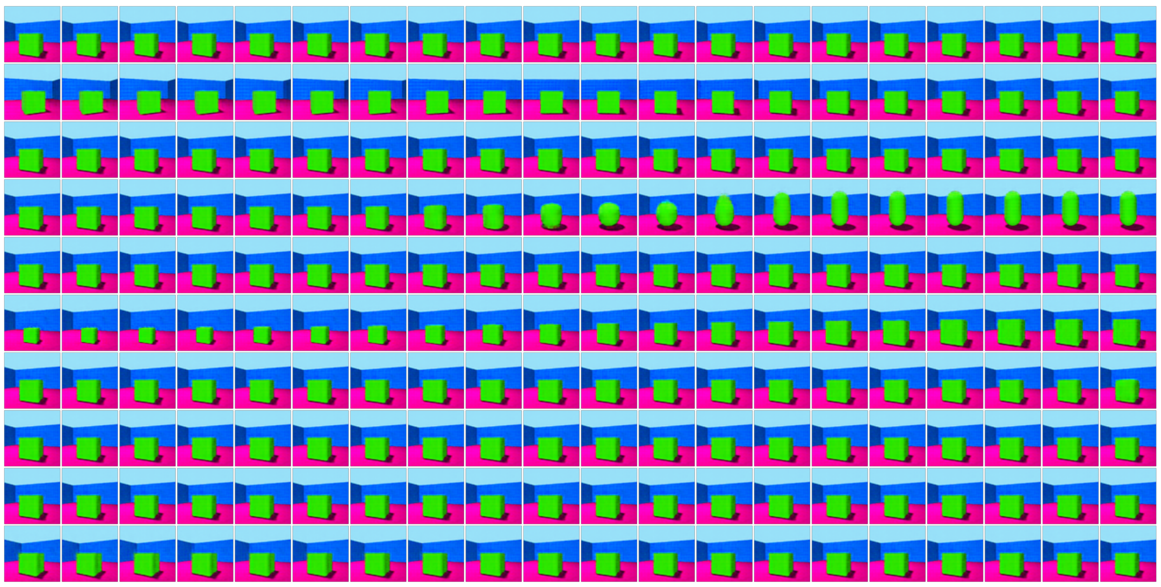


Figure 5.7.3: Interpolation of the latent space of the local VAE. Each i-th row corresponds to the reconstructed image with the latent code z_i modified in the range of $[-2, 2]$.

Figure 5.7.3 shows the learned latent space of the local VAE model. We swept each latent dimension for the specific training sample in the range of $[-2, 2]$. The model extracted the angle, shape, and size of the object as locally disentangled factors. However, the color of the reconstructed images was not changed during the interpolation since the model extracted the color information as a global factor.

Quantitative Evaluations

Then, we quantitatively evaluated the latent representations of local VAEs by using DCI scores (Eastwood and Williams, 2018). DCI scores quantify learned representations based on three aspects: disentanglement, compactness, and informativeness. All of these metrics can be computed from the importance of each dimension to the latent space in predicting a factor of variation. DCI scores require the label information of the ground truth. Since the local VAE clearly extracted the color information as the global feature, we calculated the DCI scores under two conditions: the class labels including all six aspects (w/ color) and the class labels excluding color information (w/o color). We report these values evaluated on a single trial.

Table 5.7.1 shows the empirical evaluations of the abovementioned DCI scores. The DCI scores with six labels (w/ color) of the local VAE were slightly better than those of the vanilla VAE. The model with a small α, which is closer to the value of the vanilla VAE, tended to achieve better scores for all DCI metrics in the local VAE comparison. This result is attributed to the loss of color information from the internal representation as α increases. On the other hand, the local VAEs significantly improved the DCI scores under the w/o color condition. All the DCI metrics took their maximum value at $\alpha = 1$. The performance was slightly degraded at $\alpha = 1 \times 10^1$, and the loss diverged during training at $\alpha = 1 \times 10^2$. This numerical evaluation suggests that α can control how much of the structure behind the entire dataset is regarded as global variation and from where it is regarded as a local variation. We also evaluated the performance of the β-VAE (Higgins et al., 2017) model as a reference. β-VAE modifies the KL term (Equation 5.1) through multiplication with the nozero coefficient β. Although the β-VAE models with $\beta = 8$ or $\beta = 16$ achieved higher scores than the local VAEs under the w/ color condition, the local VAE with $\alpha = 1$ significantly outperformed all the β-VAE models under the w/o color condition.

We also evaluated the quality of the generated images with the FID (Heusel et al., 2017). The FID is a metric that evaluates the similarity in the quality between real and generated images. We used the 50,000 samples within the ground truth dataset and generated images to calculate the FID. According to Table 5.7.1, the FID tended to be low at large α values and took its minimum value at $\alpha = 1$. This result is consistent with the DCI scores under the w/o color condition.

Model Comparisons

We examined that the local VAE outperformed β-VAE in terms of the DCI score and the FID. However, there are many state-of-the-art models and metrics for disentanglement.

Table 5.7.1: Quantitative evaluations of the local VAE on the 3D Shapes dataset. Highlighted cells indicate the model with the highest performance in the comparison of local VAEs. Bold numbers indicate the absolute best results.

		DCI w/ Color			DCI w/o Color			FID
		Disent.	Compl.	Inform.	Disent.	Compl.	Inform.	
Local VAE	$\alpha = 0$ (Vanilla)	0.246	0.204	0.703	0.150	0.096	0.547	134.786
	$\alpha = 1 \times 10^{-3}$	0.491	0.407	0.814	0.390	0.305	0.686	107.636
	$\alpha = 1 \times 10^{-2}$	0.449	0.385	0.797	0.173	0.132	0.635	123.288
	$\alpha = 1 \times 10^{-1}$	0.457	0.432	0.626	0.945	0.796	0.996	49.364
	$\alpha = 1$	0.424	0.406	0.594	**0.977**	**0.800**	**0.999**	**43.194**
	$\alpha = 1 \times 10^{1}$	0.393	0.370	0.587	0.871	0.733	0.998	59.555
β-VAE	$\beta = 2$	0.367	0.292	0.776	0.222	0.215	0.630	96.279
	$\beta = 4$	0.588	0.499	0.906	0.384	0.337	0.817	96.612
	$\beta = 8$	0.636	**0.584**	**0.967**	0.601	0.547	0.936	86.856
	$\beta = 16$	**0.649**	0.580	0.941	0.690	0.473	0.883	86.237

Here, we show the quantitative comparison of different state-of-the-art models, metrics, hyperparameters, and the neighborhood construction methods.

First we show the model comparison. In the following, we compared the Local VAE, β-VAE, the Factor VAE (Kim and Mnih, 2018), the DIP-VAE-I and the DIP-VAE-II (Kumar, Sattigeri, and Balakrishnan, 2018), and the Annealed VAE (Burgess et al., 2017). Figure 5.7.4 shows the full comparisons of the FID for different models and hyperparameters. The FID was remarkably reduced in the region where the α of local VAE was large. The optimal FID of the local VAE was lower than every other models.

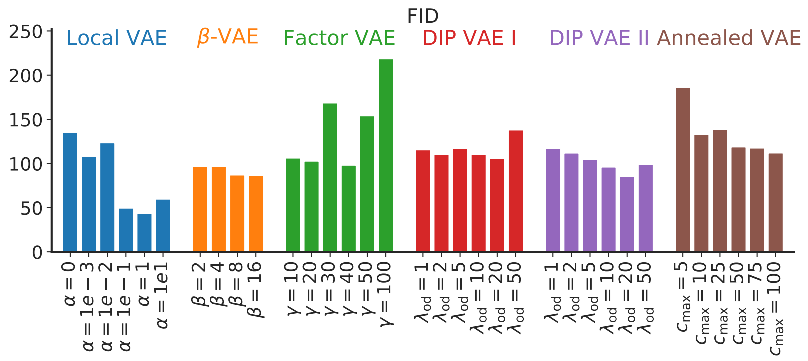


Figure 5.7.4: Quantitative comparison of the FID. Lower values are better.

Figure 5.7.5 shows the full comparisons of the DCI scores for different models and hyperparameters. Open bars correspond to the metrics for all six labels (w/ color) and solid bars correspond to the metrics for three labels (w/o color). We showed the comparisons to β-VAEs and other models with optimal hyperparameters in the main text. Similar to the results shown in the main text, this figure shows the effectiveness of local VAE, especially in the w/o color condition.

Then, we show the metrics comparison. Since Locatello et al. (2019) showed that different disentanglement scores are correlated, we evaluated disentanglement ability by the DCI scores in the main text. On the other hand, a number of state-of-the-art metrics have been proposed for disentanglement representation. We evaluated the quality of learned representations of various models in the sense of local disentanglement. We evaluated these models by the FID, the DCI scores, the SAP score (Kumar, Sattigeri, and Balakrishnan, 2018), the mutual information gap (MIG) score (Chen et al., 2018), and the Modularity measures (Ridgeway and Mozer, 2018). Figure 5.7.6 shows the comparisons about these metrics.

Similar to the case of the DCI scores, the performance of local VAE was improved in w/o color condition compared to w/ color condition, except for the Modularity score. The Modularity score uses discretized representations, so the value is higher even if the internal representation is not aligned with respect to the true factor. Such a property is suggested to be the cause that the various models including the local VAE did not show improvement. Also the Modularity score has already achieved good performance in vanilla VAE, and it is likely that it was difficult to improve it by modifying the method.

Except for the modularity score, the local VAE outperformed than the other models, especially in the w/o color condition. Almost all scores are correlated with each other and it is consistent with the report in Locatello et al. (2019).

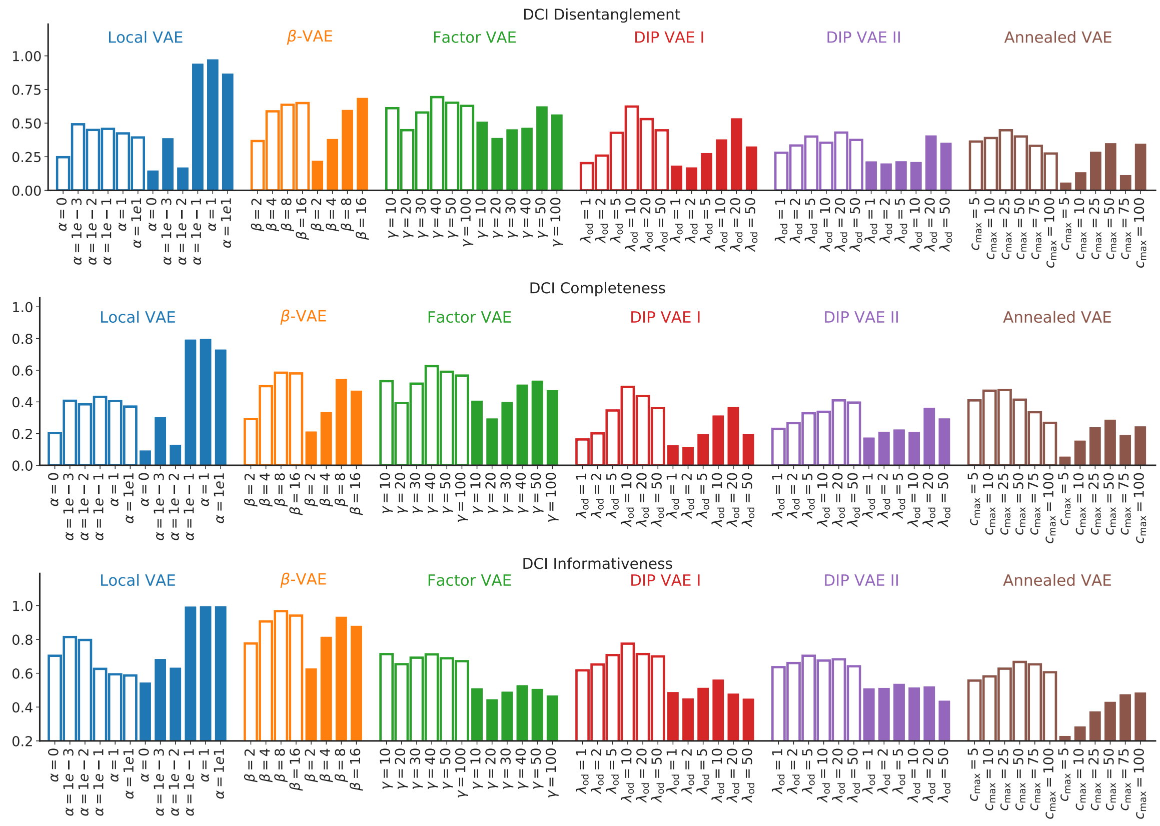


Figure 5.7.5: Quantitative comparison of the DCI scores. Higher values are better. Open bars correspond to the metrics for all six labels (w/ color) and solid bars correspond to the metrics for three labels (w/o color).

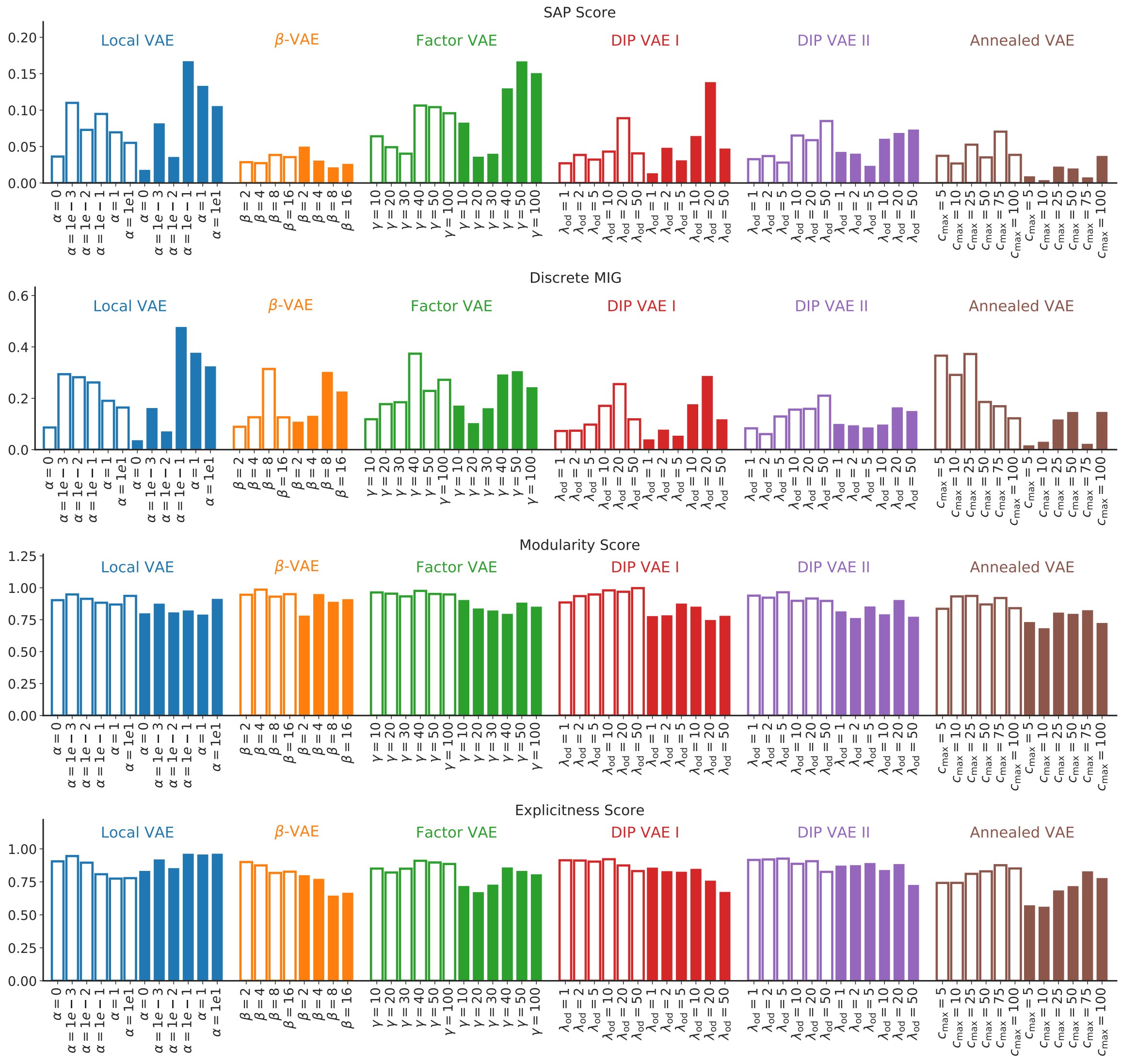


Figure 5.7.6: Quantitative comparison of various metrics. Higher values are better. Open bars correspond to the metrics for all six labels (w/ color) and solid bars correspond to the metrics for three labels (w/o color).

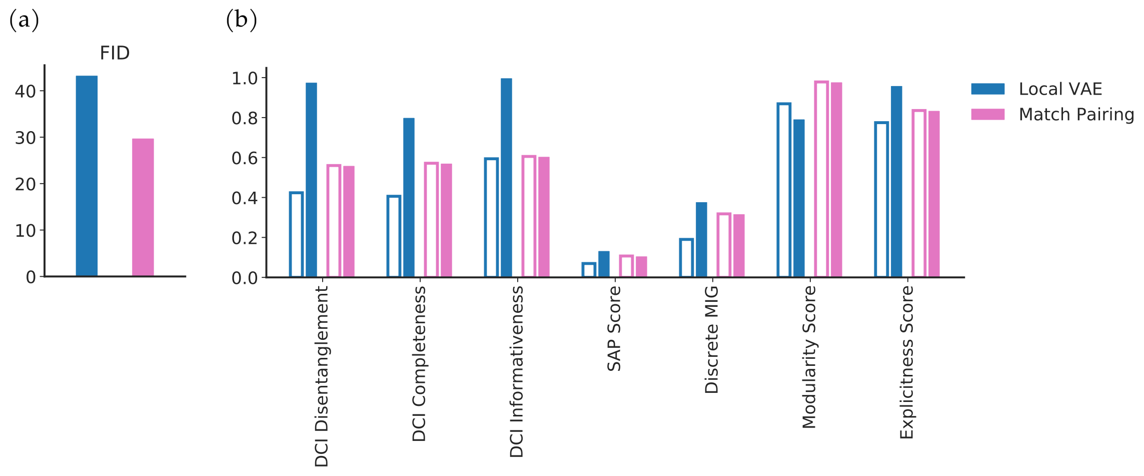


Figure 5.7.7: Model comparisons to the "match-pairing" scheme (Shu et al., 2020). (a): FID. Lower is better. (b): Various disentanglement metrics. Higher is better. Open bars correspond to the metrics for all six labels (w/ color) and solid bars correspond to the metrics for three labels (w/o color).

In Section 5.6, we argued the relationships and the difference to the representation learning presented by Shu et al. (2020). Shu et al. (2020) discussed the theoretical guarantees and limitations of some types of weak supervision for the disentanglement. While the way of supervision of our approach corresponds to one of their proposals (called *match-pairing*), the model and learning algorithm is different. Our method uses meta-learning and aims to capture global disentanglement factors by the modulation of network parameters rather than the latent variable. Here, we numerically evaluate the difference in the performance of those two approaches.

Figure 5.7.7 shows the overall comparisons of the two approaches. We fixed the all hyperparameters and network architectures to the default values. Note that Shu et al. (2020) proposed using a GAN-based generator (Goodfellow et al., 2014) rather than the VAE and separately train the encoder for the generator. Therefore, our training algorithm based on MAML and their algorithm are completely different. We see that the match-pairing scheme achieved better performance against ours in terms of the FID (Figure 5.7.7a). This result is reasonable because GAN-based generators are experimentally known to be able to generate high-fidelity samples than VAE. On the other hand, for w/o color condition, our local VAE outperformed the match-pairing scheme in terms of all metrics for the disentanglement except the Modularity score (Figure 5.7.7b). These results are consistent with the results mentioned above. As mentioned earlier, local VAEs attempt to capture global disentanglement factors by performing meta-learning using network parameters rather than latent variables alone. These results suggest that the difference in such a setup has contributed to the performance of the method as a strong inductive bias.

Choice of the Neighborhood Construction Method

Since the method used to calculate neighborhood was arbitrary, we also compared the synthetic neighborhood with a widely used approach. We evaluated two types of methods: synthetic neighborhood by sampling and k-nearest neighborhood on input space. We compare the performance of the synthetic neighborhood to the ℓ^2 distance on input space for the same dataset.

Table 5.7.2 shows the comparison of the methods to compute neighborhood. We used the 3D Shapes dataset and evaluated the performance with different values of α. Based on the table, both the synthetic neighborhood and the k-nearest neighborhood outperformed vanilla VAE, and they achieved comparable scores with each other. We note that the synthetic neighborhood by sampling achieved slightly better performance than the ℓ^2 distance on input space in this experiment. This result suggests that distinguishing the global factors is essential, as we discussed in Section 5.5. Choosing the appropriate distance will improve the quality of the learned representations and generated images in practical use.

Table 5.7.2: Quantitative comparison of the neighborhood construction on the 3D Shapes dataset.

	DCI w/ Color			DCI w/o Color			FID
	Disent.	Compl.	Inform.	Disent.	Compl.	Inform.	
$\alpha = 1 \times 10^{-3}$	0.217	0.184	0.668	0.237	0.152	0.486	123.593
$\alpha = 1 \times 10^{-2}$	0.200	0.166	0.660	0.175	0.113	0.480	115.726
$\alpha = 1 \times 10^{-1}$	0.420	0.342	0.722	0.311	0.220	0.628	110.932
$\alpha = 1$	0.397	0.342	0.582	0.676	0.520	0.813	49.187
$\alpha = 1 \times 10^{1}$	0.434	0.379	0.595	0.701	0.542	0.816	59.114
Sampling	0.424	0.406	0.594	0.977	0.800	0.999	43.194
Vanilla VAE	0.246	0.204	0.703	0.150	0.096	0.547	134.786

5.7.2 *Concatenated Dataset of Cars3D and SmallNORB*

Finally, we evaluated our proposed model on a dataset with explicit locality. In this section, we concatenated two datasets: the Cars3D dataset (Reed et al., 2014) and the SmallNORB dataset (LeCun, Huang, and Bottou, 2004). These datasets both comprise sets of images of 3D objects. The Cars3D dataset has three disentanglement factors (elevation, azimuth, and object type), while the SmallNORB dataset has four (elevation, azimuth, category, and lighting condition). The elevation and azimuth are global control factors common to the entire concatenated dataset, while the others are sub-dataset specific factors. Each image has no information about which sub-dataset it originated. We refer to this dataset as the CarsNORB dataset. Note that learning this entire dataset is more challenging than learning each subdataset since these two subdatasets have different structures.

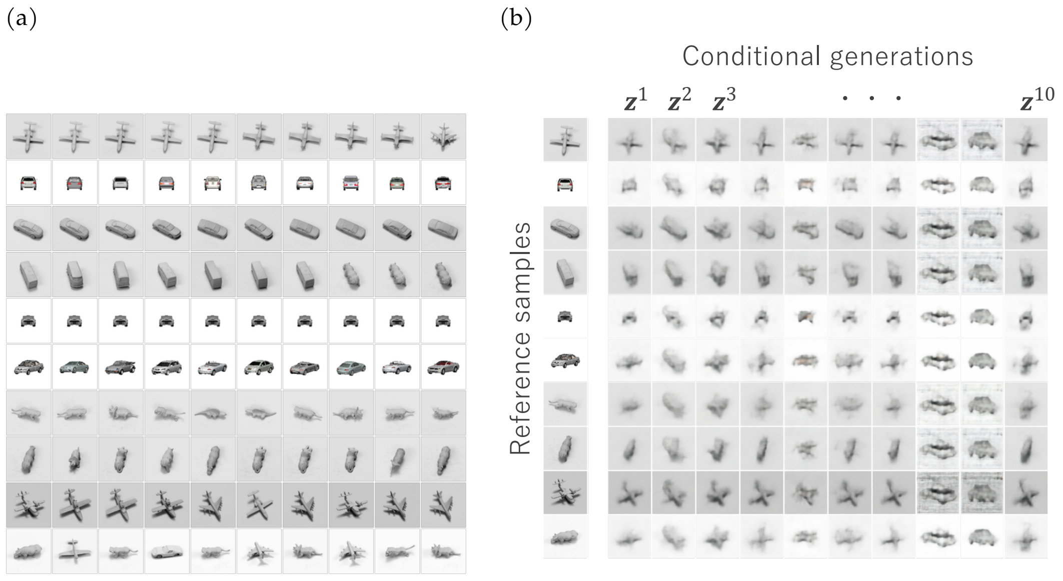


Figure 5.7.8: (a): Reference samples and its k-nearest neighbors. (b): Reference training samples and the corresponding conditionally generated images for the CarsNORB dataset. The leftmost column shows the reference training samples. Each row visualizes the generated images conditioned on the reference sample, and the images shown in the same column share their latent code.

Figure 5.7.8 shows the reference samples and the corresponding generated images of the local VAE for the CarsNORB dataset. According to the Figure 5.7.8b, the generated images shares the basic shape and lighting condition as the reference sample. Figure 5.7.8a shows the typical training samples. We can see that the local VAE was learning using tasks conditioned on the rough shape and lighting conditions. Based on the subjective assessment, the result seen in Figure 5.7.8b achieved this learning objective. We note that the training of the local VAE and its generated images highly depends on the definition of tasks (neighborhoods).

Table 5.7.3 shows the empirical evaluation on the CarsNORB dataset. The hyperparameters follow the same settings as the experiment on the 3D Shapes dataset. We calculated the DCI disentanglement, completeness and FID for each subdataset. According to the table, the disentanglement and completeness of the vanilla VAE was remarkably low for the Cars3D dataset. We believe this result comes from a difference in statistics: the Cars3D dataset ($N = 17,568$) has fewer samples than the SmallNORB dataset ($N = 48,600$) and has a more complicated structure, including colors. The DCI scores reached their maxima at $\alpha = 1 \times 10^{-2}$ on the SmallNORB dataset and at $\alpha = 1$ on the Cars3D dataset. These results indicate that the locality enables the proposed model to achieve a disentangled representation for each subspace without any label information. The FID also reached its minima at $\alpha = 1 \times 10^{-2}$ on the SmallNORB dataset and at $\alpha = 1 \times 10^{-1}$ on the Cars3D dataset. The optimal locality was suggested to be different for each subdataset because both the DCI scores and FID took their best values at different α. Overall, the local VAE achieved better performances than the vanilla VAE in terms of the disentanglement and generation quality.

Table 5.7.3: Quantitative evaluations on the CarsNORB dataset.

		NORB			Cars		
		Disent.	Compl.	FID	Disent.	Compl.	FID
Local VAE	$\alpha = 0$	0.254	0.257	143.336	0.064	0.060	196.501
	$\alpha = 1 \times 10^{-2}$	**0.315**	0.329	**136.003**	0.119	0.117	197.093
	$\alpha = 1 \times 10^{-1}$	0.189	0.165	155.469	0.125	0.105	**194.962**
	$\alpha = 1$	0.254	0.257	164.927	**0.254**	0.144	202.778
β-VAE	$\beta = 2$	0.306	0.321	137.821	0.123	0.097	196.828
	$\beta = 4$	0.305	0.334	141.702	0.204	0.150	202.577
	$\beta = 8$	0.289	**0.338**	150.847	0.162	**0.167**	205.811
	$\beta = 16$	0.271	0.334	170.169	0.179	**0.167**	217.428

5.7.3 *CelebA Dataset*

Last, we show the qualitative evaluation on the CelebFaces Attributes (CelebA) dataset (Liu et al., 2015). The CelebA dataset is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. We used the k-neighbor method as a neighborhood construction, and the network structure was identical to the previous numerical experiments. However, since the number of attributes is larger than the aforementioned dataset, we set the dimension of latent variables to 100 dimensions.

Figure 5.7.9 shows the average of reference samples and the corresponding generated images for the CelebA dataset. The leftmost column shows the average of k reference training samples. We see that the colors of skin and hair are preserved in the same row and other shape-related information remains in the same column. It means that the local VAE spontaneously extracted color information as the global feature. This result is qualitatively consistent to the results of the 3D Shapes dataset and the CarsNORB dataset. It is expected that we can obtain more complex shape information as global feature by changing the neighborhood construction method or by taking multiple steps of MAML optimization.

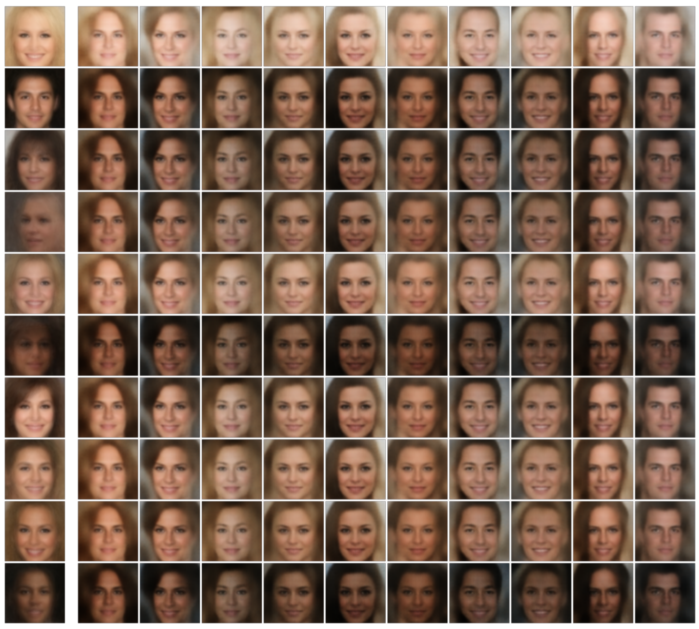


Figure 5.7.9: Average of reference training samples and the corresponding conditionally generated images of the local VAE with α =1. The leftmost column shows the average of k reference training samples. Each row visualizes the generated images conditioned on the reference sample, and the images shown in the same column share their latent code. The trained model extracted color information as global features and other information as the local features.

5.8 CONCLUSION

In this study, we proposed the local VAE, a deep generative model suitable for datasets with locally disentangled structures. Disentanglement is known to be fundamentally impossible without inductive biases both in models and in datasets, although such a bias was less considered in previous studies. We considered the locality of disentanglement factors as the explicit form of an inductive bias for the dataset. We used model-agnostic meta-learning by considering each local subset as a task and proposed the local VAE accordingly. We evaluated our proposed model with the 3D Shapes dataset, a concatenated dataset of Cars3D and SmallNORB, and the CelebA dataset. Our experimental results showed that the local VAE spontaneously extracted color-related information as the global structure on the 3D Shapes dataset. The learned representations of the local VAE were more disentangled than that of comparable models in terms of the DCI scores for both datasets. Moreover, the local VAE improved the quality of the generated images according to subjective evaluations and FID scores.

CHAPTER APPENDIX

5.A DEFINITION OF DISENTANGLEMENT BY USING GROUP ACTION

In this section, we will briefly summarize the definition of disentangled representation proposed by Higgins et al. (2018). The motivation for studying a disentangled representation is to extract independent parameters that uniquely control the properties of the dataset in each dimension of the latent space. However, disentanglement has been studied without a consensus on what the disentangled representation means. In this context, Higgins et al. (2018) presented a formal mathematical definition of disentanglement. Specifically, Higgins et al. (2018) define the disentanglement in the sense of a group action on a set. We will explain the definition of the disentanglement group action in Section 5.A.1. Then, we will introduce the definition of the disentanglement representation in Section 5.A.2.

5.A.1 *Disentanglement group action*

Here, we consider a group action $\cdot : A \times X \to X$. Suppose that group A is decomposable as a direct product $A = A_1 \times A_2$. We refer to the actions of a group and a subgroup of the group as $\cdot$ and $\cdot_i$, respectively.

Definition 5.A.1. *An action is disentangled if there is a decomposition $X = X_1 \times X_2$ such that*

$$(a_1, a_2) \cdot (x_1, x_2) = (a_1 \cdot_1 x_1, a_2 \cdot_2 x_2). \tag{5.1}$$

This definition states that an element of A_1 acts only on X_1 and not on X_2. Additionally, if X is a vector space, we are interested in the direct sum decomposition:

$$X = X_1 \oplus X_2, \tag{5.2}$$

which preserves the structure of the space. We also generalize the above definition to d subspaces.

5.A.2 *Disentanglement representation*

Let W be the set of world-states and X be the observation. We also assume a generative process $b : W \to X$, and an inference process $g : X \to Z$ where the internal representation Z is a vector space. We refer to the composition of these processes as $h = g \circ b$. Here, we will consider a group A acting on X, that is $\cdot : A \times W \to W$. The goal is to find a corresponding group action $\cdot : G \times Z \to Z$. The group A on Z should reflect the symmetry of world-states W. In other worlds, A should satisfy following properties:

$$a \cdot h(w) = h(a \cdot w) \ \forall a \in A, w \in W. \tag{5.3}$$

The above equation states that the action should commute with map h. This equation is as same as the following commutative diagram:

$$\begin{tikzcd} A \times W \arrow[r, "\cdot W"] \arrow[d, "\mathrm{id}_A \times h"] & W \arrow[d, "h"] \\ A \times Z \arrow[r, dashed, "\cdot Z"] & Z \end{tikzcd} \tag{5.4}$$

Note that there is no guarantee that we can construct an action $\cdot : A \times Z \to Z$ satisfying Equation 5.3. If h is bijective, there exists a unique group action that satisfies Equation 5.3 by definition. If h is not surjective but injective, Equation 5.3 cannot determine a group action on Z that is not the image of $h(W)$. However, this is not particularly important because we are interested in only Z of the image of $h(W)$. Finally, if h is not injective, or there exists $w_1 \neq w_2$ such that $h(w_1) = h(w_2)$, the representation in Z is degenerate. For example, if an agent cannot observe some control parameters because of occlusion, the difference in these parameters will be undistinguishable. However, we can solve this problem by using active sensing in practical situations.

Suppose that there exists an action $\cdot : G \times Z \to Z$ and an equivalent map $h : W \to Z$. We define the disentanglement representation as follows:

Definition 5.A.2. *An agent's representation Z is disentangled with respect to decomposition $A = A_1 \times \cdots \times A_d$ if*

1. *There is an action $\cdot : A \times Z \to Z$.*
2. *The map $h : W \to Z$ is equivariant between the actions on W and Z, and*
3. *There is a decomposition $Z = Z_1 \times \cdots \times Z_d$ or $Z = Z_1 \oplus \cdots \oplus Z_d$ such that each Z_i is unaffected by all actions $A_j, j \neq i$ and affected only by A_i.*

5.B ANALYZING LOCAL VAES FROM THE VIEWPOINT OF BASIS DECOMPOSITION

In this section, we first describe an interpretation of the computational graph of the local VAE as a linear combination of fixed orthonormal and input-dependent bases. Then, we show that an inference model of this type can express any locally disentangled representation under certain conditions.

Suppose that the VAE's encoder is $g : \mathcal{X} \times \Phi \to \mathcal{Z}$ and its decoder is $f : \mathcal{Z} \times \Theta \to \mathcal{X}$. We define objective function $\mathcal{L}_{\mathcal{D}} : \Theta \times \Phi \to \mathbb{R}$ as follows:

$$\mathcal{L}_{\mathcal{D}}(\theta, \phi) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ell(x; \theta, \phi), \tag{5.5}$$

where $\mathcal{D}$ represents the dataset and ℓ is the objective function for each data point. In a VAE, ℓ corresponds to Equation 5.1 in Section 5.2.1.

We define the training procedure of a VAE with MAML as follows:

$$\theta_\tau = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}_\tau}(\theta, \phi), \tag{5.6}$$

$$\phi_\tau = \phi - \alpha \nabla_\phi \mathcal{L}_{\mathcal{D}_\tau}(\theta, \phi), \tag{5.7}$$

$$\theta \leftarrow \theta - \eta \nabla_\theta \frac{1}{|T|} \sum_{\tau \in T} \mathcal{L}_{\mathcal{D}_\tau}(\theta_\tau, \phi_\tau), \tag{5.8}$$

$$\phi \leftarrow \phi - \eta \nabla_\phi \frac{1}{|T|} \sum_{\tau \in T} \mathcal{L}_{\mathcal{D}_\tau}(\theta_\tau, \phi_\tau). \tag{5.9}$$

Note that $\tau \in T = \{1, 2, 3, \ldots\}$ indexes the tasks and that $\mathcal{D}_\tau \subseteq \mathcal{D}$ is a set composed of data points included in task τ. Variables α and η are learning rates of the inner loop and the outer loop, respectively. MAML is an optimization algorithm consisting of an inner loop (Equation 5.6 and Equation 5.7) and an outer loop (Equation 5.8 and Equation 5.9). In this study, we present an another view of this algorithm as a computational graph. We interpret the inner loop (Equation 5.6 and Equation 5.7) as part of the inference model rather than an optimization.

Next, we discuss the dependency of encoder g on task τ. With the encoder's parameter denoted by ϕ_τ, the latent vectors are written as follows:

$$\begin{aligned} z_\tau &= g(x; \phi_\tau) \\ &= g(x; \phi - \alpha \nabla_\phi \mathcal{L}_{\mathcal{D}_\tau}(\theta, \phi)). \end{aligned} \tag{5.10}$$

Taking the Taylor series expansion of z_τ for the first-order term of α yields

$$\begin{aligned} z_\tau &\simeq g(x; \phi_0) - \alpha \langle \nabla_\phi g(x; \phi_0), \nabla_\phi \mathcal{L}_{\mathcal{D}_\tau}(\theta_0, \phi_0) \rangle + O(\alpha^2) \\ &= g(x; \phi_0) - \alpha \nabla_\phi g(x; \phi_0) \frac{1}{|\mathcal{D}_\tau|} \sum_{x' \in \mathcal{D}_\tau} \nabla_\phi g(x'; \phi_0)^\top \nabla_{z'} \ell^\top + O(\alpha^2). \end{aligned} \tag{5.11}$$

Note that we denote the actual values of θ and ϕ by θ_0 and ϕ_0 to avoid the confusion with variables to be differentiated. $\nabla_{z'} \ell$ is the gradient of ℓ with respect to the encoder's output. According to the equation, an encoded latent vector decomposes into the first term, which is invariant to tasks, and the second term, which depends on the task.

We also study the task dependency of decoder f. As done above, a map of data with task-dependent parameters θ_τ, ϕ_τ can be decomposed as follows:

$$\begin{aligned}
f(z_\tau;\theta_\tau) =& f\left(g(\boldsymbol{x};\phi_0) - \alpha \nabla_\phi g(\boldsymbol{x};\phi_0) \frac{1}{|\mathcal{D}_\tau|} \sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \nabla_\phi g(\boldsymbol{x}';\phi_0)^\top \nabla_{z'} \ell^\top ;\, \theta_0 - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}_\tau}(\theta_0,\phi_0) \right) \\
\simeq& \underbrace{f(g(\boldsymbol{x};\phi_0);\theta_0)}_{\text{invariant VAE}} - \frac{\alpha}{2} \Bigg\{ \underbrace{\left\langle \frac{1}{|\mathcal{D}_\tau|} \sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \nabla_{z'} \ell \nabla_\phi g(\boldsymbol{x}';\phi_0) \nabla_\phi g(\boldsymbol{x};\phi_0)^\top, \nabla_z f(g(\boldsymbol{x};\phi_0);\theta_0) \right\rangle}_{\tau-\text{dependent encoder}} \\
&+ \underbrace{\left\langle \frac{1}{|\mathcal{D}_\tau|} \sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \nabla_{\hat{x}'} \ell \nabla_\theta f(z';\theta_0), \nabla_\theta f(g(\boldsymbol{x};\phi_0);\theta_0) \right\rangle}_{\tau-\text{dependent decoder}} \Bigg\} + \mathcal{O}(\alpha^2), \qquad (5.12)
\end{aligned}$$

where $\nabla_{\hat{x}'} \ell$ is the gradient of ℓ with respect to the decoder's output. We refer to the task-invariant term as an invariant VAE, to the task-dependent term related to the derivative of the encoder parameter ϕ as the τ-dependent encoder, and to the task-dependent term related to the decoder parameter θ as the τ-dependent decoder. This computational graph corresponds to a vanilla VAE in the limit of $\alpha \to 0$.

Next, we explicitly write the output of the encoder and decoder as a linear combinations of bases. Ignoring the second-order term $\mathcal{O}(\alpha^2)$ of Equation 5.11, we can rewrite the encoder as follows:

$$\begin{aligned}
z_\tau =& g(\boldsymbol{x};\phi_0) - \alpha \nabla_\phi g(\boldsymbol{x};\phi_0) \frac{1}{|\mathcal{D}_\tau|} \sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \nabla_\phi g(\boldsymbol{x}';\phi_0)^\top \nabla_{z'} \ell^\top \\
=& \sum_{i=1}^{d} g_i(\boldsymbol{x};\phi_0) \boldsymbol{e}^i + \alpha \sum_{i=1}^{|\Phi|} p^i_{\mathcal{D}_\tau} \frac{\partial}{\partial \phi_i} g(\boldsymbol{x};\phi_0), \qquad (5.13)
\end{aligned}$$

where $\boldsymbol{p}_{\mathcal{D}_\tau} = -\frac{1}{|\mathcal{D}_\tau|} \sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \nabla_\phi g(\boldsymbol{x}';\phi_0)^\top \nabla_{z'} \ell^\top$ only depends on $\mathcal{D}_\tau$ but on $\boldsymbol{x}$. d is the dimension of the latent space and $|\Phi|$ is the number of encoder parameters. The first-order term is explicitly expressed as a linear combination of orthonormal bases $[\boldsymbol{e}^1, \dots, \boldsymbol{e}^d]$. As in the case of the encoder, the output of the decoder becomes

$$\begin{aligned}
\hat{x}_\tau =& f(g(\boldsymbol{x};\phi_0);\theta_0) - \frac{\alpha}{2} \Bigg\{ \left\langle \frac{1}{|\mathcal{D}_\tau|} \sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \nabla_{z'} \ell \nabla_\phi g(\boldsymbol{x}';\phi_0) \nabla_\phi g(\boldsymbol{x};\phi_0)^\top, \nabla_z f(g(\boldsymbol{x};\phi_0);\theta_0) \right\rangle \\
&+ \left\langle \frac{1}{|\mathcal{D}_\tau|} \sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \nabla_{\hat{x}'} \ell \nabla_\theta f(z';\theta_0), \nabla_\theta f(g(\boldsymbol{x};\phi_0);\theta_0) \right\rangle \Bigg\} \\
=& \sum_{i=1}^{|X|} f(g(\boldsymbol{x};\phi_0);\theta_0)_i \boldsymbol{e}^i + \frac{\alpha}{2} \left\{ \sum_{i=1}^{|\Phi|} p^i_{\mathcal{D}_\tau} \nabla_z f(z;\theta_0) \frac{\partial}{\partial \phi_i} g(\boldsymbol{x};\phi_0) + \sum_{i=1}^{|\Theta|} q^i_{\mathcal{D}_\tau} \frac{\partial}{\partial \theta_i} f(z;\theta_0) \right\}, \qquad (5.14)
\end{aligned}$$

where $\boldsymbol{q}_{\mathcal{D}_\tau} = -\frac{1}{|\mathcal{D}_\tau|}\sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \nabla_{\hat{\boldsymbol{x}}'} \ell \nabla_\theta f(\boldsymbol{z}'; \theta_0)$.

As we can see, the first-order term of Equation 5.13 is a linear combination of coefficients dependent on input $\boldsymbol{x}$ and a fixed basis. On the other hand, the coefficients of the second term $p^i_{\mathcal{D}_\tau}$ do not depend on input $\boldsymbol{x}$ during the inference phase but exclusively depend on task dataset $\mathcal{D}_\tau$. The basis $\partial g(\boldsymbol{x}; \phi_0)/\partial \phi_i$ rather than the coefficient depends on the input.

Then, we show the capability of the local VAE's encoder (Equation 5.9). The formal statement of Theorem 5.4.1 in Section 5.4 is as follows:

Theorem 5.B.1. *The encoder of the local VAE can express any locally disentangled representation with an error of up to $\mathcal{O}(\alpha^2)$ if*

1. *There is an action $\cdot : A \times Z \to Z$ that satisfies Definition 5.4.1 in Section 5.4,*
2. *The encoder g, its derivative with respect to the parameter $\partial g(\cdot; \phi_0)/\partial \phi_i$, and the Jacobian of the decoder $\partial f(\boldsymbol{z})/\partial \boldsymbol{z}$ are arbitrary complex functions, and*
3. *We can take task-specific dataset such as*

$$\mathcal{D}_\tau(\boldsymbol{x}) \subseteq \left\{\boldsymbol{x}' | \boldsymbol{x}' = b(\boldsymbol{w}'), \forall \boldsymbol{w}' \in W \text{ and } \boldsymbol{w}'^{\ell} = \boldsymbol{w}^{\ell}\right\}. \tag{5.15}$$

Although the first condition is included in the definition of disentangled representation, we do not consider this condition because there is no guarantee that such a group action exists in general cases. Hence, we assume the existence of a group action.

The definition proposed by Higgins does not necessarily require control parameters to decompose into an orthonormal basis in the latent space. For instance, one can arbitrarily take basis vectors as a direct sum decomposition of a vector space. Given some conditions, we will show that a local VAE can express locally disentangled representations defined in Theorem 5.4.1.

Proof. By definition, there exists a decomposition $W = W_1 \times \cdots \times W_r \times W^{\ell}$ and we can consider an isomorphism from W^{ℓ} to $W_{r+1} \times \cdots \times W_d$ depending on $\boldsymbol{w}_r \in W_1 \times \cdots \times W_r$. Additionally, the first condition ensures the existence of $\cdot : A \times Z \to Z$ satisfying Equation 5.7 in Section 5.4 such that Z is locally decomposable. Since we assume Z to be a vector space, the latent space Z has a direct sum decomposition $Z = Z_1 \oplus \cdots \oplus Z_r \times Z_{r+1} \oplus \cdots \oplus Z_d$ for each $\boldsymbol{z}_r$, and the set of the entire space is

$$Z = \left\{ \boldsymbol{z} \,\middle|\, \boldsymbol{z} = \sum_{i=1}^{r} z_i \boldsymbol{e}^i + \sum_{i=r+1}^{d} z_i \boldsymbol{e}^i(\boldsymbol{z}_r) \right\}, \tag{5.16}$$

where $z_i \in \mathbb{R}$, and $\boldsymbol{e}^i$ is a basis vector. Notation $\boldsymbol{e}^i(\boldsymbol{z}_r)$ represents a basis vector that depends on the first r dimensions of $\boldsymbol{z}$. Whole basis vectors $[\boldsymbol{e}^1, \cdots, \boldsymbol{e}^r, \boldsymbol{e}^{r+1}(\boldsymbol{z}_r), \cdots, \boldsymbol{e}^d(\boldsymbol{z}_r)]$ are linearly independent.

According to the encoder of the local VAE (Equation 5.13), if $\partial g(\cdot; \phi_0)/\partial \phi_i : \mathcal{X} \to \mathcal{Z}$ is complex enough (the second condition), it can express an arbitrary basis dependent on input $\boldsymbol{x}$ and the underlying $\boldsymbol{w}_r$. We can take up to d linearly independent bases. Additionally, if $\nabla_z f(\cdot;\theta_0) : \mathcal{Z} \to \mathcal{X}$ is complex enough, the term

$$p^i_{\mathcal{D}_\tau} = -\frac{1}{|\mathcal{D}_\tau|} \sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \frac{\partial}{\partial \phi_i} g(\boldsymbol{x}';\phi_0)^\top \nabla_{z'} \ell^\top = -\frac{1}{|\mathcal{D}_\tau|} \sum_{\boldsymbol{x}' \in \mathcal{D}_\tau} \frac{\partial}{\partial \phi_i} g(\boldsymbol{x}';\phi_0)^\top \nabla_{\boldsymbol{x}'} \ell^\top \nabla_{z'} f(z';\theta_0) \quad (5.17)$$

can be an arbitrarily complex function with respect to task dataset $\mathcal{D}_\tau$. By assuming that we can take a task dataset satisfying the third condition of Theorem 5.4.1, we can choose the ground truth of z_i as $p^i_{\mathcal{D}_\tau(\boldsymbol{x})}$. At last, we take $g_i(\boldsymbol{x};\phi_0) = 0$ for all $\boldsymbol{x}, i \in \{r+1,\dots,d\}$ and $p^i_{\mathcal{D}_\tau} = 0$ for all $\mathcal{D}_\tau, i \in \{d-r+1,\dots,|\Phi|\}$ to choose d linearly independent bases. Based on this construction, the space that the encoder can express corresponds to Equation 5.16. $\square$

As we can see, the encoder/decoder of the local VAE can be interpreted as a linear combination of fixed and input-dependent bases, whereas a vanilla VAE only has fixed orthonormal bases. As a result, a vanilla VAE and its variants (including β-VAE and most of the disentanglement models) do not seem to be well suited for datasets with the local disentanglement assumption even if they can achieve a zero training loss. In the second condition of the theorem, we assumed that the encoder/decoder and their derivatives to be arbitrary complex functions. Such an assumption is reasonable for neural networks because they are typically highly nonlinear and over-parameterized.

5.C EXPERIMENTAL CONDITIONS AND HYPERPARAMETERS

In this section, we show the experimental conditions and hyperparameters which are used for all the numerical experiments in Section 5.7. Table 5.C.1 shows the encoder and the decoder architectures of the VAE. We used the multivariate isotropic Gaussian for the latent variable. The outputs of the encoder correspond to $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma}$ of the variational poseterior distribution $q(\boldsymbol{z}|\boldsymbol{x})$. Table 5.C.2 shows the hyperparameters for the model and the training procedure. In addition to the parameters shown in the table, we used the gradient boosted trees from Scikit-learn with the default setting for computing the DCI scores. We also used the Inception-v3 network from Keras, which is pre-trained on the ImageNet dataset to compute the FID.

Table 5.C.1: Network architecture for the numerical experiments.

Encoder	Decoder
Input: $64 \times 64 \times 3$	Input: $\mathbb{R}^{10}$
4×4 conv, 32 ReLU, stride 2	FC, 256 ReLU
4×4 conv, 32 ReLU, stride 2	FC, $4 \times 4 \times 64$ ReLU
4×4 conv, 64 ReLU, stride 2	4×4 upconv, 64 ReLU, stride 2
4×4 conv, 64 ReLU, stride 2	4×4 upconv, 32 ReLU, stride 2
FC 256, F2 2×10	4×4 upconv, 32 ReLU, stride 2
	4×4 upconv, 3, stride 2

Table 5.C.2: The hyperparameters which are used for the numerical experiments.

Parameter	Value
Batch size (corresponds to the number of tasks)	25
Inner batch size (corresponds to K)	10
Latent space dimension	10
Optimizer	Adam
Adam: beta1	0.9
Adam: beta2	0.999
Adam: epsilon	1×10^{-8}
Adam: learning rate	1×10^{-4}
Decoder type	Bernoulli
Training steps	300,000

6 CONCLUSION AND FUTURE DIRECTION

In this thesis, we studied the framework for capturing the structure between instances of a dataset using VAE as a central technique.

In Chapter 3, we numerically analyzed the dynamics of the inference of the deep generative model for a dataset with a cluster structure. When we performed repeated inference with the VAE trained on MNIST, the activity pattern in the latent space rapidly approaches the center of the cluster of training data. We also found that the dynamics on the latent space changed depending on the amount of noise added to the input. These results suggest that the dynamics of repeated inference reflect the structure behind the dataset. The effect of repeated inference, which has been reported so far, is thought to be due to this approach to the cluster center. It is suggested that the abstraction of the output can be manipulated by the number of times the inference is repeated, even in the same model.

In Chapter 4, we proposed a practical probability distribution for datasets with hierarchical structures and a representation learning method using it. By constructing a projection-based probability distribution on hyperbolic space, we proposed a probability distribution whose likelihoods can be expressed in a closed-form and can be easily sampled and differentiated. To validate the proposed probability distributions, we performed numerical experiments with word embedding tasks and training VAEs for natural image datasets. We used the MNIST dataset and the Atari 2600 Breakout observation series as natural images and the WordNet dataset for word embedding. In particular, the observation series of Breakout is thought to implicitly have a tree structure with a root node as the starting state. In the latent space of hyperbolic VAEs trained in Breakout, the norm on hyperbolic space correlated with the progression of the game in Breakout.

In Chapter 5, we proposed a method to train a deep generative model that is useful for datasets with local structures. We have focused on cases where we can explicitly give a structure in above two studies. In this chapter, we discussed a representation learning under the broader assumption. For a disentanglement that aims to extract a small number of control factors from high-dimensional observations, we defined a local disentanglement, assuming different scales for each factor. We showed that representation learning is possible in meta-learning aimed at fast adaptation to new tasks by considering the generation of neighboring data points in the data space as a single task. We proposed a local VAE using a hierarchical

Bayesian interpretation of MAML as a generative model with task-specific parameters for each local neighborhood of the dataset. We showed that local VAE improves the quality of both the generated images and latent representations in experiments with 3D Shapes, Cars3D, and SmallNORB.

In this thesis, we studied the inference and learning of deep generative models for several instance-wise structures using neural networks. The advantage of neural networks is that any computational graph can be used if it is differentiable. Since we can treat the model independently of the optimization method, we can straightforwardly model the problem. All of our proposed inference and learning methods can be interpreted as computational graphs. We can consider the repeated inference, which is the subject of analysis in Chapter 3, as a deep network of modules in series that share weights. The discovery of cluster structures with longer computational graphs is consistent with the finding that complex features are extracted at higher layers of a deep neural network (Bengio et al., 2013; Lee et al., 2009). Moreover, the probability distribution on hyperbolic space proposed in Chapter 4 is indeed a differentiable computational graph, and the learning algorithm proposed in Chapter 5 can be interpreted as a computational graph as described in Section 5.4. Restricting task-specific parameters to a range that can be reached in a few steps from global parameters is a strong inductive bias, resulting in a disentanglement. Such an approach to modeling a dataset as a computational graph is widely applicable to other subjects than those covered in this thesis.

Current deep learning techniques have outperformed humans in a variety of tasks under massive datasets and large computational resources. On the other hand, in terms of adaptation to environmental changes and generalization from a few evidence, the algorithms of deep learning have room for improvement. In addition to engineering applications, we need to address these issues from a scientific perspective of revealing intelligence. In particular, we believe that the structure of data is vital because it is universal knowledge that we use implicitly in our daily decision-making.

We have focused on cluster, hierarchical and local structures as typical structures behind a dataset. However, the structure of the real world is not limited to such a simple case. In particular, many machine learning algorithms, including the methods proposed in this thesis, are assumed to have access to all data, or a uniform random sample from entire dataset. On the other hand, in the real world, there is no guarantee that we can uniformly sample data points against the background structure. It will be important to learn the structure in such a situation where only non-stationary samples can be obtained for the background structure. Recently, Arjovsky et al. (2019) proposed a framework to minimize the spurious correlation inherent in training data with a few assumptions. Moreover, Khemakhem, Kingma, and Hyvärinen (2019) showed that the disentangled representation of VAE becomes identifiable for samples from non-stationary environments by using the label of the environment. Both of these studies imply that when the environment is non-stationary, the active use of that non-stationarity instead increases information. Therefore, investigating a policy for acquiring data for representational learning in an interactive environment is also important. In an interactive environment, such as the one targeted by reinforcement learning, the observations themselves change as a result of one's actions. It would be useful to consider policies that focus on learning the representation of the environment rather than maximizing the reward. Recently, the generative model learning of interactive environments (Ha and Schmidhuber, 2018) and the interpretation of reinforcement learning as probabilistic inference (Levine, 2018) have been studied. Based on these studies, it will be possible to study policies for representation learning in an interactive environment.

Another direction could be to incorporate the instance-wise structure as a layer architecture of the network rather than a generative model. For example, batch normalization (Ioffe and Szegedy, 2015), a regularization method widely used in deep learning, computes statistics using other samples in the same batch. For datasets with graph structures, graph neural networks (Wu et al., 2020) take the approach of giving each node a vector representation and learning the interaction through the edges. Although these methods are proposed for completely different purposes, they can be considered as models that incorporate some kind of structure between instances. The improvement of computational graphs based on the objective of reflecting the structure between instances assumed in the dataset could be one way forward to consider suitable computational graphs.

We believe that one goal of intelligence research is to answer the question of how and why humans can think. For HOW in this question, it is necessary to investigate the operating mechanism of a specific intelligent system. Model proposals and theoretical analysis, which are widely conducted in machine learning research, tackle exactly this question. In contrast, for WHY, we believe that it is essential to consider the external world intensively. We cannot explain the reason why the information processing of organisms is in its present form without the constraints of the external world to which the system is exposed. Based on these motivations, we discussed the relationship between the structure of the external world and information processing systems. Yet, the structures dealt with in this thesis remain simple and static. To obtain better insight into the relationship between information processing systems and the outside world, we believe that we need more sophisticated assumptions, such as an interaction between agent and environment. It will be essential to study and consider the external world itself *from the perspective of representation learning*. We expect that this thesis will be a part of such future intelligence research.

BIBLIOGRAPHY

Achille, Alessandro, Tom Eccles, Loic Matthey, Chris Burgess, Nicholas Watters, Alexander Lerchner, and Irina Higgins (2018). "Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies." In: *Advances in Neural Information Processing Systems 31*, pp. 9873–9883 (cit. on p. 69).

Amari, Shunichi (1977). "Neural theory of association and concept-formation." In: *Biological Cybernetics* 26.3, pp. 175–185 (cit. on pp. 24, 31, 33).

Amit, Daniel J, Hanoch Gutfreund, and Haim Sompolinsky (1985). "Spin-glass models of neural networks." In: *Physical Review A* 32.2, p. 1007 (cit. on p. 33).

Andrychowicz, Marcin, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas (2016). "Learning to learn by gradient descent by gradient descent." In: *Advances in neural information processing systems*, pp. 3981–3989 (cit. on p. 71).

Arjovsky, Martin, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz (2019). "Invariant risk minimization." In: *arXiv preprint arXiv:1907.02893* (cit. on pp. 4, 104).

Arora, Sanjeev, Rong Ge, Behnam Neyshabur, and Yi Zhang (2018). "Stronger Generalization Bounds for Deep Nets via a Compression Approach." In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, pp. 254–263 (cit. on p. 2).

Arulkumaran, Kai, Antonia Creswell, and Anil Anthony Bharath (2016). "Improving Sampling from Generative Autoencoders with Markov Chains." In: *arXiv preprint arXiv:1610.09296* (cit. on pp. 16, 34).

Behrmann, Jens, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Joern-Henrik Jacobsen (2019). "Invertible Residual Networks." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 573–582 (cit. on p. 14).

Bengio, Samy, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei (1992). "On the optimization of a synaptic learning rule." In: *Preprints Conf. Optimality in Artificial and Biological Neural Networks*. Univ. of Texas, pp. 6–8 (cit. on p. 71).

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013). “Representation learning: A review and new perspectives.” In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1798–1828 (cit. on pp. 3, 4).

Bengio, Yoshua, Grégoire Mesnil, Yann Dauphin, and Salah Rifai (2013). “Better mixing via deep representations.” In: *Proceedings of the 30th International Conference on Machine Learning*, pp. 552–560 (cit. on pp. 34, 104).

Bouchacourt, Diane, Ryota Tomioka, and Sebastian Nowozin (2018). “Multi-Level Variational Autoencoder: Learning Disentangled Representations From Grouped Observations.” In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 2095–2102 (cit. on p. 78).

Brand, Matthew (2003). “Charting a manifold.” In: *Advances in neural information processing systems*, pp. 985–992 (cit. on pp. 5, 79).

Brock, Andrew, Jeff Donahue, and Karen Simonyan (2019). “Large Scale GAN Training for High Fidelity Natural Image Synthesis.” In: *International Conference on Learning Representations* (cit. on p. 10).

Burda, Yuri, Roger B. Grosse, and Ruslan Salakhutdinov (2016). “Importance Weighted Autoencoders.” In: *Proceedings of the 4th International Conference on Learning Representations* (cit. on p. 56).

Burgess, Chris and Hyunjik Kim (2018). *3D Shapes Dataset*. `https://github.com/deepmind/3dshapes-dataset/` (cit. on pp. 71, 77).

Burgess, Christopher P, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner (2017). “Understanding disentangling in β-VAE.” In: *Workshop on Learning Disentangled Representations at the 31st Conference on Neural Information Processing Systems* (cit. on pp. 70, 78, 85).

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille (2014). “Semantic image segmentation with deep convolutional nets and fully connected crfs.” In: *arXiv preprint arXiv:1412.7062* (cit. on p. 5).

– (2017). “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.” In: *IEEE transactions on pattern analysis and machine intelligence* 40.4, pp. 834–848 (cit. on p. 5).

Chen, Tian Qi, Xuechen Li, Roger B Grosse, and David K Duvenaud (2018). “Isolating sources of disentanglement in variational autoencoders.” In: *Advances in Neural Information Processing Systems*, pp. 2610–2620 (cit. on pp. 70, 78, 86).

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). “Learning phrase representations using

RNN encoder-decoder for statistical machine translation." In: *arXiv preprint arXiv:1406.1078* (cit. on p. 5).

Davidson, Tim R., Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak (2018). "Hyperspherical Variational Auto-Encoders." In: *34th Conference on Uncertainty in Artificial Intelligence* (cit. on pp. 54, 79).

Dayan, Peter, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel (1995). "The helmholtz machine." In: *Neural computation* 7.5, pp. 889–904 (cit. on pp. 1, 10).

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2017). "Density estimation using real nvp." In: *Proceedings of the 5th International Conference on Learning Representations* (cit. on pp. 11, 13, 14).

Eastwood, Cian and Christopher K. I. Williams (2018). "A framework for the quantitative evaluation of disentangled representations." In: *International Conference on Learning Representations* (cit. on pp. 81, 84).

Ekman, Paul and Dacher Keltner (1997). "Universal facial expressions of emotion." In: *Segerstrale U, P. Molnar P, eds. Nonverbal communication: Where nature meets culture*, pp. 27–46 (cit. on p. 71).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). "Model-agnostic Meta-learning for Fast Adaptation of Deep Networks." In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR.org, pp. 1126–1135 (cit. on pp. 71, 72, 79).

Fukushima, Kunihiko and Sei Miyake (1982). "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position." In: *Pattern recognition* 15.6, pp. 455–469 (cit. on pp. 4, 5).

Ganea, Octavian-Eugen, Gary Bécigneul, and Thomas Hofmann (2018a). "Hyperbolic Entailment Cones for Learning Hierarchical Embeddings." In: *Proceedings of the 35th International Conference on Machine Learning*, pp. 1632–1641 (cit. on pp. 46, 54).

– (2018b). "Hyperbolic neural networks." In: *Advances in Neural Information Processing Systems 31*, pp. 5345–5355 (cit. on pp. 46, 54).

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets." In: *Advances in Neural Information Processing Systems 27*, pp. 2672–2680 (cit. on pp. 10, 13, 15, 69, 89).

Grant, Erin, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths (2018). "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes." In: *International Conference on Learning Representations* (cit. on pp. 71, 72, 79).

Grattarola, Daniele, Lorenzo Livi, and Cesare Alippi (2018). "Adversarial Autoencoders with Constant-Curvature Latent Manifolds." In: *CoRR* abs/1812.04314 (cit. on pp. 34, 54).

Graves, Alex (2013). "Generating sequences with recurrent neural networks." In: *arXiv preprint arXiv:1308.0850* (cit. on p. 5).

Gretton, Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola (Mar. 2012). "A Kernel Two-sample Test." In: *The Journal of Machine Learning Research* 13, pp. 723–773 (cit. on p. 54).

Gülçehre, Çaglar et al. (2019). "Hyperbolic Attention Networks." In: *Proceedings of the 7th International Conference on Learning Representations* (cit. on pp. 43, 54).

Ha, David and Jürgen Schmidhuber (2018). "World Models." In: *CoRR* abs/1803.10122. arXiv: 1803.10122 (cit. on p. 105).

Hebb, Donald Olding (1949). *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall (cit. on p. 1).

Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter (2017). "Gans trained by a two time-scale update rule converge to a local nash equilibrium." In: *Advances in Neural Information Processing Systems*, pp. 6626–6637 (cit. on pp. 14, 81, 84).

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner (2017). "β-VAE: Learning basic visual concepts with a constrained variational framework." In: *Proceedings of the 6th International Conference on Learning Representations* (cit. on pp. 34, 52, 69, 70, 78, 84).

Higgins, Irina, David Amos, David Pfau, Sébastien Racanière, Loïc Matthey, Danilo J. Rezende, and Alexander Lerchner (2018). "Towards a Definition of Disentangled Representations." In: *CoRR* abs/1812.02230. eprint: 1812.02230 (cit. on pp. 70, 71, 75, 78, 95).

Hinton, Geoffrey E (2002). "Training products of experts by minimizing contrastive divergence." In: *Neural computation* 14.8, pp. 1771–1800 (cit. on p. 10).

Hoogeboom, Emiel, Rianne Van Den Berg, and Max Welling (2019). "Emerging Convolutions for Generative Normalizing Flows." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 2771–2780 (cit. on p. 14).

Hopfield, John J (1982). "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the National Academy of Sciences* 79.8, pp. 2554–2558 (cit. on p. 33).

Hsu, Kyle, Sergey Levine, and Chelsea Finn (2019). "Unsupervised Learning via Meta-Learning." In: *International Conference on Learning Representations* (cit. on p. 79).

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 448–456 (cit. on pp. 68, 105).

Jacot, Arthur, Franck Gabriel, and Clément Hongler (2018). "Neural tangent kernel: Convergence and generalization in neural networks." In: *Advances in neural information processing systems*, pp. 8571–8580 (cit. on p. 2).

Jang, Eric, Shixiang Gu, and Ben Poole (2017). "Categorical Reparameterization with Gumbel-Softmax." In: *Proceedings of the 5th International Conference on Learning Representations* (cit. on pp. 54, 79).

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (2017). "Billion-scale similarity search with GPUs." In: *arXiv preprint arXiv:1702.08734* (cit. on p. 78).

Jonas, Eric and Konrad Paul Kording (2017). "Could a neuroscientist understand a microprocessor?" In: *PLoS computational biology* 13.1 (cit. on p. 3).

Kambhatla, Nandakishore and Todd K. Leen (1997). "Dimension Reduction by Local Principal Component Analysis." In: *Neural Computation* 9.7, pp. 1493–1516. DOI: 10.1162/neco.1997.9.7.1493 (cit. on p. 79).

Kemp, Charles and Joshua B. Tenenbaum (2008). "The discovery of structural form." In: *Proceedings of the National Academy of Sciences* 105.31, pp. 10687–10692. ISSN: 0027-8424. DOI: 10.1073/pnas.0802631105 (cit. on p. 69).

Khemakhem, Ilyes, Diederik P Kingma, and Aapo Hyvärinen (2019). "Variational autoencoders and nonlinear ica: A unifying framework." In: *arXiv preprint arXiv:1907.04809* (cit. on pp. 78, 104).

Kim, Hyunjik and Andriy Mnih (2018). "Disentangling by Factorising." In: *International Conference on Machine Learning*, pp. 2654–2663 (cit. on pp. 70, 78, 85).

Kingma, Diederik P (2017). "Variational inference & deep learning: A new synthesis. Intelligent Sensory Information Systems." PhD thesis. University of Amsterdam (cit. on p. 11).

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization." In: *Proceedings of the 3rd International Conference on Learning Representations* (cit. on p. 19).

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes." In: *Proceedings of the 2nd International Conference on Learning Representations* (cit. on pp. 10, 11, 15–17, 45, 52, 69, 70, 72).

Kingma, Diederik P, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling (2014). "Semi-supervised learning with deep generative models." In: *Advances in neural information processing systems*, pp. 3581–3589 (cit. on p. 78).

Kingma, Durk P and Prafulla Dhariwal (2018). "Glow: Generative flow with invertible 1x1 convolutions." In: *Advances in Neural Information Processing Systems*, pp. 10215–10224 (cit. on pp. 10, 14).

Kingma, Durk P, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling (2016). "Improved variational inference with inverse autoregressive flow." In: *Advances in neural information processing systems*, pp. 4743–4751 (cit. on pp. 14, 54).

Klys, Jack, Jake Snell, and Richard Zemel (2018). "Learning Latent Subspaces in Variational Autoencoders." In: *Advances in Neural Information Processing Systems 31*, pp. 6444–6454 (cit. on p. 78).

Kornblith, Simon, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton (2019). "Similarity of Neural Network Representations Revisited." In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR, pp. 3519–3529 (cit. on p. 3).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*, pp. 1097–1105 (cit. on p. 2).

Kumar, Abhishek, Prasanna Sattigeri, and Avinash Balakrishnan (2018). "Variational Inference of Disentangled Latent Concepts from Unlabeled Observations." In: *International Conference on Learning Representations* (cit. on pp. 70, 78, 85, 86).

Lake, Brenden M, Ruslan Salakhutdinov, and Joshua B Tenenbaum (2015). "Human-level concept learning through probabilistic program induction." In: *Science* 350.6266, pp. 1332–1338 (cit. on p. 69).

Le, Quoc V, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y Ng (2012). "Building high-level features using large scale unsupervised learning." In: *Proceedings of the 29th International Conference on Machine Learning* (cit. on p. 3).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning." In: *Nature* 521.7553, p. 436 (cit. on pp. 2, 80).

LeCun, Yann, Fu Jie Huang, Leon Bottou, et al. (2004). "Learning methods for generic object recognition with invariance to pose and lighting." In: *Computer Vision and Pattern Recognition, 2004*. Citeseer, pp. 97–104 (cit. on pp. 71, 90).

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989). "Backpropagation applied to handwritten zip code recognition." In: *Neural computation* 1.4, pp. 541–551 (cit. on pp. 4, 5).

Lee, Honglak, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng (2009). “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.” In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 609–616 (cit. on pp. 34, 104).

Leibo, Joel Z, Cyprien de Masson d’Autume, Daniel Zoran, David Amos, Charles Beattie, Keith Anderson, Antonio García Castañeda, Manuel Sanchez, Simon Green, Audrunas Gruslys, et al. (2018). “Psychlab: a psychology laboratory for deep reinforcement learning agents.” In: *arXiv preprint arXiv:1801.08116* (cit. on pp. 2, 3).

Levine, Sergey (2018). “Reinforcement learning and control as probabilistic inference: Tutorial and review.” In: *arXiv preprint arXiv:1805.00909* (cit. on p. 105).

Li, Jiwei, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky (2017). “Adversarial learning for neural dialogue generation.” In: *The 2017 Conference on Empirical Methods on Natural Language Processing.* (Cit. on p. 15).

Lillicrap, Timothy P, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton (2020). “Backpropagation and the brain.” In: *Nature Reviews Neuroscience*, pp. 1–12 (cit. on p. 2).

Liu, Ziwei, Ping Luo, Xiaogang Wang, and Xiaoou Tang (2015). “Deep Learning Face Attributes in the Wild.” In: *Proceedings of International Conference on Computer Vision (ICCV)* (cit. on p. 92).

Locatello, Francesco, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem (2019). “Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations.” In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR, pp. 4114–4124 (cit. on pp. 70, 78, 81, 86).

Marblestone, Adam H, Greg Wayne, and Konrad P Kording (2016). “Toward an integration of deep learning and neuroscience.” In: *Frontiers in computational neuroscience* 10, p. 94 (cit. on p. 2).

Mathieu, Emile, Charline Le Lan, Chris J. Maddison, Ryota Tomioka, and Yee Whye Teh (2019). “Hierarchical Representations with Poincaré Variational Auto-Encoders.” In: *CoRR* abs/1901.06033 (cit. on pp. 34, 54, 79).

Matsumoto, Narihisa, Masato Okada, Yasuko Sugase-Miyamoto, and Shigeru Yamane (2005). “Neuronal mechanisms encoding global-to-fine information in inferior-temporal cortex.” In: *Journal of Computational Neuroscience* 18.1, pp. 85–103 (cit. on pp. 24, 31, 33).

McCulloch, Warren S and Walter Pitts (1943). “A logical calculus of the ideas immanent in nervous activity.” In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133 (cit. on p. 1).

Metz, Luke, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein (2019). "Learning Unsupervised Learning Rules." In: *International Conference on Learning Representations* (cit. on p. 79).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality." In: *Advances in Neural Information Processing Systems 26*, pp. 3111–3119 (cit. on p. 54).

Miller, George (1998). *WordNet: An electronic lexical database*. MIT press (cit. on p. 58).

Mirza, Mehdi and Simon Osindero (2014). "Conditional generative adversarial nets." In: *arXiv preprint arXiv:1411.1784* (cit. on p. 78).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. (2015). "Human-level control through deep reinforcement learning." In: *Nature* 518.7540, pp. 529–533 (cit. on pp. 2, 57).

Morcos, Ari S., David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick (2018). "On the importance of single directions for generalization." In: *International Conference on Learning Representations* (cit. on p. 3).

Nagano, Yoshihiro, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama (2019). "A Wrapped Normal Distribution on Hyperbolic Space for Gradient-Based Learning." In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97, pp. 4693–4702 (cit. on pp. 34, 79).

Nickel, Maximilian and Douwe Kiela (2017). "Poincaré Embeddings for Learning Hierarchical Representations." In: *Advances in Neural Information Processing Systems 30* (cit. on pp. 34, 43, 46, 54, 59, 60).

– (2018). "Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry." In: *Proceedings of the 35th International Conference on Machine Learning*, pp. 3776–3785 (cit. on pp. 46, 54).

Okada, Masato (1996). "Notions of associative memory and sparse coding." In: *Neural Networks* 9.8, pp. 1429–1458 (cit. on pp. 31, 33).

Oord, Aaron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu (2016). "Wavenet: A generative model for raw audio." In: *arXiv preprint arXiv:1609.03499* (cit. on pp. 2, 10).

Oord, Aaron van den, Oriol Vinyals, et al. (2017). "Neural discrete representation learning." In: *Advances in Neural Information Processing Systems*, pp. 6306–6315 (cit. on p. 10).

Ovinnikov, Ivan (2019). "Poincaré Wasserstein Autoencoder." In: *CoRR* abs/1901.01427 (cit. on pp. 34, 54, 79).

Poole, Ben, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli (2016). “Exponential expressivity in deep neural networks through transient chaos.” In: *Advances in neural information processing systems*, pp. 3360–3368 (cit. on p. 2).

Quiroga, R Quian, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried (2005). “Invariant visual representation by single neurons in the human brain.” In: *Nature* 435.7045, pp. 1102–1107 (cit. on p. 3).

Radford, Alec, Luke Metz, and Soumith Chintala (2015). “Unsupervised representation learning with deep convolutional generative adversarial networks.” In: *arXiv preprint arXiv:1511.06434* (cit. on pp. 15, 57, 68).

Ravi, Sachin and Hugo Larochelle (2017). “Optimization as a model for few-shot learning.” In: *Proceedings of the 5th International Conference on Learning Representations* (cit. on p. 71).

Reed, Scott, Kihyuk Sohn, Yuting Zhang, and Honglak Lee (2014). “Learning to Disentangle Factors of Variation with Manifold Interaction.” In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2, pp. 1431–1439 (cit. on pp. 71, 90).

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). “Stochastic backpropagation and approximate inference in deep generative models.” In: *arXiv preprint arXiv:1401.4082* (cit. on pp. 10, 11, 13, 15–17, 34, 45, 52, 69, 70, 72).

Rezende, Danilo and Shakir Mohamed (2015). “Variational Inference with Normalizing Flows.” In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 1530–1538 (cit. on pp. 11, 54, 69).

Ridgeway, Karl and Michael C Mozer (2018). “Learning deep disentangled embeddings with the f-statistic loss.” In: *Advances in Neural Information Processing Systems*, pp. 185–194 (cit. on p. 86).

Rifai, Salah, Yann N Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller (2011). “The manifold tangent classifier.” In: *Advances in Neural Information Processing Systems 24*, pp. 2294–2302 (cit. on pp. 9, 28).

Rolfe, Jason Tyler (2017). “Discrete Variational Autoencoders.” In: *Proceedings of the 5th International Conference on Learning Representations* (cit. on pp. 54, 79).

Rosenblatt, Frank (1958). “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6, p. 386 (cit. on p. 1).

Roweis, Sam T. and Lawrence K. Saul (2000). “Nonlinear Dimensionality Reduction by Locally Linear Embedding.” In: *Science* 290.5500, pp. 2323–2326 (cit. on pp. 5, 74, 79).

Saito, Masaki, Eiichi Matsumoto, and Shunta Saito (2017). “Temporal generative adversarial nets with singular value clipping.” In: *2017 IEEE International Conference on Computer Vision* (cit. on p. 15).

Sala, Frederic, Chris De Sa, Albert Gu, and Christopher Re (2018). “Representation Tradeoffs for Hyperbolic Embeddings.” In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80, pp. 4460–4469 (cit. on pp. 45, 46).

Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen (2016). “Improved techniques for training gans.” In: *Advances in neural information processing systems*, pp. 2234–2242 (cit. on p. 14).

Sarkar, Rik (2012). “Low Distortion Delaunay Embedding of Trees in Hyperbolic Plane.” In: *Graph Drawing*, pp. 355–366 (cit. on p. 45).

Saxe, Andrew M, James L McClelland, and Surya Ganguli (2014). “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks.” In: *Proceedings of the 3nd International Conference on Learning Representations* (cit. on p. 34).

Saxe, Andrew M., James L. McClelland, and Surya Ganguli (2019). “A mathematical theory of semantic development in deep neural networks.” In: *Proceedings of the National Academy of Sciences* 116.23, pp. 11537–11546 (cit. on p. 69).

Schmidhuber, Jurgen (1987). *Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook*. Diploma Thesis (cit. on p. 71).

Schoenholz, Samuel S, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein (2017). “Deep information propagation.” In: *Proceedings of the 5th International Conference on Learning Representations* (cit. on p. 2).

Serban, Iulian Vlad, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio (2017). “A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues.” In: *AAAI*, pp. 3295–3301 (cit. on p. 15).

Shu, Rui, Yining Chen, Abhishek Kumar, Stefano Ermon, and Ben Poole (2020). “Weakly Supervised Disentanglement with Guarantees.” In: *International Conference on Learning Representations* (cit. on pp. 78, 89).

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. (2017). “Mastering the game of go without human knowledge.” In: *Nature* 550.7676, pp. 354–359 (cit. on p. 2).

Smolensky, Paul (1986). *Information processing in dynamical systems: Foundations of harmony theory*. Tech. rep. Colorado Univ at Boulder Dept of Computer Science (cit. on p. 10).

Sohn, Kihyuk, Honglak Lee, and Xinchen Yan (2015). “Learning Structured Output Representation using Deep Conditional Generative Models.” In: *Advances in Neural Information*

Processing Systems 28. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., pp. 3483–3491 (cit. on p. 78).

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to sequence learning with neural networks." In: *Advances in neural information processing systems*, pp. 3104–3112 (cit. on p. 5).

Tenenbaum, Joshua B., Vin de Silva, and John C. Langford (2000). "A Global Geometric Framework for Nonlinear Dimensionality Reduction." In: *Science* 290.5500, pp. 2319–2323 (cit. on pp. 5, 79).

Tishby, Naftali, Fernando C Pereira, and William Bialek (2000). "The information bottleneck method." In: *arXiv preprint physics/0004057* (cit. on p. 4).

Tomczak, Jakub M and Max Welling (2016). "Improving variational auto-encoders using householder flow." In: *arXiv preprint arXiv:1611.09630* (cit. on p. 14).

– (2017). "Improving Variational Auto-Encoders using convex combination linear Inverse Autoregressive Flow." In: *Benelearn 2017: Proceedings of the Twenty-Sixth Benelux Conference on Machine Learning*, p. 162 (cit. on pp. 14, 34).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need." In: *Advances in neural information processing systems*, pp. 5998–6008 (cit. on p. 5).

Vikram, Sharad, Matthew D Hoffman, and Matthew J Johnson (2018). "The LORACs prior for VAEs: Letting the Trees Speak for the Data." In: *CoRR* abs/1810.06891 (cit. on p. 55).

Vilnis, Luke and Andrew McCallum (2015). "Word representations via gaussian embedding." In: *Proceedings of the 3rd International Conference on Learning Representations* (cit. on pp. 53, 54, 59).

Vincent, Pascal and Yoshua Bengio (2003). "Manifold parzen windows." In: *Advances in neural information processing systems*, pp. 849–856 (cit. on pp. 5, 79).

Vondrick, Carl, Hamed Pirsiavash, and Antonio Torralba (2016). "Generating videos with scene dynamics." In: *Advances In Neural Information Processing Systems 29*, pp. 613–621 (cit. on p. 15).

Walker, Jacob, Carl Doersch, Abhinav Gupta, and Martial Hebert (2016). "An uncertain future: Forecasting from static images using variational autoencoders." In: *The 14th European Conference on Computer Vision*, pp. 835–851 (cit. on p. 15).

Whittington, James CR and Rafal Bogacz (2019). "Theories of error back-propagation in the brain." In: *Trends in cognitive sciences* (cit. on p. 2).

Wu, Zonghan, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip (2020). "A comprehensive survey on graph neural networks." In: *IEEE Transactions on Neural Networks and Learning Systems* (cit. on p. 105).

Xiao, Han, Kashif Rasul, and Roland Vollgraf (Aug. 28, 2017). *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv: `cs.LG/1708.07747 [cs.LG]` (cit. on p. 26).

Yamins, Daniel LK and James J DiCarlo (2016). "Using goal-driven deep learning models to understand sensory cortex." In: *Nature neuroscience* 19.3, p. 356 (cit. on pp. 2, 3).

Yu, Lantao, Weinan Zhang, Jun Wang, and Yong Yu (2017). "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient." In: *AAAI*, pp. 2852–2858 (cit. on p. 15).

hardmaru (2019a). *"experimental philosophy" if there's even such a thing* [*Tweet*]. URL: `https://twitter.com/status/hardmaru/1208972152428474368` (cit. on p. 1).

– (2019b). *machine learning is my favorite branch of philosophy* [*Tweet*]. URL: `https://twitter.com/status/hardmaru/1208971872815271937` (cit. on p. 1).

LIST OF PUBLICATIONS

Nagano, Yoshihiro, Ryo Karakida, and Masato Okada (accepted). "Collective Dynamics of Repeated Inference in Variational Autoencoder Rapidly Find Cluster Structure." In: *Scientific Reports*.

Nagano, Yoshihiro, Ryo Karakida, Norifumi Watanabe, Atsushi Aoyama, and Masato Okada (2016). "Input Response of Neural Network Model with Lognormally Distributed Synaptic Weights." In: *Journal of the Physical Society of Japan* 85.7, p. 074001.

Nagano, Yoshihiro, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama (2019). "A Wrapped Normal Distribution on Hyperbolic Space for Gradient-Based Learning." In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 4693–4702.

Nagano, Yoshihiro, Shiro Takagi, Yuki Yoshida, and Masato Okada (under review). "Extracting Locally Disentangled Representation by Self-Supervised Meta-Learning."